SIAM J. COMPUT.
Vol. 49, No. 6, pp. 1291–1331

A FRAMEWORK FOR EXPONENTIAL-TIME-HYPOTHESIS–TIGHT ALGORITHMS AND LOWER BOUNDS IN GEOMETRIC INTERSECTION GRAPHS*

MARK DE BERG†, HANS L. BODLAENDER‡, SÁNDOR KISFALUDI-BAK§, DÁNIEL MARX¶, AND TOM C. VAN DER ZANDEN‖

Abstract. We give an algorithmic and lower bound framework that facilitates the construction of subexponential algorithms and matching conditional complexity bounds. It can be applied to intersection graphs of similarly-sized fat objects, yielding algorithms with running time $2^{O(n^{1-1/d})}$ for any fixed dimension $d \geq 2$ for many well-known graph problems, including INDEPENDENT SET, r-DOMINATING SET for constant r, and STEINER TREE. For most problems, we get improved running times compared to prior work; in some cases, we give the first known subexponential algorithm in geometric intersection graphs. Additionally, most of the obtained algorithms are representation-agnostic, i.e., they work on the graph itself and do not require the geometric representation. Our algorithmic framework is based on a weighted separator theorem and various treewidth techniques. The lower bound framework is based on a constructive embedding of graphs into d-dimensional grids, and it allows us to derive matching $2^{\Omega(n^{1-1/d})}$ lower bounds under the exponential time hypothesis even in the much more restricted class of d-dimensional induced grid graphs.

Key words. unit disk graph, separator, fat objects, subexponential, ETH

AMS subject classifications. 68U05, 68W05, 68Q25, 05C10, 05C69

DOI. 10.1137/20M1320870

1. Introduction. Many hard graph problems that seem to require $2^{\Omega(n)}$ time on general graphs, where n is the number of vertices, can be solved in subexponential time on planar graphs. In particular, many of these problems can be solved in $2^{O(\sqrt{n})}$ time on planar graphs. Examples of problems for which this so-called *square-root phenomenon* [40] holds include INDEPENDENT SET, VERTEX COVER, HAMILTONIAN CYCLE. The great speed-ups that the square-root phenomenon offers lead to the question are there other graph classes that also exhibit this phenomenon, and is there an overarching framework to obtain algorithms with subexponential running time for these graph classes? The planar separator theorem [38, 39] and treewidth-based algorithms [18] offer a partial answer to this question. They give a general framework to obtain subexponential algorithms on planar graphs or, more generally, on H-minor free graphs. It builds heavily on the fact that H-minor free graphs have

*Received by the editors February 24, 2020; accepted for publication (in revised form) September 23, 2020; published electronically December 15, 2020. An excerpt of this article has appeared in the proceedings of STOC 2018 [5] and the article shares material with parts of Kisfaludi-Bak's thesis [34] and van der Zanden's thesis [53].

https://doi.org/10.1137/20M1320870

Funding: This work was supported by the NETWORKS project, funded by the Netherlands Organization for Scientific Research NWO under project 024.002.003. and by the ERC Consolidator Grant SYSTEMATICGRAPH (725978) of the European Research Council.

†Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands (M.T.d.Berg@tue.nl).

‡Department of Computer Science, Utrecht University, Utrecht, The Netherlands (H.L.Bodlaender@uu.nl).

§Max Planck Institute for Informatics, Saarbrücken, Germany (skisfalu@mpi-inf.mpg.de).

¶CISPA Helmholtz Center for Information Security, Saarbrücken, Germany (marx@cispa.saarland).

‖Department of Data Analytics and Digitalization, Maastricht University, The Netherlands (T.vanderZanden@maastrichtuniversity.nl).

treewidth $O(\sqrt{n})$ and, hence, admit a separator of size $(\sqrt{n})$. A similar line of work is emerging in the area of geometric intersection graphs, with running times of the form $n^{O(n^{1-1/d})}$ or, in one case, $2^{O(n^{1-1/d})}$ in the d-dimensional case [42, 46]. The main goal of our paper is to establish a framework for a wide class of geometric intersection graphs that is similar to the framework known for planar graphs, while guaranteeing the running time $2^{O(n^{1-1/d})}$.

The *intersection graph* $G[F]$ of a set F of objects in $\mathbb{R}^d$ is the graph whose vertex set is F and in which two vertices are connected when the corresponding objects intersect. *(Unit-)disk graphs*, where F consists of (unit) disks in the plane are a widely studied class of intersection graphs. Disk graphs form a natural generalization of planar graphs, since any planar graph can be realized as the intersection graph of a set of disks in the plane. In this paper we consider intersection graphs of a set F of *fat objects*, where an object $o \subseteq \mathbb{R}^d$ is α-*fat*, for some $0 < \alpha \leqslant 1$ if there are balls B_{in} and B_{out} in $\mathbb{R}^d$ such that $B_{\text{in}} \subseteq o \subseteq B_{\text{out}}$ and $\text{radius}(B_{\text{in}})/\,\text{radius}(B_{\text{out}}) \geqslant \alpha$. For example, disks are 1-fat and squares are $(1/\sqrt{2})$-fat. From now on we assume that α is an absolute constant, and often simply speak of fat objects. When dealing with arbitrarily-sized fat objects we also require the objects to be convex. Most of our results are about similarly-sized fat objects, however, and then we do not need the objects to be convex; in fact, they do not even need to be connected.[1] (A set of objects is *similarly-sized* when the ratio of the largest and smallest diameter of the objects in the set is bounded by a fixed constant.) Thus our definition of fatness for similarly-sized fat objects is very general. In particular, it does not imply that F has near-linear union complexity, as is the case for so-called locally-fat objects [2].

Several important graph problems have been investigated for (unit-)disk graphs or other types of intersection graphs [1, 8, 20, 22, 42]. However, an overarching framework that helps in designing subexponential algorithms has remained elusive. A major hurdle to obtain such a framework is that even unit-square graphs can already have arbitrarily large cliques and so they do not necessarily have small separators or small treewidth. One may hope that intersection graphs have low clique-width or rankwidth—this has proven to be useful for various dense graph classes [17, 43]—but unfortunately this is not the case even when considering only unit interval graphs [26]. One way to circumvent this hurdle is to restrict attention to intersection graphs of disks of *bounded ply* [3, 27]. This prevents large cliques, but the restriction to bounded-ply graphs severely limits the inputs that can be handled. A major goal of our work is thus to give a framework that can even be applied when the ply is unbounded.

Our first contribution: An algorithmic framework for geometric intersection graphs of fat objects. As mentioned, many subexponential results for planar graphs rely on planar separators. Our first contribution is a generalization of this result to intersection graphs of (arbitrarily-sized and convex, or similarly-sized) fat objects in $\mathbb{R}^d$. Since these graphs can have large cliques we cannot bound the number of vertices in the separator. Instead, we build a separator consisting of cliques. We then define a weight function γ on these cliques, and we define the weight of a separator as the sum of the weights of its constituent cliques C_i. This is useful since for many problems a separator can intersect the solution vertex set in $2^{O(\sum_i \gamma(|C_i|))}$ many ways, for a suitable function γ. Although we state Theorem 1.1 with the strongest bound on γ possible, in our applications it suffices to define the weight of a clique C as $\gamma(|C|) := \log(|C| + 1)$. Our theorem can now be stated as follows (see Figure 1).

[1] In the conference version of our paper we erroneously claimed that for arbitrarily-sized objects the restriction to convex objects is not necessary either.

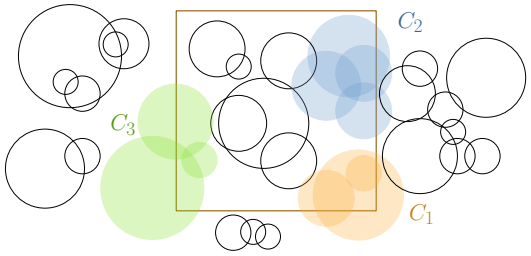

FIG. 1. *An example for Theorem* 1.1: *a disk graph with a separator partitioned into cliques* C_1, C_2, C_3.

THEOREM 1.1. *Let* $d \geqslant 2, \alpha > 0$, *and* $\varepsilon > 0$ *be constants and let* γ *be a weight function such that* $\gamma(t) = O(t^{1-1/d-\varepsilon})$. *Let* F *be a set of* n α*-fat objects in* $\mathbb{R}^d$ *that are all convex, or similarly-sized. Then the intersection graph* $G[F]$ *has a* $(6^d/(6^d+1))$*-balanced separator* F_{sep} *and a clique partition* $\mathcal{C}(F_{\text{sep}})$ *of* F_{sep} *with weight* $O(n^{1-1/d})$. *Such a separator and a clique partition* $\mathcal{C}(F_{\text{sep}})$ *can be computed in* $O(n^{d+2})$ *time if the objects have constant complexity.*

Remark 1.2. The time stated in the theorem to compute the separator is $O(n^{d+2})$. It is probably possible to reduce this, but since in our applications this does not make a difference for the final time bounds, we do not pursue this.

A direct application of our separator theorem is a $2^{O(n^{1-1/d})}$ algorithm for INDEPENDENT SET for any fixed constant d. (The dependence on d is double-exponential.) For general fat objects, only the two-dimensional case was known to have such an algorithm [41].

Our separator theorem can be seen as a generalization of the work of Fu [23] who considers a weighting scheme similar to ours. However, Fu's result is significantly less general as it only applies to unit balls and his proof is arguably more complicated. Our result can also be seen as a generalization of the separator theorem of Har-Peled and Quanrud [27] which gives a small separator for constant ply—indeed, our proof borrows some ideas from theirs.

Finally, the technique employed by Fomin et al. [20] in two dimensions has also similar qualities; in particular, the idea of using cliques as a basis for a separator can also be found there, and leads to subexponential parameterized algorithms, even for some problems that we do not tackle here.

After proving the weighted separator theorem for fat objects, we apply it to obtain an algorithmic framework for similarly-sized fat objects. Here the idea is as follows: we find a suitable clique-decomposition $\mathcal{P}$ of the intersection graph $G[F]$, contract each clique to a single vertex, and then work with the contracted graph $G_{\mathcal{P}}$ where the node corresponding to a clique C gets weight $\gamma(|C|)$. We then prove that the graph $G_{\mathcal{P}}$ has constant degree and, using our separator theorem, we prove that $G_{\mathcal{P}}$ has weighted treewidth $O(n^{1-1/d})$. (The big-O notation hides an exponential factor in d.) Moreover, we can compute a tree decomposition of this weight in $2^{O(n^{1-1/d})}$ time.

Thus we obtain a framework that gives $2^{O(n^{1-1/d})}$-time algorithms for intersection graphs of similarly sized[2] fat objects for many problems for which treewidth-based

[2]With separator-based results it is often possible to state theorems for all subgraphs of a given graph class. This is not possible in our case: taking subgraphs destroys the cliques that we rely on. Additionally, every graph class we consider contains all complete graphs, so a statement about their subgraphs would have to hold for all graphs.

TABLE 1

Summary of our results. In each case we list the most inclusive class where our framework leads to algorithms with $2^{O(n^{1-1/d})}$ running time, and the most restrictive class for which we have a matching lower bound. We also list whether the algorithm is representation-agnostic (rep.-agnostic).

Problem	Algorithm class	Rep.-agnostic	Lower bound class
INDEPENDENT SET	Convex fat	no	Unit ball, $d \geq 2$
INDEPENDENT SET	Sim. sized fat	yes	Unit ball, $d \geq 2$
r-DOMINATING SET, $r = \text{const}$	Sim. sized fat	yes	Induced grid, $d \geq 2$
STEINER TREE	Sim. sized fat	yes	Induced grid, $d \geq 2$
FEEDBACK VERTEX SET	Sim. sized fat	yes	Induced grid, $d \geq 2$
CONN. VERTEX COVER	Sim. sized fat	yes	Unit ball, $d \geq 2$ **or** induced grid, $d \geq 3$
CONN. DOMINATING SET	Sim. sized fat	yes	Induced grid, $d \geq 2$
CONN. FEEDBACK VERTEX SET	Sim. sized fat	yes	Unit ball, $d \geq 2$ **or** induced grid, $d \geq 3$
HAMILTONIAN CYCLE/PATH	Sim. sized fat	no	Induced grid, $d \geq 2$

algorithms are known. Our framework recovers and often slightly improves the best known results for several problems,[3] including INDEPENDENT SET, HAMILTONIAN CYCLE, and FEEDBACK VERTEX SET. Our framework also gives the first subexponential algorithms in geometric intersection graphs for, among other problems, r-DOMINATING SET for constant r, STEINER TREE, and CONNECTED DOMINATING SET.

Furthermore, we show that our approach can be combined with the *rank-based approach* [10], a technique to speed up algorithms for connectivity problems. Table 1 summarizes the results we obtain by applying our framework; in each case we have matching upper and lower bounds on the time complexity of $2^{\Theta(n^{1-1/d})}$ (where the lower bounds are conditional on the exponential time hypothesis (ETH)).

A desirable property of algorithms for geometric graphs is that they are *representation-agnostic*, meaning that they can work directly on the graph without a geometric representation of F. Most of the known algorithms do in fact require a representation, which could be a problem in applications, since finding a geometric representation (e.g., with unit disks) of a given geometric intersection graph is NP-hard [15], and many recognition problems for geometric graphs are $\exists\mathbb{R}$-complete [33]. Note that in the absence of a representation, some of our algorithms require that the input graphs are from the proper graph class, otherwise they might give incorrect answers.

One of the advantages of our framework is that it yields representation-agnostic algorithms for many problems. To this end we need to generalize our scheme slightly: we no longer work with a clique partition to define the contracted graph $G_{\mathcal{P}}$, but with a partition whose classes are the union of constantly many cliques. We show that such a partition can be found efficiently without knowing the set F defining the given intersection graph. Thus we obtain representation-agnostic algorithms for many of the problems mentioned above, in contrast to known results which almost all need the underlying set F as input.

Our second contribution: A framework for lower bounds under ETH. The $2^{O(n^{1-1/d})}$-time algorithms that we obtain for many problems immediately lead to the question is it possible to obtain even faster algorithms? For many problems on planar graphs, and for certain problems on ball graphs, the answer is no, assuming the ETH [29]. However, these lower bound results in higher dimensions are scarce, and often very problem specific. Our second contribution is a framework to obtain tight ETH-based lower bounds for problems on d-dimensional grid graphs (which are

[3]Note that most of the earlier results are in the parameterized setting, but we do not consider parameterized algorithms here.

a subset of intersection graphs of similarly sized fat objects). The obtained lower bounds match the upper bounds of the algorithmic framework. Our lower bound technique is based on a constructive embedding of graphs into d-dimensional grids, for $d \geqslant 3$, thus avoiding the invocation of deep results from Robertson and Seymour's graph minor theory. This *cube wiring theorem* implies that for any constant $d \geq 3$, any connected graph on m edges is the minor of the d-dimensional grid hypercube of side length $O(m^{\frac{1}{d-1}})$ (see Theorem 3.8).

As it turns out, we can easily derive the cube wiring theorem from a result of Thompson and Kung [48]. We also prove a slightly stronger version of cube wiring, which may be of independnet interest.

For $d = 2$, we give a lower bound for a customized version of the 3-SAT problem. Now, these results make it possible to design simple reductions for our problems using just three custom gadgets per problem; the gadgets model variables, clauses, and connections between variables and clauses, respectively. By invoking cube wiring or our custom satisfiability problem, the wires connecting the clause and variable gadgets can be routed in a very tight space. Giving these three gadgets immediately yields the tight lower bound in d-dimensional grid graphs (under ETH) for all $d \geq 2$. Naturally, the same conditional lower bounds are implied in all containing graph classes, such as unit-ball graphs, unit cube graphs, and also in intersection graphs of similarly sized fat objects. Similar lower bounds are known for various problems in the parameterized complexity literature [42, 8]. The embedding in [42] in particular has a denser target graph than a grid hypercube, where the "edge length" of the cube contains an extra logarithmic factor compared to ours (see Theorem 2.17 in [42]) and thereby gives slightly weaker lower bounds.

Moreover, our lower bound for Hamiltonian Cycle in induced grid graphs implies the same lower bound for Euclidean TSP, which turns out to be ETH-tight [4].

2. The algorithmic framework.

2.1. Separators for fat objects. Let F be a set of n α-fat objects in $\mathbb{R}^d$ for some constant $\alpha > 0$, and let $G[F] = (F, E)$ be the intersection graph induced by F. We say that a subset $F_{\text{sep}} \subseteq F$ is a *β-balanced separator* for $G[F]$ if $F \setminus F_{\text{sep}}$ can be partitioned into two subsets F_1 and F_2 with no edges between them and with $\max(|F_1|, |F_2|) \leqslant \beta n$. For a given decomposition $\mathcal{C}(F_{\text{sep}})$ of F_{sep} into cliques and a given weight function γ we define the *weight* of F_{sep}, denoted by $\text{weight}(F_{\text{sep}})$, as $\text{weight}(F_{\text{sep}}) := \sum_{C \in \mathcal{C}(F_{\text{sep}})} \gamma(|C|)$. Next we prove that $G[F]$ admits a balanced separator of weight $O(n^{1-1/d})$ for any cost function $\gamma(t) = O(t^{1-1/d-\varepsilon})$ with $\varepsilon > 0$. Our approach borrows ideas from Har-Peled and Quanrud [27], who show the existence of small separators for low-density sets of objects, although our arguments are significantly more involved.

Step 1: *Finding candidate separators.* Let H_0 be a minimum-size hypercube containing at least $n/(6^d + 1)$ objects from F, and assume without loss of generality that H_0 is the unit hypercube centered at the origin. Let $H_1, \ldots, H_m$ be a collection of $m := n^{1/d}$ hypercubes, all centered at the origin, where H_i has edge length $1 + \frac{2i}{m}$. Note that the largest hypercube, H_m, has edge length 3, and that the distance between the corresponding faces of consecutive hypercubes H_i and H_{i+1} is $1/n^{1/d}$.

Each hypercube H_i induces a partition of F into three subsets: a subset $F_{\text{in}}(H_i)$ containing all objects whose convex hull lies completely in the interior of H_i, a subset $F_{\partial}(H_i)$ containing all objects whose convex hull intersects the boundary ∂H_i of H_i,

and a subset $F_{\text{out}}(H_i)$ containing all objects whose convex hull lies completely in the exterior of H_i. Obviously an object from $F_{\text{in}}(H_i)$ cannot intersect an object from $F_{\text{out}}(H_i)$, and so $F_\partial(H_i)$ defines a separator in a natural way. (Note that disconnected objects could lie partly inside H_i and partly outside H_i without intersecting ∂H_i. This is the reason why we define the sets $F_{\text{out}}(H_i)$, $F_\partial(H_i)$, and $F_{\text{in}}(H_i)$ with respect to the convex hulls of the objects. If each object is connected, we can also define these sets with respect to the objects themselves.) It will be convenient to add some more objects to these separators, as follows. We call an object *large* when its diameter is at least $1/4$, and *small* otherwise. We will add all large objects that intersect H_m to our separators. Thus our candidate separators are the sets $F_{\text{sep}}(H_i) := F_\partial(H_i) \cup F_{\text{large}}$, where F_{large} is the set of all large objects intersecting H_m. We show that our candidate separators are balanced.

LEMMA 2.1. *For any $0 \leqslant i \leqslant m$ we have*

$$\max\left(|F_{\text{in}}(H_i) \setminus F_{\text{large}}|, |F_{\text{out}}(H_i) \setminus F_{\text{large}}|\right) < \frac{6^d}{6^d+1} n.$$

Proof. Consider a hypercube H_i. Because H_0 contains at least $n/(6^d+1)$ objects from F, we immediately obtain

$$\left|(F_{\text{out}}(H_i) \setminus F_{\text{large}})\right| \leqslant |F_{\text{out}}(H_0)| \leqslant |F \setminus F_{\text{in}}(H_0)| < \left(1 - \frac{1}{6^d+1}\right) n = \frac{6^d}{6^d+1} n.$$

To bound $\left|F_{\text{in}}(H_i) \setminus F_{\text{large}}\right|$, consider a subdivision of H_i into 6^d subhypercubes of edge length $\frac{1}{6}(1 + \frac{2i}{m}) \leqslant 1/2$. We claim that any subhypercube H_{sub} intersects fewer than $n/(6^d+1)$ small objects from F. To see this, recall that small objects have diameter less than $1/4$. Hence, all small objects intersecting H_{sub} are fully contained in a hypercube of edge length less than 1. Since H_0 is a smallest hypercube containing at least $n/(6^d+1)$ objects from F, H_{sub} must thus intersect fewer than $n/(6^d+1)$ objects from F, as claimed. Each object in $F_{\text{in}}(H_i)$ intersects at least one of the 6^d subhypercubes, so we can conclude that $\left|F_{\text{in}}(H_i) \setminus F_{\text{large}}\right| < \left(6^d/(6^d+1)\right)n$. ∎

Step 2: *Defining the cliques and finding a low-weight separator.* Define $F^* := F \setminus (F_{\text{in}}(H_0) \cup F_{\text{out}}(H_m) \cup F_{\text{large}})$. Note that $F_\partial(H_i) \setminus F_{\text{large}} \subseteq F^*$ for all i. We partition F^* into *size classes* F^*_s, based on the diameter of the objects. More precisely, for integers s with $1 \leqslant s \leqslant s_{\max}$, where $s_{\max} := \lceil (1 - 1/d) \log n \rceil - 2$, we define

$$F^*_s := \left\{ o \in F^* : \frac{2^{s-1}}{n^{1/d}} \leqslant \text{diam}(o) < \frac{2^s}{n^{1/d}} \right\}.$$

We furthermore define F^*_0 to be the subset of objects $o \in F^*$ with $\text{diam}(o) < 1/n^{1/d}$. Note that $2^{s_{\max}}/n^{1/d} \geqslant 1/4$, which means that every object in F^* is in exactly one size class.

Each size class other than F^*_0 can be decomposed into cliques as follows: fix a size class F^*_s, with $1 \leqslant s \leqslant s_{\max}$. Since the objects in F are α-fat for a fixed constant $\alpha > 0$, each $o \in F^*_s$ contains a ball of radius $\alpha \cdot (\text{diam}(o)/2) = \Omega(\frac{2^s}{n^{1/d}})$. Moreover, each object $o \in F^*_s$ lies fully or partially inside the outer hypercube H_m, which has edge length 3. This implies we can stab all objects in F^*_s using a set P_s of $O((\frac{n^{1/d}}{2^s})^d)$ points. Thus there exists a decomposition $\mathcal{C}(F^*_s)$ of F^*_s consisting of $O(\frac{n}{2^{sd}})$ cliques. Next we show that there exists such a decompostion for F_{large} as well.

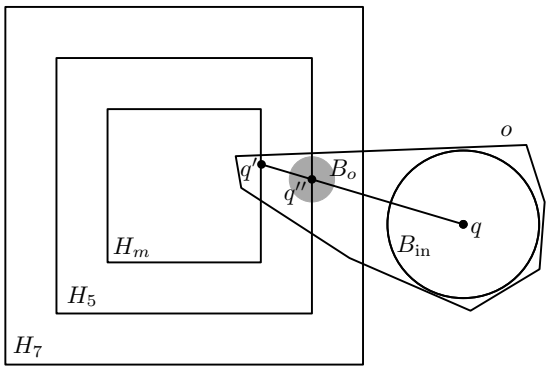

FIG. 2. *A convex fat object $o \in F_{\text{large}}$ contains a ball B_o of at least constant radius that is inside the hypercube H_7.*

LEMMA 2.2. *Let F be a set of similarly-sized fat objects or a set of arbitrarily-sized convex fat objects. Then F_{large} can be decomposed into a collection $\mathcal{C}(F_{\text{large}})$ of $O(1)$ cliques.*

Proof. Recall that H_0 is a smallest hypercube containing at least $n/(6^d + 1)$ objects, and that we assumed H_0 to be a unit hypercube centered at the origin. Let $H(t)$ denote a copy of H_0 scaled by a factor t with respect to the origin. Note that $H_0 = H(1)$ and $H_m = H(3)$.

To prove the lemma for the case of similarly-sized objects, let $d_{\min}$ and $d_{\max}$ be the minimum and maximum diameter of any of the objects in F, respectively. Since H_0 has unit size and fully contains at least one object from F, we have $d_{\min} \leqslant \sqrt{d} = O(1)$. Moreover, the objects are similarly sized and so we also have $d_{\max} = O(1)$. Because all objects in F_{large} intersect $H_m = H(3)$, they must lie completely inside $H(t^*)$ for $t^* = 3 + 2d_{\max} = O(1)$. Since $\text{diam}(o) \geqslant 1/4$ for all $o \in F_{\text{large}}$ and the objects are fat, each object contains a ball of radius $\Omega(1)$. Hence, we can stab all objects in F_{large} with a grid of $O(1)$ points inside $H(t^*)$.

Next we prove the lemma for the case where the objects in F_{large} are arbitrarily-sized but convex. It suffices to show that for any $o \in F_{\text{large}}$ there exists a ball $B_o \subseteq o$ of radius $\Theta(1)$ that is fully contained in $H(7)$. To show the existence of B_o, let $B_{\text{in}} \subseteq o$ be a ball of radius $\Theta(\text{diam}(o))$; such a ball exists because o is fat. Note that $\text{diam}(o) \geqslant 1/4$ since $o \in F_{\text{large}}$. Let q be the center of B_{in}, see Figure 2. If $q \in H(5)$ then the ball centered at q of radius $\min(\text{radius}(B_{\text{in}}), 1)$ is a ball of radius $\Theta(1)$ that lies completely inside $H(7)$ and we are done. Otherwise, let $q' \in o \cap H_m$ and let $q'' := qq' \cap \partial H(5)$ be the point where the segment qq' intersects $\partial H(5)$. We now take B_o to be the ball centered at q'' and of radius $\min\left(1, \frac{|q'q''|}{|q'q|} \cdot \text{radius}(B_{\text{in}})\right)$. By convexity of o, we have $B_o \subset o$. Moreover, $B_o \subset H(7)$. Finally, since $|q'q''| \geqslant 1$ and $q'q \leqslant \text{diam}(o)$ we have

$$\frac{|q'q''|}{|q'q|} \cdot \text{radius}(B_{\text{in}}) \geqslant \frac{1}{\text{diam}(o)} \cdot \text{radius}(B_{\text{in}}) = \Theta(1)$$

which shows $\text{radius}(B_o) = \Theta(1)$. This finishes the proof that B_o has the desired properties. ∎

The set F_0^* cannot be decomposed into few cliques since objects in F_0^* can be arbitrarily small. Hence, we create a singleton clique for each object in F_0^*. Together with

the decompositions of the size classes F_s^* and of F_{large} we thus obtain a decomposition $\mathcal{C}(F^*)$ of F^* into cliques.

Note that $\mathcal{C}(F^*)$ induces a decomposition of $F_{\text{sep}}(H_i)$ into cliques for any i. We denote this decomposition by $\mathcal{C}(F_{\text{sep}}(H_i))$. Thus, for a given weight function γ, the weight of $F_{\text{sep}}(H_i)$ is $\sum_{C\in\mathcal{C}(F_{\text{sep}}(H_i))}\gamma(|C|)$. Our goal is now to show that at least one of the separators $F_{\text{sep}}(H_i)$ has weight $O(n^{1-1/d})$, when $\gamma(t)=O(t^{1-1/d-\varepsilon})$ for some $\varepsilon>0$. To this end we will bound the total weight of all separators $F_{\text{sep}}(H_i)$ by $O(n)$. Using that the number of separators is $n^{1/d}$ we then obtain the desired result.

LEMMA 2.3. *If $\gamma(t)=O(t^{1-1/d-\varepsilon})$ for some $\varepsilon>0$ then $\sum_{i=1}^{m}\text{weight}(F_{\text{sep}}(H_i))=O(n)$.*

Proof. First consider the cliques in $\mathcal{C}(F_0^*)$, which are singletons. Since objects in F_0^* have diameter less than $1/n^{1/d}$, which is the distance between consecutive hypercubes H_i and H_{i+1}, each such object is in at most one set $F_\partial(H_i)$. Hence, its contribution to the total weight $\sum_{i=1}^{m}\text{weight}(F_{\text{sep}}(H_i))$ is $\gamma(1)=O(1)$. Together, the cliques in $\mathcal{C}(F_0^*)$ thus contribute $O(n)$ to the total weight.

Next, consider $\mathcal{C}(F_{\text{large}})$. It consists of $O(1)$ cliques. In the worst case each clique appears in all sets $F_\partial(H_i)$. Hence, their total contribution to $\sum_{i=1}^{m}\text{weight}(F_{\text{sep}}(H_i))$ is bounded by $O(1)\cdot\gamma(n)\cdot n^{1/d}=O(n)$.

Now consider a set $\mathcal{C}(F_s^*)$ with $1\leqslant s\leqslant s_{\max}$. A clique $C\in\mathcal{C}(F_s^*)$ consists of objects of diameter at most $2^s/n^{1/d}$ that are stabbed by a common point. Since the distance between consecutive hypercubes H_i and H_{i+1} is $1/n^{1/d}$, this implies that C contributes to the weight of $O(2^s)$ separators $F_{\text{sep}}(H_i)$. The contribution to the weight of a single separator is at most $\gamma(|C|)$. (It can be less than $\gamma(|C|)$ because not all objects in C need to intersect ∂H_i.) Hence, the total weight contributed by all cliques, which equals the total weight of all separators, is

$$\sum_{s=1}^{s_{\max}}\sum_{C\in\mathcal{C}(F_s^*)}(\text{weight contributed by } C)\leqslant\sum_{s=1}^{s_{\max}}\sum_{C\in\mathcal{C}(F_s^*)}2^s\gamma(|C|)$$
$$=\sum_{s=1}^{s_{\max}}\left(2^s\sum_{C\in\mathcal{C}(F_s^*)}\gamma(|C|)\right).$$

Next we wish to bound $\sum_{C\in\mathcal{C}(F_s^*)}\gamma(|C|)$. Define $n_s:=|F_s^*|$ and observe that $\sum_{s=1}^{s_{\max}}n_s\leqslant n$. Recall that $\mathcal{C}(F_s^*)$ consists of $O(n/2^{sd})$ cliques, that is, of at most $cn/2^{sd}$ cliques for some constant c. To make the formulas below more readable we assume $c=1$ (so we can omit c), but it is easily checked that this does not influence the final result asymptotically. Similarly, we will be using $\gamma(t)=t^{1-1/d-\varepsilon}$ instead of $\gamma(t)=O(t^{1-1/d-\varepsilon})$. Because γ is positive and concave, the sum $\sum_{C\in\mathcal{C}(F_s^*)}\gamma(|C|)$ is maximized when the number of cliques is maximal, namely, $\min(n_s,n/2^{sd})$, and when the objects are distributed as evenly as possible over the cliques. Hence,

$$\sum_{C\in\mathcal{C}(F_s^*)}\gamma(|C|)\leqslant\begin{cases}n_s & \text{if } n_s\leqslant n/2^{sd},\\ (n/2^{sd})\cdot\gamma\left(\frac{n_s}{n/2^{sd}}\right) & \text{otherwise.}\end{cases}$$

We now split the set $\{1,\ldots,s_{\max}\}$ into two index sets S_1 and S_2, where S_1 contains

all indices s such that $n_s \leqslant n/2^{sd}$, and S_2 contains all remaining indices. Thus

(2.1)
$$\sum_{s=1}^{s_{\max}} \left(2^s \sum_{C \in \mathcal{C}(F_s^*)} \gamma(|C|) \right) = \sum_{s \in S_1} \left(2^s \sum_{C \in \mathcal{C}(F_s^*)} \gamma(|C|) \right) + \sum_{s \in S_2} \left(2^s \sum_{C \in \mathcal{C}(F_s^*)} \gamma(|C|) \right).$$

The first term in (2.1) can be bounded by

$$\sum_{s \in S_1} \left(2^s \sum_{C \in \mathcal{C}(F_s^*)} \gamma(|C|) \right) \leqslant \sum_{s \in S_1} 2^s n_s \leqslant \sum_{s \in S_1} 2^s (n/2^{sd}) = n \sum_{s \in S_1} 1/2^{s(d-1)} = O(n),$$

where the last step uses that $d \geqslant 2$. For the second term we get

$$\begin{aligned}
\sum_{s \in S_2} \left(2^s \sum_{C \in \mathcal{C}(F_s^*)} \gamma(|C|) \right) &\leqslant \sum_{s \in S_2} \left(2^s (n/2^{sd}) \cdot \gamma\left(\frac{n_s}{n/2^{sd}} \right) \right) \\
&\leqslant \sum_{s \in S_2} \left(\frac{n}{2^{s(d-1)}} \cdot \left(\frac{n_s 2^{sd}}{n} \right)^{1-1/d-\varepsilon} \right) \\
&\leqslant n \sum_{s \in S_2} \left(\frac{n_s}{n} \right)^{1-1/d-\varepsilon} \frac{1}{2^{sd\varepsilon}} \\
&\leqslant n \sum_{s \in S_2} \left(\frac{1}{2^{d\varepsilon}} \right)^s \\
&= O(n). \qquad \square
\end{aligned}$$

We are now ready to prove Theorem 1.1.

Proof of Theorem 1.1. Each candidate separator $F_{\text{sep}}(H_i)$ is $(6^d/(6^d+1))$-balanced by Lemma 2.1. Their total weight is $O(n)$ by Lemma 2.3, and since we have $n^{1/d}$ candidates one of them must have weight $O(n^{1-1/d})$. Finding this separator can be done in $O(n^{d+2})$ time by brute force. Indeed, to find the hypercube $H_0 = [x_1, x_1'] \times \cdots \times [x_d, x_d']$ in $O(n^{d+2})$ time we first guess the object defining x_i for all $1 \leqslant i \leqslant d$, then guess the object defining x_1' (and, hence, the size of the hypercube), and finally determine the number of objects inside the hypercube. Once we have H_0, we can generate the hypercubes $H_1, \ldots, H_{n^{1/d}}$, generate the cliques as described above, and then compute the weights of the separators $F_{\text{sep}}(H_i)$ by brute force within the same time bound. $\square$

Corollary 2.4. *Let F be a set of n fat objects in $\mathbb{R}^d$ that are either convex or similarly-sized, where d is a constant. Then* Independent Set *on the intersection graph $G[F]$ can be solved in $2^{O(n^{1-1/d})}$ time.*

Proof. Let $\gamma(t) \stackrel{\text{def}}{=} \log(t+1)$, and compute a separator F_{sep} for $G[F]$ using Theorem 1.1. For each subset $S_{\text{sep}} \subseteq F_{\text{sep}}$ of independent (that is, pairwise nonadjacent) vertices we find the largest independent set S of G such that $S \supseteq S_{\text{sep}}$, by removing the closed neighborhood of S_{sep} from G and recursing on the remaining connected components. Finally, we report the largest of all these independent sets. Because a clique $C \in \mathcal{C}(F_{\text{sep}})$ can contribute at most one vertex to S_{sep}, we have that the number of candidate sets S_{sep} is at most

$$\prod_{C \in \mathcal{C}(F_{\text{sep}})} (|C|+1) = 2^{\sum_{C \in \mathcal{C}(F_{\text{sep}})} \log(|C|+1)} = 2^{O(n^{1-1/d})}.$$

Since all components on which we recurse have at most $(6^d/(6^d+1))n$ vertices, the running time $T(n)$ satisfies

$$T(n) = 2^{O(n^{1-1/d})} T\left(\frac{6^d}{6^d+1} n\right) + 2^{O(n^{1-1/d})},$$

which solves to $T(n) = 2^{O(n^{1-1/d})}$. ∎

2.2. An algorithmic framework for similarly sized fat objects. We restrict our attention to *similarly sized* fat objects. More precisely, we consider intersection graphs of sets F of objects such that, for each $o \in F$, there are balls B_{in} and B_{out} in $\mathbb{R}^d$ such that $B_{\text{in}} \subseteq F \subseteq B_{\text{out}}$, and $\text{radius}(B_{\text{in}}) = \alpha$ and $\text{radius}(B_{\text{out}}) = 1$ for some fatness constant $\alpha > 0$. The restriction to similarly sized objects makes it possible to construct a clique cover of F with the following property: if we consider the intersection graph $G[F]$ where the cliques are contracted to single vertices, then the contracted graph has constant degree. Moreover, the contracted graph admits a tree decomposition whose weighted treewidth is $O(n^{1-1/d})$. This tool allows us to solve many problems on intersection graphs of similarly sized fat objects.

Our tree-decomposition construction uses the separator theorem from the previous subsection. That theorem also states that we can compute the separator for $G[F]$ in polynomial time, provided we are given F. However, finding the separator if we are only given the graph and not the underlying set F is not easy. Note that deciding whether a graph is a unit-disk graph is already $\exists\mathbb{R}$-complete [33]. Nevertheless, we show that for similarly sized fat objects we can find certain tree decompositions with the desired properties, purely based on the graph $G[F]$.

κ-partitions, $\mathcal{P}$-contractions, and separators. Let $G = (V, E)$ be the intersection graph of an (unknown) set F of similarly sized fat objects, as defined above. The separators in the previous section use cliques as basic components. We need to generalize this slightly, by allowing connected unions of a constant number of cliques as basic components. Thus we define a *κ-partition* of G as a partition $\mathcal{P} = (V_1, \ldots, V_k)$ of V such that every partition class V_i induces a connected subgraph that is the union of at most κ cliques. Note that a 1-partition corresponds to a clique cover of G. A natural way to define the weight of a partition class V_i would be the sum of the weights of the cliques contained in it. However, it will be more convenient to define the weight of a partition class V_i to be $\gamma(|V_i|)$. Since κ is a constant, this is within a constant factor of the more natural weight.

Given a κ-partition $\mathcal{P}$ of G we define the *$\mathcal{P}$-contraction* of G, denoted by $G_{\mathcal{P}}$, to be the graph obtained by contracting all partition classes V_i to single vertices and removing loops and parallel edges. In many applications it is essential that the $\mathcal{P}$-contraction we work with has maximum degree bounded by a constant. From now on, when we speak of the degree of a κ-partition $\mathcal{P}$ we refer to the degree of the corresponding $\mathcal{P}$-contraction.

The following theorem is very similar to Theorem 1.1, but it applies only for similarly sized objects because of the degree bound on $G_{\mathcal{P}}$. The other main difference is that the separator is defined on the $\mathcal{P}$-contraction of a given κ-partition, instead of on the intersection graph G itself. The statement is purely existential; we prove a more constructive theorem in section 2.3.

THEOREM 2.5. *Let $d \geqslant 2$ and $\varepsilon > 0$ be constants and let γ be a weight function such that $\gamma(t) = O(t^{1-1/d-\varepsilon})$. Let $G = (V, E)$ be the intersection graph of a set of n similarly sized fat objects in $\mathbb{R}^d$. Suppose we are given a κ-partition $\mathcal{P}$ of G such that*

$G_{\mathcal{P}}$ has maximum degree at most Δ, where κ and Δ are constants. Then there exists a $(6^d/(6^d+1))$-balanced separator for $G_{\mathcal{P}}$ of weight $O(n^{1-1/d})$.

Proof sketch. Within each class $C \in \mathcal{P}$, we replace each object in C with the larger object $\bigcup_{o \in C} o$. Notice that these new objects are similarly sized and fat (with worse constants). The new objects define a supergraph G', which is also an intersection graph of similarly sized fat objects; moreover, $\mathcal{P}$ is a clique-partition of G'. By the condition that $G_{\mathcal{P}}$ has maximum degree at most Δ, any new object intersects at most Δ different new objects, that is, if we were to remove all duplicate objects from the family, then the resulting family would have ply at most $\Delta + 1$.

The arguments in the proof of Theorem 1.1 can be applied for G': the clique partition $\mathcal{P}^*$ is created by using stabbing points. Each class of $\mathcal{P}^*$ can contain objects from at most $\Delta + 1$ classes of $\mathcal{P}$. We can also make sure that $\mathcal{P}^*$ is a partition that is a coarsening of $\mathcal{P}$. By Theorem 1.1, the graph G' has a weighted separator (with respect to $\mathcal{P}^*$) of weight $O(n^{1-1/d})$. Since $\Delta = O(1)$ and each class of $\mathcal{P}^*$ contains objects from at most $\Delta + 1$ classes of $\mathcal{P}$, this converts into a weighted separator wrt $\mathcal{P}$ of weight $O(n^{1-1/d})$. Since G is a subgraph of G', this is also a weighted separator of G with respect to $\mathcal{P}$, and it has the desired weight. ∎

The following lemma shows that a partition $\mathcal{P}$ as needed in Theorem 2.5 can be computed even in the absence of geometric information. Such partitions can be computed in a greedy manner, as explained next.

Definition 2.6 (greedy partition). *Given a graph G, a partition $\mathcal{P}$ of $V(G)$ is a greedy partition if there is a maximal independent set S of G such that each partition class $C \in \mathcal{P}$ contains exactly one vertex v_C of S together with some neighbors of v_C.*

Lemma 2.7. *Let $d \geqslant 2$ be a constant. Then 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 greedy κ-partition $\mathcal{P}$ for which $G_{\mathcal{P}}$ has maximum degree Δ can be computed in polynomial time.*

Proof. Let $S \subseteq V$ be a maximal independent set in G (i.e., it is inclusionwise maximal). We assign each vertex $v \in V \setminus S$ to an arbitrary vertex $s \in S$ that is a neighbor of v; such a vertex s always exists since S is maximal. For each vertex $s \in S$ define $V_s := \{s\} \cup \{v \in V \setminus S : v \text{ is assigned to } s\}$. We prove that the partition $\mathcal{P} := \{V_s : s \in S\}$, which can be computed in polynomial time, has the desired properties.

Let o_v denote the (unknown) object corresponding to a vertex $v \in V$, and for a partition class V_s define $U(V_s) := \bigcup_{v \in V_s} o_v$. We call $U(V_s)$ a *union-object*. Let $\mathcal{U}_S := \{U(V_s) : s \in S\}$. Because the objects defining G are similarly sized and fat, there are balls $B_{\text{in}}(o_v)$ of radius $\alpha = \Omega(1)$ and $B_{\text{out}}(o_v)$ of radius 1 such that $B_{\text{in}}(o_v) \subseteq o_v \subseteq B_{\text{out}}(o_v)$.

Now observe that each union-object $U(V_s)$ is contained in a ball of radius 3. Hence, we can stab all balls $B_{\text{in}}(o_v)$, $v \in V_s$, using $O(1)$ points, which implies that $\mathcal{P}$ is a κ-partition for some $\kappa = O(1)$.

To prove that the maximum degree of $G_{\mathcal{P}}$ is $O(1)$, we note that any two balls $B_{\text{in}}(s)$, $B_{\text{in}}(s')$ with $s, s' \in S$ are disjoint (because S is an independent set in G). Since all union-objects $U(s')$ that intersect $U(s)$ are contained in a ball of radius 9, an easy packing argument now shows that $U(s)$ intersects $O(1)$ union-objects $U(s)$. Hence, the node in $G_{\mathcal{P}}$ corresponding to V_s has degree $O(1)$. ∎

Note that Lemma 2.7 requires the promise that the input graph is indeed an intersection graph, since otherwise the greedy partition may not be a κ-partition.

2.3. From separators to $\mathcal{P}$-flattened treewidth. Recall that a *tree decomposition* of a graph $G = (V, E)$ is a pair (T, σ), where T is a tree and σ is a mapping from the vertices of T to subsets of V called *bags*, with the following properties. Let $\text{Bags}(T, \sigma) := \{\sigma(u) : u \in V(T)\}$ be the set of bags associated with the vertices of T. Then we have (1) for any vertex $u \in V$ there is at least one bag in $\text{Bags}(T, \sigma)$ containing it; (2) for any edge $(u, v) \in E$ there is at least one bag in $\text{Bags}(T, \sigma)$ containing both u and v; (3) for any vertex $u \in V$ the collection of bags in $\text{Bags}(T, \sigma)$ containing u forms a subtree of T.

The *width* of a tree decomposition is the size of its largest bag minus 1, and the *treewidth* of a graph G equals the minimum width of a tree decomposition of G. We will need the notion of *weighted treewidth* [51]. Here each vertex has a weight, and the weighted width of a tree decomposition is the maximum over the bags of the sum of the weights of the vertices in the bag (note: without the -1). The weighted treewidth of a graph is the minimum weighted width over its tree decompositions.

Now let $\mathcal{P} = (V_1, \ldots, V_k)$ be a κ-partition of a given graph G which is the intersection graph of similarly sized fat objects, and let γ be a given weight function on partition classes. We apply the concept of weighted treewidth to $G_{\mathcal{P}}$, where we assign each vertex V_i of $G_{\mathcal{P}}$ a weight $\gamma(|V_i|)$. Because we have a separator for $G_{\mathcal{P}}$ of low weight by Theorem 2.5, we can prove a bound on the weighted treewidth of $G_{\mathcal{P}}$ using standard techniques.

LEMMA 2.8. *Let $\mathcal{P}$ be a κ-partition of a family of similarly sized fat objects such that $G_{\mathcal{P}}$ has maximum degree at most Δ, where κ and Δ are constants. Then the weighted treewidth of $G_{\mathcal{P}}$ is $O(n^{1-1/d})$ for any weight function γ with $\gamma(t) = O(t^{1-1/d-\varepsilon})$.*

Proof. The lemma follows from Theorem 2.5 by a minor variation on standard techniques—see, for example, [9, Theorem 20]. Take a separator S of $G_{\mathcal{P}}$ as indicated by Theorem 2.5. Recursively, make tree decompositions of the connected components of $G_{\mathcal{P}} \setminus S$. Take the disjoint union of these tree decompositions, add edges to make the disjoint union connected, and then add S to all bags. We now have a tree decomposition of $G_{\mathcal{P}}$. As a base case, when we have a subgraph of $G_{\mathcal{P}}$ with total weight $O(n^{1-1/d})$, then we take one bag with all vertices in this subgraph.

The weight of bags for subgraphs of $G_{\mathcal{P}}$ with r vertices fulfils $w(r) = O(r^{1-1/d}) + w(\frac{6^d}{6^d+1}r)$, which gives that the weighted width of this tree decomposition is $w(n) = O(n^{1-1/d})$. ∎

In all of our applications, we can fix $\gamma(x) = \log(x+1)$. For a partition $\mathcal{P}$, we define the *$\mathcal{P}$-flattened treewidth* of G as the weighted treewidth of $G_{\mathcal{P}}$ under the weighting $\gamma(x) = \log(x + 1)$.

A *blowup* of a vertex v by an integer t results in a graph where we replace the vertex v with a clique of size t (called the clique of v), in which we connect every vertex to the neighborhood of v. Note that when blowing up multiple vertices of a graph in succession, the resulting graph does not depend on the order of blowups.

Consider the following algorithm to compute a weighted tree decomposition of graph G with weight function w.

1. Construct an unweighted graph H by blowing up each vertex v of G by $w(v)$. Let $H(v)$ denote the vertices of H that were gained from blowing up v.
2. Compute a tree decomposition of H, denoted by (T_H, σ_H).
3. Construct a tree decomposition $(T_G \sim T_H, \sigma_G)$ using the same tree layout the following way: a vertex $v \in G$ is added to a bag if and only if the corresponding bag in T_H contains all vertices of $H(v)$.

LEMMA 2.9. *The weighted width of* (T_G, σ_G) *is at most the width of* (T_H, σ_H) *plus* 1*; furthermore, the weighted treewidth of* G *is equal to* 1 *plus the treewidth of* H*.*

Proof. The proof is a simple modification of folklore insights on treewidth; for related results see [14, 12]. The proof relies on the following well-known observation [13].

Observation 2.10. Let $W \subseteq V$ form a clique in $G = (V, E)$. Then all tree decompositions (T, σ) of G have a bag $\sigma(u) \in \text{Bags}(T, \sigma)$ with $W \subseteq \sigma(u)$.

First, we prove that (T_G, σ_G) is a tree decomposition of G. From the observation stated above, we have that for each vertex v and edge $\{v, w\}$ there is a bag in (T_G, σ_G) that contains v, respectively, $\{v, w\}$. For the third condition of tree decompositions, suppose j_2 is in T_G on the path from j_1 to j_3. If v belongs to the bags of j_1 and j_3, then all vertices in $H(v)$ belong in (T_H, σ_H) to the bags of j_1 and j_3, hence, by the properties of tree decompositions to the bag of j_2 and, hence, $v \in \sigma_G(j_2)$. It follows that the preimage of each vertex in V_G is a subtree of T_G. The total weight of vertices in a bag in (T_G, σ_G) is never larger than the size of the corresponding bag in (T_H, σ_H). This proves the first claim.

The above algorithm can also be reversed. If we take a tree decomposition (T_G, σ_G) of G, we can obtain one of H by replacing in each bag each vertex v by the clique that results from blowing up G. The size of a bag in the tree decomposition of H now equals the total weight of the vertices in G; hence the width of (T_G, σ_G) equals the weighted width of the obtained tree decomposition of H; it follows that the weighted width of (T_G, σ_G) is equal to the width of (T_H, σ_H) minus 1.

Running the algorithm on an optimal tree decomposition of H and the reverse on an optimal weighted tree decomposition of G shows that the the weighted treewidth of G is at most (resp., at least) 1 plus the treewidth of H, concluding our proof. ∎

We are now ready to prove our main theorem for algorithms.

THEOREM 2.11. *Let* $d \geqslant 2$*,* $\alpha > 0$*, and* $\varepsilon > 0$ *be constants and let* γ *be a weight function such that* $1 \leqslant \gamma(t) = O(t^{1-1/d-\varepsilon})$*. Then there exist constants* κ *and* Δ *such that for any intersection graph* $G = (V, E)$ *of an (unknown) set of* n *similarly sized* α*-fat objects in* $\mathbb{R}^d$*, there is a* κ*-partition* $\mathcal{P}$ *with the following properties:* (i) $G_{\mathcal{P}}$ *has maximum degree at most* Δ*, and* (ii) $G_{\mathcal{P}}$ *has weighted treewidth* $O(n^{1-1/d})$*. Moreover, such a partition* $\mathcal{P}$ *and a corresponding tree decomposition of weight* $O(n^{1-1/d})$ *can be computed in* $2^{O(n^{1-1/d})}$ *time.*

Proof. We use the greedy partition $\mathcal{P}$ built around a maximal independent set (Definition 2.6). By Lemma 2.8, the weighted treewidth of $G_{\mathcal{P}}$ is $O(n^{1-1/d})$.

To get a tree decomposition, consider the above partition again, with weight function $\gamma(t) = O(t^{1-1/d-\varepsilon})$. We work on the contracted graph $G_{\mathcal{P}}$; we intend to simulate the weight function by modifying $G_{\mathcal{P}}$. Let H be the graph we get from $G_{\mathcal{P}}$ by blowing up each vertex v_C by an integer that is approximately the weight of the corresponding class, more precisely, we blow up v_C by $\lceil \gamma(|C|) \rceil$. By Lemma 2.9, its treewidth (plus one) is a 2-approximation of the weighted treewidth of G (since $\gamma(t) \geq 1$). Therefore, we can run a treewidth approximation algorithm that is single exponential in the treewidth of H. We can use the algorithm from either [45] or [11] for this; both have running time $2^{O(tw(H))}|V(H)|^{O(1)} = 2^{O(n^{1-1/d})}(n\,\gamma(n))^{O(1)} = 2^{O(n^{1-1/d})}$, and provide a tree decomposition whose width is a c-approximation of the treewidth of H. From this tree decomposition we gain a tree decomposition whose weighted treewidth is a $2c$-approximation of the weighted treewidth of $G_{\mathcal{P}}$.

In particular, we get a $2c$-approximation of the $\mathcal{P}$-flattened treewidth of G. This concludes the proof. ∎

2.4. Basic algorithmic applications. In this section, we give examples of how κ-partitions and weighted tree decompositions can be used to obtain subexponential-time algorithms for classical problems on geometric intersection graphs.

First, we make the following observation about tree decompositions.

Observation 2.12. Given a κ-partition $\mathcal{P}$ and a weighted tree decomposition of $G_{\mathcal{P}}$ of width τ, there exists a nice tree decomposition of G (i.e., a "traditional," non-partitioned tree decomposition) with the property that each bag is a subset of the union of a number of partition classes, such that the total weight of those classes is at most τ.

We can do this by creating a nice version of the weighted tree decomposition of $G_{\mathcal{P}}$, and then replacing every introduce/forget bag (that introduces/forgets a class of the partition) by a series of introduce/forget bags (that introduce/forget the individual vertices). We call such a decomposition a *traditional tree decomposition*. Using such a decomposition, it becomes easy to give algorithms for problems for which we already have dynamic-programming algorithms operating on nice tree decompositions. We can reuse the algorithms for the leaf, introduce, join and forget cases, and either show that the number of partial solutions remains bounded (by exploiting the properties of the underlying κ-partition) or show that we can discard some irrelevant partial solutions.

We present several applications for our framework, resulting in $2^{O(n^{1-1/d})}$ algorithms for various problems. In addition to the Independent Set algorithm for fat objects based on our separator, we also give a representation-agnostic algorithm for similarly sized fat objects. In contrast, the state of the art algorithm [42] requires the geometric representation as input. In the rest of the applications, our algorithms work on intersection graphs of d-dimensional similarly sized fat objects; this is usually a larger graph class than what has been studied. We have representation-dependent algorithms for Hamiltonian Path and Hamiltonian Cycle; this is a simple generalization from the algorithm for unit disks that has been known before [20, 35]. For Feedback Vertex Set, we give a representation-agnostic algorithm with the same running time improvement, over a representation-dependent algorithm that works in 2-dimensional unit disk graphs [20]. For r-Dominating Set, we give a representation-agnostic algorithm for $d \geq 2$, which is the first subexponential algorithm in dimension $d \geq 3$, and the first representation-agnostic subexponential for $d = 2$ [41]. (The algorithm in [41] is for Dominating Set in unit disk graphs.) Finally, we give representation-agnostic algorithms for Steiner Tree, r-Dominating Set, Connected Vertex Cover, Connected Feedback Vertex Set, and Connected Dominating Set, which are—to our knowledge—also the first subexponential algorithms in geometric intersection graphs for these problems.

In the following, we let t refer to a node of the tree decomposition T, let $\sigma(t)$ denote the set of vertices in the bag associated with t, and let $G[t]$ denote the subgraph of G induced by the vertices appearing in bags in the subtree of T rooted at t. We fix our weight function to be $\gamma(k) = \log(k+1)$.

Theorem 2.13. *Let $\gamma(k) = \log(k+1)$. If a κ-partition and a weighted tree decomposition of width at most τ is given,* Independent Set *and* Vertex Cover *can be solved in time $2^{\kappa\tau} n^{O(1)}$.*

Proof. A well-known algorithm (see, e.g., [18]) for solving Independent Set on graphs of bounded treewidth, computes, for each bag $\sigma(t)$ and subset $S \subseteq \sigma(t)$, the maximum size $c[t, S]$ of an independent subset $\hat{S} \subset G[t]$ such that $\hat{S} \cap \sigma(t) = S$.

Let t be a node of the weighted tree decomposition of $G_{\mathcal{P}}$. Recall that the corresponding bag is $\sigma(t)$, and it consists of partition classes. Let $X_t = \bigcup_{C \in \sigma(t)} C$ be the set of vertices that occur in a partition class in $\sigma(t)$. An independent set never contains more than one vertex of a clique. Therefore, since from each partition class we can select at most κ vertices (one vertex from each clique), the number of subsets $\hat{S}$ that need to be considered is at most

$$\prod_{C \in \sigma(t)} (|C| + 1)^{\kappa} = \exp\left(\sum_{C \in \sigma(t)} \kappa \log (|C| + 1) \right) = 2^{\kappa\tau}.$$

This bound also holds for all the bags of the traditional tree decomposition that we create. Applying the standard algorithm for Independent Set on the traditional tree decomposition, using the fact that only solutions that select at most one vertex from each clique get a nonzero value, we obtain the claimed algorithm for Independent Set. Vertex Cover is simply the complement of Independent Set. ∎

Combining this result with Theorem 2.11 gives the following result.

Corollary 2.14. *Let $d \geqslant 2$ be a fixed constant, and let G be an intersection graph of similarly sized fat objects in $\mathbb{R}^d$. Then we can solve* Independent Set *and* Vertex Cover *on G in $2^{O(n^{1-1/d})}$ time, even if the geometric representation is not given.*

In the remainder of this section, because we need additional assumptions that are derived from the properties of intersection graphs, we state our results in terms of algorithms operating directly on intersection graphs. However, note that underlying each of these results is an algorithm operating on a weighted tree decomposition of the contracted graph.

To obtain the algorithm for Independent Set, we exploited the fact that we can select at most one vertex from each clique and that, thus, we can select at most κ vertices from each partition class. For Dominating Set, our bound for the treewidth is however not enough. Instead, we need the following, stronger result, which states that the weight of a bag in the decomposition can still be bounded by $O(n^{1-1/d})$, even if we take the weight to be the total weight of the classes in the bag *and* of classes at most r hops away in $G_{\mathcal{P}}$.

Theorem 2.15. *Let $d \geqslant 2, r \geqslant 1$ be constants and let $\gamma(t) = O(t^{1-1/d-\varepsilon})$ be a weight function. Then there exist constants κ, Δ, such that for any intersection graph G of n similarly sized d-dimensional fat objects there exists a κ-partition $\mathcal{P}$ and a corresponding $G_{\mathcal{P}}$ of maximum degree at most Δ, where $G_{\mathcal{P}}$ has a weighted tree decomposition with the additional property that for any node t, the total weight of the partition classes*

$$\{C \in \mathcal{P} \mid \text{ there exist } v \in C,\, C' \in \sigma(t),\, v' \in C' \text{ such that } v \text{ and } v' \text{ are at distance at most } r \text{ in } G\}$$

is $O(n^{1-1/d})$.

Proof. As per Theorem 2.11, there exist constants $\kappa, \Delta = O(1)$ such that G has a κ-partition $\mathcal{P}$ in which each class of the partition is adjacent to at most Δ other classes.

For any pair of classes $C, C' \in V(G_{\mathcal{P}})$ whose distance in $G_{\mathcal{P}}$ is at most r, we introduce a new copy of every object in C. This gives a new intersection graph G^r. Note that for each object we now have at most $1+\Delta+\Delta(\Delta-1)+\cdots+\Delta(\Delta-1)^{r-1} = c = O(\Delta^r)$ copies. We create the following κc-partition $\mathcal{P}^r$: for each class C of the original partition, create a class that contains a copy of each object of C and a copy of each object from the classes at distance at most r from C. The resulting graph G^r has at most $cn = O(n)$ vertices, and it is an intersection graph of similarly sized objects, and $G^r_{\mathcal{P}^r}$ has constant-bounded maximum degree. Therefore, we can find a weighted tree decomposition of $G^r_{\mathcal{P}^r}$ of width $O(n^{1-1/d})$ by using the machinery of section 2.3.

This decomposition can also be used as a decomposition for G and the original κ-partition $\mathcal{P}$, by replacing each partition class of $\mathcal{P}^r$ with the original partition classes contained within; this increases the width of the tree decomposition by at most a constant multiplicative factor. ∎

THEOREM 2.16. *Let $d \geqslant 2$ and $r \geqslant 1$ be constants, and let G be an intersection graph of similarly sized fat objects in $\mathbb{R}^d$. Then* DISTANCE-r-DOMINATING SET *can be solved on G in $2^{O(n^{1-1/d})}$ time.*

Proof. We first present the argument for DOMINATING SET. It is easy to see that from each partition class, we need to select at most $\kappa^2(\Delta+1)$ vertices: each partition class can be partitioned into at most κ cliques, and each of these cliques is adjacent to at most $\kappa(\Delta+1)$ other cliques. If we select at least $\kappa(\Delta+1)+1$ vertices from a clique, we can instead select only one vertex from the clique, and select at least one vertex from each neighboring clique.

We once again proceed by dynamic programming on a traditional tree decomposition (see, e.g. [18] for an algorithm solving DOMINATING SET using tree decompositions). Rather than needing just two states per vertex (in the solution or not), we need three: a vertex can be either in the solution, not in the solution and not dominated, or not in the solution and dominated. After processing each bag, we discard partial solutions that select more than $\kappa^2(\Delta+1)$ vertices from any class of the partition. Note that all vertices of each partition class are introduced before any are forgotten, so we can guarantee that we indeed never select more than $\kappa^2(\Delta+1)$ vertices from each partition class.

Whether a given vertex v outside the solution is dominated or not is completely determined by the vertices that are in its class and in neighboring classes. While the partial solution does not track this explicitly for vertices that are forgotten, by using the fact that we need to select at most $\kappa^2(\Delta+1)$ vertices from each class of the partition, and the fact that Theorem 2.15 bounds the total weight of the neighborhood of the partition classes in a bag, we see that the number of partial solutions to be considered is at most

$$\prod_{C \in N(\sigma(t))} (|C_i|+1)^{\kappa^2(\Delta+1)} = \exp\left(\kappa^2(\Delta+1) \sum_{C \in N(\sigma(t))} \log(|C|+1)\right) = 2^{O(n^{1-1/d})},$$

where the product (resp., sum) is taken over $N(\sigma(t))$, which denotes the set of partition classes C that appear in the current bag $\sigma(t)$ or are neighbors of such a class. Therefore, we can solve DOMINATING SET in $2^{O(n^{1-1/d})}$ time.

For the generalization where $r > 1$, the argument that we need to select at most $\kappa^2(\Delta+1)$ vertices from each clique still holds: moving a vertex from a clique with more than $\kappa(\Delta+1)$ vertices selected to an adjacent clique only decreases the distance

to any vertices it helps cover. The dynamic programming algorithm needs, in a partial solution, to track at what distance from a vertex in the solution each vertex is. This, once again, is completely determined by the solution in partition classes at distance at most r; the number of such cases we can bound using Theorem 2.15. ∎

2.5. Connectivity problems and the rank-based approach. It is possible to combine our framework with the rank-based approach [10] to obtain $2^{O(n^{1-1/d})}$-time algorithms for problems with connectivity constraints, avoiding the extra logarithmic factor associated with tracking connectivity constraints in a naïve way. To illustrate this, we now give an algorithm for Steiner Tree. We consider the following variant of Steiner Tree:

Steiner Tree
Input: A graph $G = (V, E)$, a set of terminal vertices, $K \subseteq V$, and integer s.
Question: Decide if there is a vertex set $X \subseteq V$ of size at most s, such that $K \subseteq X$, and X induces a connected subgraph of G.

The proof requires the following lemma.

Lemma 2.17. *Let $\mathcal{P}$ be a κ-partition of a graph G, where $G_{\mathcal{P}}$ has maximum degree Δ. Let A be a clique of a κ-sized clique partition of a class $C \in \mathcal{P}$. Suppose X is a minimal solution for* Steiner Tree *(i.e., no proper subset X' of X is also a solution). Then X contains at most $\kappa(\Delta+1)$ vertices from A that are not also in K. Any minimal solution thus contains at most $\kappa^2(\Delta+1)$ vertices (that are not also in K) from each partition class.*

Proof. Consider the following process: to every vertex $v \in (A \cap X) \setminus K$ we greedily assign a neighbor $u \in X \setminus A$ such that u is adjacent to v and u is not adjacent to any other previously assigned neighbor. We call such a neighbor a *private neighbor*. We repeat this process until every vertex $v \in (A \cap X) \setminus K$ has been assigned a private neighbor.

If at some point it is not possible to assign a private neighbor to a vertex $v \in (A \cap X) \setminus K$, then v has no neighbors in $X \setminus A$ that are not adjacent to a private neighbor of some vertex $v' \in (A \cap X) \setminus K$. Then either v has no neighbors in X outside A, or all such neighbors are already adjacent to some previously assigned private neighbor. In the former case all neighbors of v are already adjacent (since they are in the clique A) and X remains connected after removing v. In the latter case, X also remains connected after removing v, since any neighbor (in X) of v can be reached from some other vertex $v' \in (A \cap X) \setminus K$ through its private neighbor. Both cases contradict our assumption that X is a minimal solution, so every vertex can be assigned a private neighbor.

We now note that since the neighborhood of A can be covered by at most $\kappa(\Delta+1)$ cliques, this gives us an upper bound on the number of private neighbors that can be assigned and thus bounds the number of vertices that can be selected from any partition class. ∎

Theorem 2.18. *Let $d \geqslant 2$ be a constant, and let G be an intersection graph of similarly sized fat objects in $\mathbb{R}^d$. Then* Steiner Tree *can be solved on G in $2^{O(n^{1-1/d})}$ time.*

Proof. The algorithm again works by dynamic programming on a traditional tree decomposition. The leaf, introduce, join, and forget cases can be handled as they are in the conventional algorithm for Steiner Tree based on tree decompositions; see, e.g., [10]. However, after processing each bag, we can reduce the number of partial

solutions that need to be considered by exploiting the properties of the underlying κ-partition.

To this end, we first need a bound on the number of vertices that can be selected from each class of the κ-partition $\mathcal{P}$.

The algorithm for Steiner Tree presented in [10] is for the weighted case, but we can ignore the weights by setting them to 1. A partial solution is then represented by a subset $S \subseteq \sigma(t)$ (representing the intersection of the partial solution with the vertices in the bag $\sigma(t)$), together with an equivalence relation on S (which indicates which vertices are in the same connected component of the partial solution).

By Lemma 2.17 we select at most $\kappa^2(\Delta+1)$ vertices from each partition class, so we can discard partial solutions that select more than this number of vertices from any partition class. Then the number of subsets S considered is at most

$$\prod_{C\in\sigma(t)} (|C|+1)^{\kappa^2(\Delta+1)} = \exp\left(\kappa^2(\Delta+1)\cdot \sum_{C\in\sigma(t)} \log(|C|+1)\right) \leqslant \exp\left(\kappa^2(\Delta+1)n^{1-1/d}\right).$$

For any such subset S, the number of possible equivalence relations is $2^{\Theta(|S|\log|S|)}$. However, the rank-based approach [10] provides an algorithm called *reduce* that, given a set of equivalence relations[4] on S, outputs a representative set of equivalence relations of size at most $2^{|S|}$. Thus, by running the reduce algorithm after processing each bag, we can keep the number of equivalence relations considered bounded by $2^{O(|S|)}$.

Since $|S| = O(\kappa^2(\Delta+1)n^{1-1/d})$ (we select at most $\kappa^2(\Delta+1)$ vertices from each partition class and each bag contains at most $O(n^{1-1/d})$ partition classes), for any subset S, the rank-based approach guarantees that we need to consider at most $2^{O(\kappa^2(\Delta+1)n^{1-1/d})}$ representative equivalence classes of S. ∎

Theorem 2.19. *Let $d \geqslant 2$ be a constant, and let G be an intersection graph of similarly sized fat objects in $\mathbb{R}^d$. Then* Maximum Induced Forest *(and* Feedback Vertex Set*) can be solved in $2^{O(n^{1-1/d})}$ time in G.*

Proof. We once again proceed by dynamic programming on a traditional tree decomposition corresponding to the weighted tree decomposition of $G_{\mathcal{P}}$ of width τ, where $\mathcal{P}$ is a κ-partition, and the maximum degree of $G_{\mathcal{P}}$ is at most Δ. We describe the algorithm for Maximum Induced Forest, which also gives an algorithm for Feedback Vertex Set as it is simply its complement.

Using the rank-based approach with Maximum Induced Forest requires some modifications to the problem, since the rank-based approach is designed to get maximum connectivity, whereas in Maximum Induced Forest, we aim to "minimize" connectivity (i.e., avoid creating cycles). To overcome this issue, the authors of [10] add a special universal vertex v_0 to the graph (increasing the width of the decomposition by 1) and ask (to decide if a maximum induced forest of size k exists in the graph) whether we can delete some of the edges incident to v_0 such that there exists an induced, connected subgraph, including v_0, of size $k+1$ in the modified graph that has exactly k edges. Essentially, the universal vertex allows us to arbitrarily glue together the trees of an induced forest into a single (connected) tree. This thus reformulates the problem such that we now aim to find a connected solution.

The main observation that allows us to use our framework, is that from each clique we can select at most 2 vertices (otherwise, the solution would become cyclic), and that thus, we only need to consider partial solutions that select at most 2κ

[4] What we refer to as "equivalence relation" is called a "partition" in [10].

vertices from each partition class. The number of such subsets is at most $2^{O(\kappa n^{1-1/d})}$. Since we only need to track connectivity among these 2κ vertices (plus the universal vertex), the rank-based approach allows us to keep the number of equivalence relations considered single-exponential in $\kappa n^{1-1/d}$. Thus, we obtain a $2^{O(\kappa n^{1-1/d})} n^{O(1)}$-time algorithm. ∎

Additional problems. Our approach gives $2^{O(n^{1-1/d})}$-time algorithms on geometric intersection graphs of d-dimensional similarly sized fat objects for almost any problem with the property that the solution (or the complement thereof) can only contain a constant (possibly depending on the "degree" of the cliques) number of vertices of any clique. We can also use our approach for variations of the following problems, that require the solution to be connected:

- Connected Vertex Cover and Connected Dominating Set: these problems may be solved similarly to their normal variants (which do not require the solution to be connected), using the rank-based approach to keep the number of equivalence classes considered single-exponential. In the case of Connected Vertex Cover, the complement is an independent set, therefore the complement may contain at most one vertex from each clique. In the case of Connected Dominating Set, it can be shown that each clique can contain at most $O(\kappa^2\Delta)$ vertices from a minimum connected dominating set.
- Connected Feedback Vertex Set: the algorithm for maximum induced forest can be modified to track that the complement of the solution is connected, and this can be done using the same connectivity-tracking equivalence relation that keeps the solution cycle-free.

Theorem 2.20. *For any constant dimension $d \geqslant 2$,* Connected Vertex Cover, Connected Dominating Set, *and* Connected Feedback Vertex Set *can be solved in time $2^{O(n^{1-1/d})}$ on intersection graphs of similarly sized d-dimensional fat objects.*

Hamiltonian cycle. Our separator theorems imply that Hamiltonian Cycle and Hamiltonian Path can be solved in $2^{O(n^{1-1/d})}$ time on intersection graphs of similarly sized d-dimensional fat objects. However, in contrast to our other results, this requires that a geometric representation of the graph is given. Given a geometric representation, we can compute a 1-partition $\mathcal{P}$, where $G_{\mathcal{P}}$ has constant degree. It can be shown that a cycle/path only needs to use at most two edges between each pair of cliques (see, e.g., [32, 35]) and that we can obtain an equivalent instance with all but a constant number of vertices removed from each clique. Our separator theorem implies that this graph has treewidth $O(n^{1-1/d})$, and Hamiltonian Cycle and Hamiltonian Path can then be solved using dynamic programming on a tree decomposition.

Theorem 2.21. *For any constant dimnsion $d \geqslant 2$,* Hamiltonian Cycle *and* Hamiltonian Path *can be solved in time $2^{O(n^{1-1/d})}$ on the intersection graph of similarly sized d-dimensional fat objects which are given as input.*

Note that the geometric representation is only needed to ensure that we can find a 1-partition $\mathcal{P}$ such that $G_{\mathcal{P}}$ has constant degree. Without the geometric representation the complexity of computing such a 1-partition is unknown, and a challenging open question.

3. The lower-bound framework. The goal of this section is to provide a general framework to exclude algorithms with running time $2^{o(n^{1-1/d})}$ in intersection

graphs. To get the strongest results, we show our lower bounds where possible for a more restricted graph class, namely, subgraphs of d-dimensional induced grid graphs. Induced grid graphs are intersection graphs of unit balls, so they are a subclass of intersection graphs of similarly sized fat objects. We need to use a different approach for $d = 2$ than for $d > 2$; this is because of the topological restrictions introduced by planarity. Luckily, the difference between $d = 2$ and $d > 2$ is only in the need of two different "embedding theorems"; when applying the framework to specific problems, the same gadgetry works both for $d = 2$ and for $d > 2$. In particular, in $\mathbb{R}^2$, constructing crossover gadgets is not necessary with our framework. To apply our framework, we need a graph problem $\mathcal{P}$ on grid graphs in $\mathbb{R}^d$, $d \geqslant 2$. Suppose that $\mathcal{P}$ admits a reduction from 3-SAT using constant size variable and clause gadgets and a wire gadget, whose size is a constant multiple of its length. Then the framework implies that $\mathcal{P}$ has no $2^{o(n^{1-1/d})}$-time algorithm in d-dimensional grid graphs for all $d \geq 2$, unless ETH fails. We remark that such gadgets can often be obtained by looking at classical NP-hardness proofs in the literature, and introducing minor tweaks if necessary.

3.1. Lower bounds in two dimensions. To prove lower bounds in two-dimensional grids, we introduce an intermediate problem.

For an integer n, let $[n] = \{1, \ldots, n\}$. We denote by $G^2(n_1, n_2)$ the two-dimensional grid graph with vertex set $[n_1] \times [n_2]$. We say that a graph H is *embeddable* in $G^2(n_1, n_2)$ if it is a topological minor of $G^2(n_1, n_2)$, i.e., if H has a subdivision that is a subgraph of $G^2(n_1, n_2)$. Finally, for a given 3-CNF formula ϕ, its *incidence graph* G_ϕ is the bipartite graph on its variables and clauses, where a variable vertex and a clause vertex are connected by an edge if the variable appears in the clause.

We say that a CNF formula ϕ is a $(3, 3)$-CNF formula if all clauses in ϕ have size at most 3 and each variable occurs at most 3 times.[5] Note that in such formulas the number of clauses and variables is within a constant factor of each other. The $(3, 3)$-SAT problem asks to decide the satisfiability of a $(3, 3)$-CNF formula.

PROPOSITION 3.1. *There is no $2^{o(n)}$ algorithm for $(3, 3)$-SAT unless ETH fails.*

Proof. By the sparsification lemma of Impagliazzo, Paturi and Zane [30], satisfiability on 3-CNF formulas with n variables and $\Theta(n)$ clauses has no $2^{o(n)}$ algorithm under the ETH. Let ϕ be such a formula. If a variable v occurs $k > 3$ times in ϕ, then we can replace v with a new variable at each occurrence. Call these new variables v_i $(i = 1, \ldots, k)$. Now, add the following clauses to the formula:

$$(v_1 \vee \neg v_2) \wedge (v_2 \vee \neg v_3) \wedge \cdots \wedge (v_{k-1} \vee \neg v_k) \wedge (v_k \vee \neg v_1). \tag{3.1}$$

It is easy to see that the resulting formula is a $(3, 3)$-CNF formula of $O(n)$ variables and clauses, and it can be created in polynomial time from the initial formula. Next, we argue that the new formula is satisfiable if and only if ϕ is satisfiable. If ϕ is satisfiable, then the new formula is also satisfied by the assignment, where for each v we set all the variables v_i to be equal to v. If the new formula is satisfiable, then for each v the added clauses can only be satisfied if $v_1 = v_2 = \cdots = v_k$; therefore we can set $v = v_1$ and such an assignment will satisfy all the original clauses. Consequently, a $2^{o(n)}$ algorithm for $(3, 3)$-SAT would also give a $2^{o(n)}$ algorithm to evaluate the satisfiability of ϕ, which contradicts ETH. ∎

[5] Crucially, we allow clauses of size 2, as formulas with clause size exactly 3 and at most 3 occurrences per variable are trivially satisfiable [50].

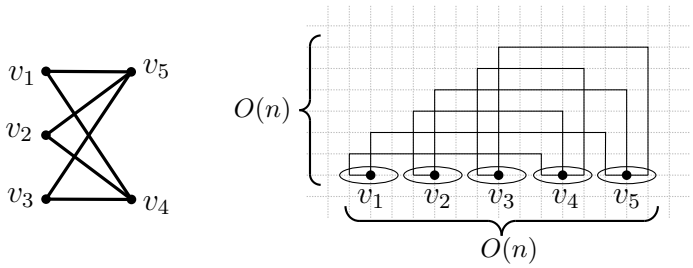

FIG. 3. *Left: The incidence graph of the formula* $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$. *Right: A drawing* $\mathcal{D}_\phi$ *of the incidence graph on the grid. Each edge is a grid path, the main part of which is a* $\sqcap$ *shape with a unique integer height.*

Our intermediate problem, GRID EMBEDDED SAT, asks to determine the satisfiability of a $(3,3)$-CNF formula whose incidence graph is embedded in an $n \times n$ grid:

GRID EMBEDDED SAT
Input: A $(3,3)$-CNF formula ϕ together with an embedding of its incidence graph G_ϕ in $G^2(n,n)$.
Question: Is there a satisfying assignment?

THEOREM 3.2. GRID EMBEDDED SAT *has no* $2^{o(n)}$ *algorithm under ETH.*

Proof. Consider a $(3,3)$-CNF formula ϕ. As a first step, we generate a grid drawing $\mathcal{D}_\phi$ in $\mathbb{R}^2$ of the incidence graph of ϕ the following way. Assign the point $(3i, 0)$ to vertex v_i of G_ϕ, as depicted in Figure 3, and add horizontal grid segments to its left (resp., left and right) to vertices of degree 2 and 3; this way each vertex in G_ϕ of degree k is assigned to a *group* of k consecutive grid points. Finally, for each edge of G_ϕ, we add two vertical segments and a horizontal segment in a $\sqcap$ shape that connects two points corresponding to the group of its endpoints; the height of this segment for edge j is set to j. This drawing assigns a unique grid point to each vertex of G_ϕ and a grid path to each edge (with intersections).

Next, we need to planarize ϕ. To this end, we use a modified version of the crossover gadget from Lichtenstein's classical planar 3-SAT reduction [37]. The crossover gadget is built on a constant size 3-CNF formula with two pairs of special variables, a, a' and b, b'. The incidence graph of the formula is planar and has maximum degree six; see Figure 4. The incidence graph has a planar embedding, where the vertices corresponding to the special variables occur in the order a, b, a', b' around the unbounded face; furthermore, the degree of a and b within this graph is two. The added variables and clauses of the gadget ensure that in all satisfying assignments $a = a'$ and $b = b'$. One uses the gadget the following way: suppose two edges, starting at variable vertices u and v are crossing in our drawing. Then we identify u and a, and also v and b. On the other side of the crossing, we add the new variable vertices a' and b'; finally, the crossing itself gets replaced by the above-mentioned embedding of the gadget.

Note that this modification can at most double the degree of each original variable vertex, and the maximum degree of newly introduced variable vertices is six. Therefore, the resulting formula ϕ' is planar, and it has degree at most six. For these high degree vertices we introduce a degree-decreasing gadget, depicted in Figure 5, where the new formulas (unlabeled vertices in the figure) are the same as in (3.1) for

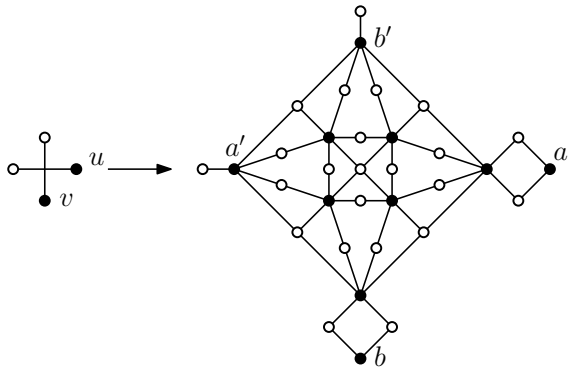

FIG. 4. *Lichtenstein's crossing gadget.*

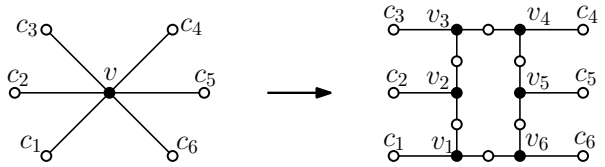

FIG. 5. *A low degree gadget to help grid embeddings.*

$k = 6$. After introducing these new variables and clauses, we get a planar $(3, 3)$-CNF formula ϕ''. Due to the nature of our modifications the formula ϕ'' is satisfiable if and only if ϕ is satisfiable.

Using the earlier drawing $\mathcal{D}_\phi$ we can easily produce a grid drawing for $G_{\phi''}$. Clearly there is a constant c_1 such that one can draw our degree decreasing gadget of size six inside, and there is a constant c_2 such that one can draw the crossing gadget (with our degree decreasing gadget added where necessary) in a grid of size $c_2 \times c_2$. By switching the grid underneath to a grid with $2\max(c_1, c_2)$ times the density, we can introduce these gadgets at the crossings, and also add the degree-decreasing gadget in place of new high degree variable vertices. This results in a grid drawing of $G_{\phi''}$, where the grid has size $O(n) \times O(n)$ and, therefore, the whole construction has $O(n^2)$ vertices. The algorithm to obtain this embedding runs in polynomial time. This completes our reduction. ∎

Note that GRID EMBEDDED SAT is solvable in $2^{O(n)}$ time, since it reduces to PLANAR SAT on $O(n^2)$ variables and clauses, which in turn has an algorithm with running time $2^{O(\sqrt{t})}$, where t is the number of variables and clauses (see, e.g., [52]).

3.2. Lower bounds in higher dimensions—cube wiring. For a vector $\mathbf{n} := (n_1, \ldots, n_d)$ in $\mathbb{Z}_+^d$, let $\mathsf{Box}_d(\mathbf{n}) = [n_1] \times \cdots \times [n_d]$. Let $G^d(\mathbf{n})$ be the graph whose vertex set $V(G)$ is $\mathsf{Box}_d(\mathbf{n})$, and where $\mathbf{x}, \mathbf{y} \in V(G)$ are connected if and only if they are at distance 1 in $\mathbb{R}^d$. The integer points of $\mathbb{R}^d$ can be divided into parallel *layers*. The layer at "height" $h \in \mathbb{Z}$ is defined as $\ell(h) = \{\mathbf{x} \in \mathbb{Z}^d \mid x_d = h\}$. Let $\mathrm{Emb}^h : \mathbb{R}^{d-1} \to \mathbb{R}^d$ be the function that maps $\mathbb{R}^{d-1}$ into $\ell(h)$ as follows: $\mathrm{Emb}^h(x_1, \ldots, x_{d-1}) = (x_1, \ldots, x_{d-1}, h)$.

In what follows, $\mathbf{n}$ denotes a $(d-1)$-dimensional vector, and we will often denote d-dimensional vectors as a concatenation of a $d-1$ and a 1-dimensional vector. For example, $G^d(\mathbf{n}, h)$ denotes the grid graph for the vertex set $[n_1] \times \cdots \times [n_{d-1}] \times [h]$.

Let P, Q be equal-size subsets of $\mathsf{Box}(\mathbf{n})$. Let M be a perfect matching of the graph $G_{P\times Q} := (P \cup Q, P \times Q)$. We say that M *can be wired in* $G^d(c\mathbf{n}, h)$, where c and h are positive integers, if there are vertex-disjoint paths in $G^d(c\mathbf{n}, h)$ that connect $\mathrm{Emb}^1(\mathbf{p})$ to $\mathrm{Emb}^h(\mathbf{q})$ for all $(\mathbf{p}, \mathbf{q}) \in M$. Note that $G^d(c\mathbf{n}, h)$ consists of h layers, each of which is a copy of $\mathsf{Box}(c\mathbf{n})$ at a different height.

We will refer to the embedding in $\mathbb{R}^d$ of the path representing a pair $(\mathbf{p}, \mathbf{q})$ as a *wire*.

Let k be a positive integer, and consider a point set $P \subseteq \mathbb{Z}^{d-1}$. The set P is *k-spaced* if there is an integer $0 \leqslant r < k$ such that for any $x = (x_1, \ldots, x_{d-1}) \in P$ we have $x_i \equiv r \mod k$ for all $i = 1, \ldots, d-1$.t

THEOREM 3.3 (cube wiring theorem). *Let $d \geq 3$, $\mathbf{n} \in \mathbb{Z}_+^{d-1}$, and let P and Q be two equal-size 2-spaced subsets of $\mathsf{Box}_{d-1}(2\mathbf{n})$. Let M be a perfect matching in $G_{P\times Q} = (P \cup Q, P \times Q)$. Then M can be wired in $G^d(2\mathbf{n}, h)$, where $h = O(\sum_{i=1}^{d-1} n_i)$.*

Our original (and qualitatively different) proof for a slightly different formulation can be found in Kisfaludi-Bak's thesis [34]. As it turns out, we can also derive the above theorem from the following result of Thompson and Kung [49] that concerns sorting on parallel processors that are connected into a $(d-1)$-dimensional grid cube. In their setting, a set of processors are connected in a $(d-1)$-dimensional hypercube. In a time step, some disjoint pairs of neighboring processors can exchange their content. In the *sorting problem*, each processor receives a number as input, and the goal is to sort the numbers so that their ordering coincides with the lexicographical order of the processors using a small number of time steps. Note that the time corresponds to the last axis in our setting. Their main result can be stated the following way.

THEOREM 3.4 (Thompson and Kung [49], paraphrased). *The sorting problem on processors connected in a grid $\mathbf{n}$ can be solved in $O(\sum_{i=1}^{d-1} n_i)$ time.*

To prove the cube wiring theorem, one just needs to define a wire for each number that tracks its location in the grid through the procedure. For a number that is at position $(x_1, \ldots, x_{d-1})$ at time t, we associate the point $2x_1, 2x_2, \ldots, 2x_{d-1}, 3t$, and ensure that it is on its wire. It is sufficient to show that the exchange of neighboring numbers can be done in a $2 \times 2 \times \cdots \times 2 \times 3$ grid cube. In fact, it is sufficient to show that the exchange is possible in a $2 \times 2 \times 3$ grid box, as the same method can be used in higher dimensions. For exchanging elements $(2x_1, 2x_2, 3t)$ and $(2(x_1 + 1), 2x_2, 3t)$, we need to get the wire of the first element to $(2(x_1 + 1), 2x_2, 3(t + 1))$ and of the second to $(2x_1, 2x_2, 3(t + 1))$. See Figure 6 for an illustration of such an exchange. The wire of the first follows the path

$$(2x_1, 2x_2, 3t), (2x_1, 2x_2, 3t + 1), (2x_1, 2x_2 + 1, 3t + 1), (2x_1 + 1, 2x_2 + 1, 3t + 1),$$
$$(2(x_1 + 1), 2x_2 + 1, 3t + 1), (2(x_1 + 1), 2x_2 + 1, 3t + 2),$$
$$(2(x_1 + 1), 2x_2, 3t + 2), (2(x_1 + 1), 2x_2, 3(t + 1)).$$

For the second, we use the wire

$$(2(x_1 + 1), 2x_2, 3t), (2(x_1 + 1), 2x_2, 3t + 1), (2x_1 + 1, 2x_2, 3t + 1),$$
$$(2x_1 + 1, 2x_2, 3t + 2), (2x_1, 2x_2, 3t + 2), (2x_1, 2x_2, 3(t + 1)).$$

A rotation of this wiring is used for a neighboring wire pair $(2x_1, 2x_2, 3t)$ and $(2x_1, 2(x_2 + 1), 3t)$. Notice that the created wires remain vertex disjoint. This concludes the proof of Theorem 3.3.

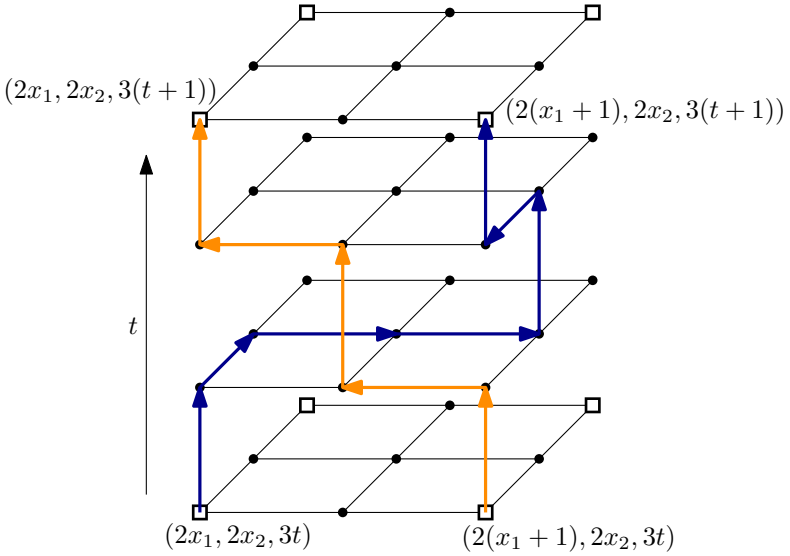

FIG. 6. *Exchanging two neighboring wires in a* $2 \times 2 \times 3$ *grid box.*

3.3. Strengthening the cube wiring theorem. We now turn to giving a stronger version of cube wiring. Note that this stronger version is not necessary for the lower bounds to hold, but it may be necessary in other applications.

THEOREM 3.5 (strong cube wiring theorem). *There exists a constant* c *such that for any* $d \geq 3$ *the following hold. Let* $\mathbf{n} \in \mathbb{Z}_+^{d-1}$*, and let* P *and* Q *be two equal-size* 2*-spaced subsets of* $\mathsf{Box}_{d-1}(2\mathbf{n})$*. Let* M *be a perfect matching in* $G_{P\times Q} = (P \cup Q, P \times Q)$*. Then* M *can be wired in* $G^d(c\mathbf{n}, cn_{\max})$*, where* $n_{\max}$ *is the largest coordinate of* $\mathbf{n}$*.*

The proof is based on the following lemma.

LEMMA 3.6. *If a matching can be wired in* $G^d((n_1, \ldots, n_{d-1}), h)$*, then it can also be wired in* $G^d((3n_1, n_2, \ldots, n_{d-1}), \lceil h/3 \rceil + 2n_1)$*.*

Proof. The idea is to add artificial turns into the wiring, and create the snake-like structure seen in Figure 7 that has roughly one-third the height of the original wiring. The wiring in three dimensions is created as depicted in the figure. Note that for each wire point we only introduce changes along the first and last coordinate x_1 and x_d, and all other wire points retain their original coordinates $x_2, \ldots, x_{d-1}$.

Let W be some set of wire points in the starting layer $\ell(1)$. We introduce elbows that bend around the x_1 axis, that is, it brings the wires from the "bottom" face of a box to an adjacent face. More precisely, for a wire w with wire point $(x_1, \ldots, x_{d-1}, 1)$ in $\ell(1)$, we first increase the last coordinate until we reach $x'_d = n_1 - x_1 + 1$, then increase along the first coordinate until we reach $x'_1 = n_1 + 1$. Doing this for all the wires in W creates a right elbow. One can also create a left elbow analogously. The elbow keeps the wires disjoint and preserves ordering: the wire points at the beginning of the elbow can be mapped to to the wire points at the end of the elbow by a rotation of angle $\pi/2$.

We can build a wiring based on the original wiring G the following way. Decompose G into four distinct wirings, by cutting the wires at their wire points of height $\lceil h/3 \rceil$, $\lceil 2h/3 - n_1 \rceil$, and $h - n_1$. Let G_1, G_2, G_3, and G_4 be the resulting partial wirings. To construct the new wiring, we start with G_1 and keep it unchanged. Then

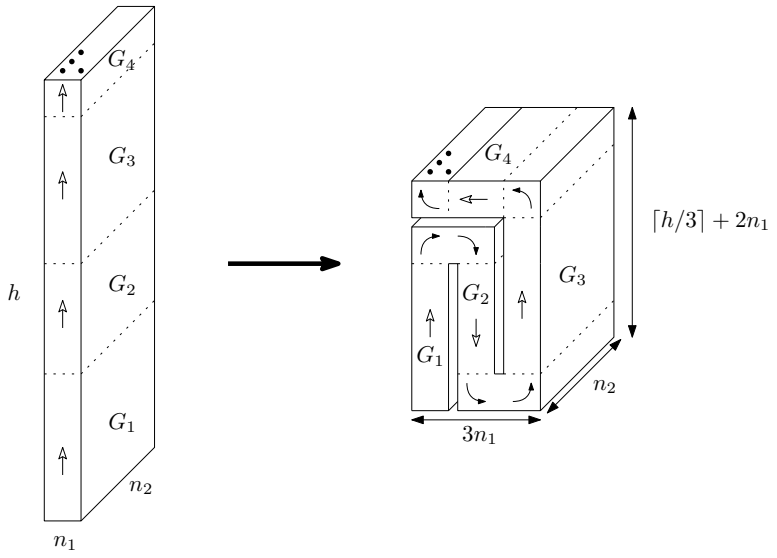

FIG. 7. *Decreasing the height of a wiring by adding elbows. We decompose the wiring into four parts, and add elbows as required (denoted by bent arrows).*

we introduce two right elbows in succession, and attach to that G_2, which is rotated in the x_1x_d plane by π. Next, we attach two left elbows in succession, and add a translate of G_3. Next, we add a left elbow, and the wiring G_4 that is rotated by $\pi/2$ in the x_1x_d plane. Notice that the resulting box has the required size: the new height is $\lceil h/3 \rceil + 2n_1$, and the width along the first coordinate has tripled from n_1 to $3n_1$; the width of the bounding box is the same along all other axes. Finally, notice that the wire points at the end of the wiring are in the same position of the box as they were in the original wiring. ∎

We are now ready to prove the strong cube wiring theorem.

Proof of Theorem 3.5. Let G be the wiring that is given by the cube wiring theorem. The wiring has base $\mathsf{Box}(2n_1, \ldots, 2n_{d-1})$ and height $h < c'dn_{\max}$ for some constant c'. Without loss of generality, assume that $n_{\max} = n_1 \geqslant n_2 \geqslant \cdots \geqslant n_{d-1}$ and that each n_i and h are powers of three. This can be achieved by rounding all of them up to the nearest power of three so that G has base $\mathsf{Box}(6n_1, \ldots, 6n_{d-1})$.

We now apply Lemma 3.6 to each of the coordinates $x_1, \ldots, x_{d-1}$ in succession. The resulting box has base $\mathsf{Box}(18n_1, \ldots, 18n_{d-1})$ and its height is

$$h' = \frac{h}{3^{d-1}} + \frac{2n_1}{3^{d-2}} + \frac{2n_2}{3^{d-3}} + \cdots + \frac{2n_{d-1}}{3^0} \leqslant \frac{h}{3^{d-1}} + 2n_{\max} \sum_{j=0}^{d-2} \frac{1}{3^j} < \frac{h}{3^{d-1}} + 3n_{\max}.$$

Notice that for all $d \geqslant 3$ we have that $3^{d-1} > d$, therefore,

$$h' < \frac{h}{3^{d-1}} + 3n_{\max} < \frac{c'dn_{\max}}{3^{d-1}} + 3n_{\max} < (c' + 3)\, n_{\max}.$$

Picking $c = \max(18, (c' + 3))$ concludes the proof. ∎

Remark 3.7. The 2-spaced requirement in both the original and the strong version of cube wiring was added for technical reasons: one can rearrange the point sets P and Q to be tightly packed in $\mathrm{Emb}^1(\mathbf{n})$ and $\mathrm{Emb}^h(\mathbf{n})$, respectively [34, Lemma 7.6].

3.4. Minors in grids. We describe a corollary of strong cube wiring in terms of graph minors. Recall that a graph G is a minor of a graph G' if there is a way to get from G' to G by some sequence of edge contractions, edge deletions, and vertex deletions. While the theorem itself is not used elsewhere in the paper, we think that it is interesting in its own right.

THEOREM 3.8. *There exists a constant c such that for all $d \geq 3$, any graph with m edges and no isolated vertices is the minor of the d-dimensional grid hypercube of side length $cm^{\frac{1}{d-1}}$.*

Proof. Let G be an arbitrary graph with m edges. We expand all vertices v of G into a path P_v of length $\deg_G(v)$, that is, replace v with a path P_v of length $\deg(v)$, where each vertex of P_v is adjacent to a single neighbor of v. We also subdivide each original edge $e = uv$ of G by two new vertices, w_{eu} (adjacent to u) and w_{ev} (adjacent to v); let G' be the graph that we end up with after these modifications. Let $P \stackrel{\text{def}}{=} \bigcup_{v \in V} V(P_v)$ and $Q \stackrel{\text{def}}{=} \{w_{ev} \mid e \in E(G) \text{ and } v \in e\}$.

There is a constant c' such that we can find an isomorphism ϕ_P from $G'[P]$ into a subgraph of $\Gamma = G_{d-1}((c'm^{1/(d-1)}, \ldots, c'm^{1/(d-1)}), c'm^{1/(d-1)})$ and, similarly, an isomorphism ϕ_Q from $G'[Q]$ into a subgraph of Γ. By the strong cube wiring theorem, there are vertex-disjoint paths connecting $\phi_P(P)$ to $\phi_Q(Q)$ in $cm^{1/(d-1)}$ layers with the matching $M = E(G') \cap (P \times Q)$, where c is a constant. Since $E(G') = G'[P] \cup G'[Q] \cup M$, this shows that G' is a minor of the grid hypercube. We also have that G is a minor of G', therefore G is a minor of the grid hypercube of side length $cm^{\frac{1}{d-1}}$. ∎

We could also consider *topological minors* instead. We would like to find an edge subdivision of some given graph G in a grid cube. In this case we are forced to bound the degree of G with $2d$, since the maximum degree of the grid is an upper bound on the maximum degree of its topological minors. As in the proof above, we need to use a grid of side length at least $c'n^{\frac{1}{d-1}}$ for some $(2d)^{1/d} < c' = O(1)$ in order to fit the starting points of the at most $2dn$ wires into a single layer. Then invoking the strong cube wiring yields the following theorem.

THEOREM 3.9. *There exists a constant c such that for all $d \geq 3$, any graph with n vertices and maximum degree $2d$ is the topological minor of the d-dimensional grid hypercube of side length $cn^{\frac{1}{d-1}}$.*

4. Applying the lower-bound framework. In order to construct reductions for our problems, we can often reuse gadgetry from classical NP-completeness proofs. The simplest approach would be to start with a problem on planar graphs, and try to create a grid graph based on that. Unfortunately, this approach is not sufficient for ETH-tight lower bounds for the following reason. Drawing a planar graph (even of maximum degree 3) on n vertices may require a $\Theta(n) \times \Theta(n)$ grid. Usually, the ETH-based lower bound for the starting planar problem is of the form $2^{\Omega(\sqrt{n})}$. Trying to realize the planar graph as a grid graph results in a graph of size n^2, since connecting distant vertices requires us to subdivide the edge with $\Omega(n)$ grid points. Therefore the resulting lower bound would be of the form $2^{\Omega(n^{1/4})}$, which is not ETH-tight.

Instead, it is usually required to maintain a grid drawing carefully in a grid of size $O(n) \times O(n)$. In our reductions, we will either start with GRID EMBEDDED SAT, or with an arbitrary (3,3)-CNF formula and a specific grid drawing of its incidence graph (with crossings), similar to what we have done in the proof of Lemma 3.2.

Recall that our goal is to prove the lower bounds in the most restricted graph class possible. Thus, where possible we aim to get a lower bound in induced grid graphs. There are two cases where we do not succeed in obtaining such a lower bound.

1. Independent Set and Vertex Cover are solvable in polynomial time on bipartite graphs, because they are equivalent to matching [36] and, therefore, can be found using a bipartite matching algorithm [28]. Since d-dimensional grid graphs are bipartite, the lower bounds can only be achieved in some larger graph class. Hence, for Independent Set and Vertex Cover we will prove our lower bounds for unit-ball graphs. Regardless, the general strategy remains the same; we can use the same type of gadgetry and realize the constructed wires by mimicking the grid-embedded drawing or cube wiring.
2. In order to prove lower bounds in two-dimensional grid graphs, one needs to be able to create gadgetry that has maximum degree 4, and other desirable properties. Such gadgets are often not known in the literature, and in the case of Connected Vertex Cover and Connected Feedback Vertex Set, we are not able to create such gadgets. Instead, we end up proving the lower bounds for unit-disk graphs, while still having the lower bound for d-dimensional grid graphs when $d \geqslant 3$. In the case of Connected Vertex Cover, we are also able to prove the lower bound for (noninduced) grid graphs in two dimensions.

A key step in many of these reductions is refinement. A *k-refinement* of a drawing $\mathcal{D} \subset \mathbb{R}^d$ inside a grid is obtained by scaling the drawing by a factor of k. This means that an axis-parallel grid segment in the drawing becomes an axis-parallel grid segment whose length is a multiple of k. If $\mathcal{D}$ is a drawing of a grid graph, then by subdividing each segment of the k-refinement using $k-1$ inner grid points, we get an induced grid graph. If we say that a drawing or a grid is refined without specifying k, then it means that we introduce some refinement for some large enough constant $k \in \mathbb{N}_+$ that is chosen so that certain conditions hold.

4.1. Dominating set. We prove the following lower bound for Dominating Set.

Theorem 4.1. *Let $d \geqslant 2$ be a fixed constant. Then there is no $2^{o(n^{1-1/d})}$ algorithm for* Dominating Set *in induced grid graphs of dimension d, unless ETH fails.*

Proof. Given an input formula ϕ, we will replace each grid point in the embedding of G_ϕ that corresponds to a variable of ϕ by a *variable gadget*, each grid point that corresponds to a clause by a *clause gadget*, and each wire (that is a path in the grid connecting a variable point to a clause point) by a *wire gadget*. Next we describe these gadgets, and how they are connected to form the construction. This will be followed by a description of how the construction can be realized as a two- and higher-dimensional grid graph.

Our variable gadget is a cycle of length 12 with an "ear" of the same size, as depicted in Figure 8. We number the vertices of the cycle from 0 to 11. The idea is to represent setting the variable to true by putting vertices with index $\equiv 1 \pmod 3$ into the dominating set; if the variable is false, we select vertices with index $\equiv 0 \pmod 3$ instead. The role of the ear is to ensure that in a minimum dominating set, one of these two scenarios is forced.

The wire gadgets are simple paths that have a length (i.e., number of edges) that is congruent to 1 modulo three. The clause gadget is a single vertex; see Figure 9.

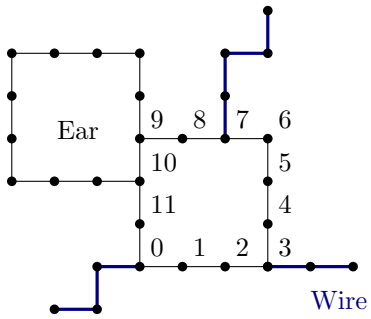

FIG. 8. *Variable gadget for* DOMINATING SET. *In blue it is shown where the wire gadgets are attached for a variable that occurs twice as a negative literal (corresponding to wires attached to vertices* 0 *and* 3*) and once as a positive literal (wire attached to vertex* 7*).*

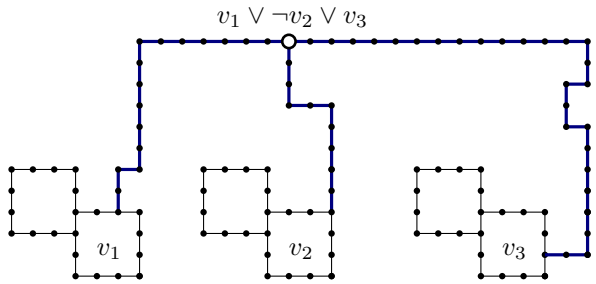

FIG. 9. DOMINATING SET *gadgetry for the formula* $(v_1 \vee \neg v_2 \vee v_3)$. *The wire connected to* v_3 *has a detour to ensure that it has length* $\equiv 1 \pmod 3$.

A wire that corresponds to a positive literal x_i will start at vertex $1, 4$, or 7 of the variable gadget of x_i, and ends at the corresponding clause vertex. For negative literals, we start at a vertex of index $\equiv 0 \pmod 3$, i.e., at vertex $0, 3$, or 6 instead. Note that selecting $k-1$ internal vertices from a wire of length $3k+1$ can dominate all internal vertices of the wire. Moreover, if the literal is true (i.e., its starting vertex in the variable cycle is in the dominating set), then selecting every third vertex on the wire will additionally dominate the clause vertex at the other end.

From each variable cycle, we must select at least four vertices to move into our dominating set, and at least three more vertices from the ear are necessary. Among the inner vertices of a wire of length $3k+1$, we have at least $k-1$ vertices in the dominating set.

Consider a dominating-set instance corresponding to a formula on n variables with a drawing that has w wires, where the wires have altogether ℓ edges. The resulting grid graph has dominating set size at least $7n + \frac{\ell - 4w}{3}$. It can be verified that this is attainable if and only if the formula is satisfied. See [7] for a similar, but more detailed argument.

Two-dimensional grid graphs. Given an instance of GRID EMBEDDED SAT, that is, a $(3,3)$-SAT formula ϕ and a grid embedded drawing $\mathcal{D}$ of G_ϕ, we need to create a grid graph which incorporates the above gadgets. This can be done by taking a 10-refinement of $\mathcal{D}$; this way, we can add the variable gadgets without overlap or unwanted induced edges, and we also have space to adjust the wire length where necessary to ensure that each wire length is $\equiv 1 \pmod 3$. This transformation can

be done in polynomial time, and the result is an induced grid graph drawn in an $O(n) \times O(n)$ grid, so the resulting induced grid graph has $O(n^2)$ vertices. Therefore, by Theorem 3.2, DOMINATING SET has no $2^{o(\sqrt{n})}$ algorithm in induced grid graphs unless ETH fails.

Higher-dimensional grid graphs. We start with a $(3,3)$-SAT formula ϕ. We place each of the above variable gadgets together with the first inner vertex of the connected wires in a $9 \times 9 \times \cdots \times 9$ small $(d-1)$-dimensional hypercube. These small hypercubes are then packed into a $(d-1)$-dimensional facet of a d-dimensional hypercube of side length $O(n^{\frac{1}{d-1}})$. The clause gadgets along with the last inner vertices of each wire are placed similarly in the opposing facet of the d-dimensional hypercube. Applying the strong cube wiring theorem to the first and last inner vertices of the wires that have been placed in the opposing facets, we can place each wire inside the hypercube, by increasing the side length by a constant factor. Finally, we adjust the wires so that all of their lengths are $\equiv 1 \pmod 3$.

The construction fits in a hypercube of side length $O(n^{\frac{1}{d-1}})$, and the number of vertices in this induced grid graph is $O(n^{\frac{d}{d-1}})$. Thus, a $2^{o(|V|^{1-1/d})}$ algorithm for DOMINATING SET would translate into a $2^{o((n^{\frac{d}{d-1}})^{1-1/d})} + \mathrm{poly}(n) = 2^{o(n)}$ algorithm for $(3,3)$-SAT, contradicting ETH. ∎

4.2. Vertex cover and independent set. It is well known that VERTEX COVER and INDEPENDENT SET are solvable in polynomial time on bipartite graphs by using an augmenting-path algorithm. Hence, these problems are also solvable in polynomial time in d-dimensional grids. Consequently, we need a slightly broader graph class for this reduction. The *augmented d-dimensional grid* for $d \geq 2$ is defined as the infinite d-dimensional grid graph together with the edges

$$\big((x_1, x_2, \ldots, x_d), (x_1+1, x_2+1, x_3, \ldots, x_d)\big), \qquad \big((x_1, \ldots, x_d) \in \mathbb{Z}^d\big).$$

In other words, the augmented d-dimensional grid is obtained from the regular d-dimensional grid by adding certain "diagonals" on two-dimensional faces of the grid cells.

The augmented d-dimensional grid is a unit-ball graph. To see this, let $\phi : \mathbb{R}^d \to \mathbb{R}^d$ be the linear transformation

$$\phi(x_1, \ldots, x_d) = \left(\frac{(1+\sqrt{2})x_1 + (1-\sqrt{2})x_2}{2\sqrt{2}}, \frac{(1-\sqrt{2})x_1 + (1+\sqrt{2})x_2}{2\sqrt{2}}, x_3, \ldots, x_d \right),$$

i.e., it pushes points closer to the hyperplane $x_2 = -x_1$. Then the intersection graph of balls of radius $1/2$ with centers $\phi(\mathbb{Z}^d)$ is the augmented grid. See Figure 10 for an illustration. Indeed, the distance between $\phi(x_1, \ldots, x_d)$ and $\phi(x_1+1, x_2+1, x_3, \ldots, x_d)$ is $1/\sqrt{2}$.

The *d-dimensional augmented grid graphs* are defined as subgraphs of the augmented d-dimensional grid. We usually consider *induced* augmented grid graphs, which are induced subgraphs of the augmented grid. Note that induced augmented grid graphs form a subclass of d-dimensional unit-ball graphs. Instead of proving the result for unit-ball graphs, we prove the following stronger statement.

THEOREM 4.2. *For any $d \geq 2$,* VERTEX COVER *and* INDEPENDENT SET *on induced augmented d-dimensional grid graphs have no $2^{o(n^{1-1/d})}$ algorithm, unless ETH fails.*

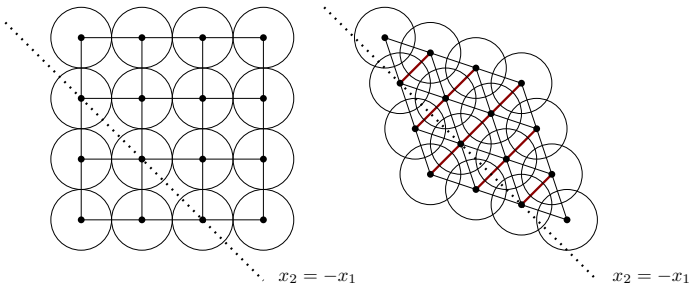

FIG. 10. *Realizing the augmented grid as an intersection graph of equal radius balls. The new edges introduced by the transformation are red.*

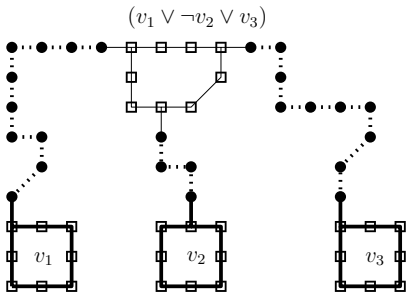

FIG. 11. *Gadgetry for the formula* $(v_1 \vee \neg v_2 \vee v_3)$.

Proof. The complement of an independent set is a vertex cover and vice versa, so it is sufficient to give a reduction for VERTEX COVER. We will use a reduction from GRID EMBEDDED SAT. Let ϕ be a $(3,3)$-CNF formula, and let G_ϕ be its incidence graph. Similarly to our DOMINATING SET gadgetry, we use a cycle as variable gadget. The literals are represented by paths of odd lengths; see Figure 11. The variable gadget for a variable v_i is a cycle of length eight with vertices $v_i(0), \ldots, v_i(7)$, where the literal edges formerly incident to v are now connected to a cycle vertex of even index for positive and of odd index for negative literals (see Figure 11). For a clause c_i that has three literals, the gadget is a cycle of length nine with vertices $c_i(0), \ldots, c_i(8)$, and we connect the wires at vertices $c_i(0), c_i(3), c_i(6)$. For clauses that have exactly two literals, we use an edge as clause gadget, and connect each wire to different endpoints. We can eliminate clauses of size one in a preprocessing step. The wire gadgets are simple paths of odd length.

Given a satisfying assignment to ϕ, we can create a vertex cover the following way. For true variables, we select vertices of even degree from the variable cycle, and for false variables, we select the vertices of odd degree. On the wires, we select every second vertex starting from the variable cycle; in case of a true literal, this means that we do not select the first inner vertex adjacent to the variable cycle, nor the endpoint in the clause gadget, while for false literals we do select both of these. Since this is a true assignment, at least one literal of each clause of size three is true; let $k \in 0, 3, 6$ be the index at which the wire of a true literal connects to the gadget of the clause c_i. Then we can select vertices $c_i(k+1), c_i(k+3), c_i(k+4), c_i(k+6), c_i(k+7)$ (where the indices are interpreted modulo 9) from the clause cycle. This covers every edge of the cycle, and also the last edges of the other wires. For a clause of size 2, we select an endpoint of the clause segment that corresponds to a false literal, or if both literals

are true, then an arbitrary endpoint. For a construction with ν variables, γ clauses of three literals, γ' clauses of two literals, and κ inner vertices on the literal paths, this gives a vertex cover of size $s \stackrel{\text{def}}{=} 4\nu + 5\gamma + \gamma' + \kappa/2$.

Suppose now that we are given a vertex cover S of size s. A vertex cover must contain at least four vertices of each variable cycle, at least five vertices per clause cycle (as introduced for clauses with three literals), and at least one vertex for each clause segment (as introduced for clauses with two literals). It is also easy to check that from a wire of $2k+1$ edges, the vertex cover must contain at least k inner vertices. It follows that S is a cover where all of these inequalities are tight; each variable cycle has to contain exactly 4 vertices of S, each clause cycle contains exactly 5 vertices of S, each clause segment contains 1 vertex of S, and among the inner vertices of each wire of length $2k+1$, there are exactly k vertices from S.

On a variable cycle, in order to cover the cycle with four vertices, all vertices of S need to have odd indices or they all need to have even indices. Let us set a variable to true if and only if the corresponding variable cycle has its even index vertices in S. We show that every clause is satisfied by this assignment. Consider first a clause with three literals. We claim that for any vertex cover of size exactly five within the clause cycle, there is at least one vertex of index divisible by three that is not in S. Suppose the contrary: the clause cycle of c_i satisfies $c_i(0), c_i(3), c_i(6) \in S$. Then S is disjoint from at least one of the sets $\{c_i(1), c_i(2)\}, \{c_i(4), c_i(5)\}, \{c_i(7), c_i(8)\}$, thus S fails to cover at least one edge among $c_i(1)c_i(2), c_i(4)c_i(5), c_i(7)c_i(8)$, which is a contradiction. It follows that the last edge of at least one of the wires connecting to the clause cycle is covered by the last inner vertex of the wire. By the size bound, we know that this wire has every second inner vertex in S, so it follows that the starting vertex of the wire in the variable gadget has to be in S, which means that the corresponding literal is true. A somewhat simpler argument shows that clauses of size 2 are also satisfied by this assignment.

Therefore, there is a vertex cover of size s if and only if the original formula is satisfiable.

Next, we need to insert these gadgets into a refined version of either the 2-dimensional grid embedding or the cube wiring. We regard this refined grid as a subgraph of the augmented grid. Using the diagonals of the augmented grid, the odd-length clause cycles can be realized. We can also enforce the parity condition on the wires by incorporating some diagonals; we introduce small local detours on the wires that have even length after the refinement to make their length odd. (Figure 11 has two wires with local parity adjustments.) ∎

4.3. Connected vertex cover. In this section we prove the following theorem.

THEOREM 4.3.

- CONNECTED VERTEX COVER *has no* $2^{o(\sqrt{n})}$ *algorithm in two-dimensional grid graphs*[6] *or in unit-disk graphs, unless ETH fails.*
- *Let* $d \geqslant 3$. *Then* CONNECTED VERTEX COVER *has no* $2^{o(n^{1-1/d})}$ *algorithm in induced* d*-dimensional grid graphs, unless ETH fails.*

We apply ideas from a reduction by Garey and Johnson [24] to make our gadgetry for CONNECTED VERTEX COVER from our original VERTEX COVER gadgets. The key step of their reduction converts a planar graph of maximum degree three into a planar graph of maximum degree four in such a way that a vertex cover of the original graph corresponds to a connected vertex cover of the constructed graph. See Lemma

[6] This refers to subgraphs of the grid that are not necessarily induced.

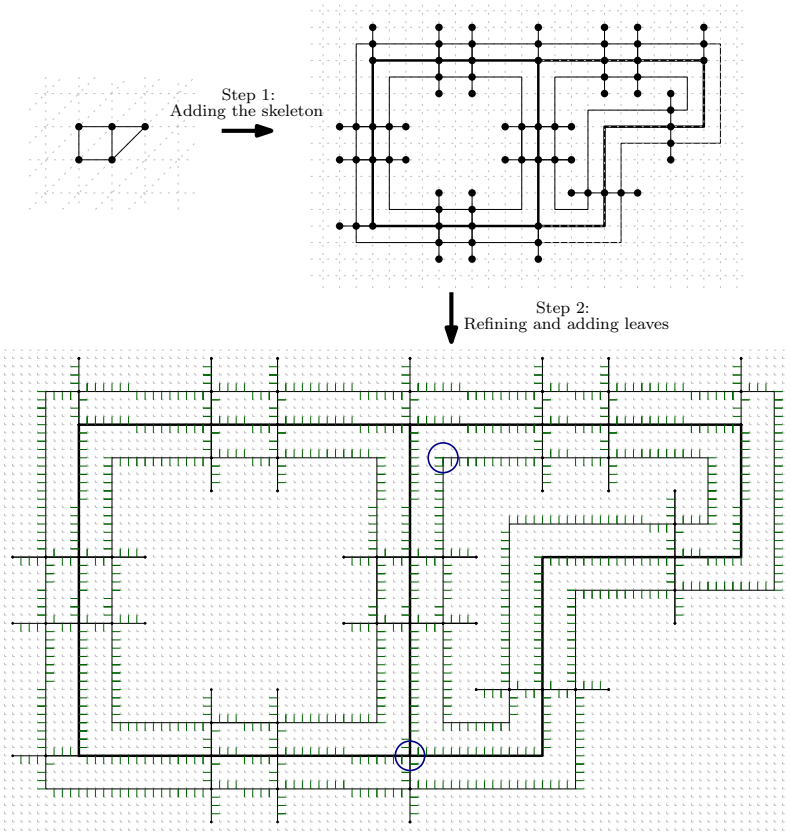

FIG. 12. *Two transformations starting from a small augmented grid graph, resulting in a (non-augmented) grid graph. The leaves can be added without conflicts, even around degree four vertices and corners (in blue circles).*

2 in their paper, which effectively adds a *skeleton* to the graph. We illustrate this on a five-vertex augmented grid graph instead of on our actual construction; see the first step of Figure 12.

By adding the skeleton to our VERTEX COVER construction, we get a planar graph of maximum degree four, which has a connected vertex cover of size k if and only if the original graph has a vertex cover of some specific size k'. Starting from our previous augmented grid embedding, the edges of this planar graph can be drawn as grid paths in a refinement of the VERTEX COVER drawing. (Note that we avoid the diagonals with these paths, and only use grid edges in the drawing.) We call this grid drawing $\mathcal{D}_\phi$. We use the following simple trick [19] to make an equivalent instance that is a grid graph.

Observation 4.4. Let $e = uv$ be an edge of a graph G that has at least two edges. Let G' be the graph that we get if we subdivide e using a vertex w_e, and add a leaf w'_e that is connected to w_e (i.e., $V(G') = V(G) \cup \{w_e, w'_e\}$, $E(G') = \big(E(G) \setminus \{uv\}\big) \cup \{uw_e, w_e v, w_e w'_e\}$). Then G has a connected vertex cover of size k if and only if G' has a connected vertex cover of size $k+1$.

We refine $\mathcal{D}_\phi$ by a factor of four; this way each old edge becomes a grid path of length at least four. We subdivide each edge by adding all the grid points that lie on its

grid path as vertices, and we add leaves to all of these new vertices. This corresponds to applying Observation 4.4 multiple times; therefore, we get an equivalent instance. Note that we need to show that these leaves can be added into the current grid drawing without trying to assign the same grid point to two different leaves. (We call such double assignments *conflicts*.) Consider first the neighborhoods of *interesting vertices*, that is, in neighborhoods of vertices of degree four and corners. Notice that we can choose the leaves in a manner (see Figure 12) that avoids the conflicts. For all other vertices, the leaves cannot introduce any conflict, as for any pair of leaves a, b, we have that the neighbor of a and the neighbor of b are already assigned to grid points that have no shared neighbors. Since we only used constantly many refinements, the resulting grid graph is drawn in an $O(n) \times O(n)$ grid.

We have shown how to modify the two-dimensional lower-bound construction for Vertex Cover to obtain an instance of Connected Vertex Cover, such that the original instance of Vertex Cover has a solution of size k if and only if the resulting instance of Connected Vertex Cover has a solution of some specific size k''. Consequently, there is no $2^{o(\sqrt{n})}$ algorithm for Connected Vertex Cover in two-dimensional grid graphs unless ETH fails.

Remark 4.5. Let V be the vertex set of the starting Vertex Cover construction, and let W and L denote the newly introduced subdividing and leaf vertices, respectively. Let G be the constructed graph itself. Note that W is a minimum (not connected) vertex cover of G, since each edge of the original vertex cover construction has been subdivided at least once. An easy argument shows that there is a minimum connected vertex cover which contains W, therefore, a minimum connected vertex cover is essentially the smallest set $S \supseteq W$ that induces a connected subgraph of G.

Note that we can realize our construction above as a unit-disk graph. The disk centers are the same as in the grid graph, but we use disks of radius 1/2. Moreover, we shift the disk centers corresponding to leaf vertices by 1/3 or 1/4 towards the neighboring disk's center, with the constraint that leaves added to neighboring vertices get a different shift.

When $d \geqslant 3$, we can do the same modifications. By being careful with adding the leaves we can even get a d-dimensional induced grid graph. Indeed, it is easy to avoid conflicts and unwanted induced edges between leaves that are attached to neighbors of interesting vertices. (The most challenging case is vertices of degree four, but these have the property that the four neighboring edges lie in the same 2-flat, so by placing the leaves outside this 2-flat we can avoid conflicts between them.) As for the straight paths of length at least four connecting these vertices, the leaves can be adjusted on the paths so that the leaves attached to the first and last inner vertex are pointing in the desired direction, as required by the interesting vertex at the endpoint.

4.4. Some easy consequences. This section sketches further ETH-based lower bounds of the form $2^{\Omega(n^{1-1/d})}$ for several problems.

Steiner Tree. We apply a 2-refinement to our connected vertex cover construction from section 4.3. We then subdivide every edge with the new grid point in the middle, and define the set of terminals to be these new vertices. The nonterminal vertices of a Steiner tree in this graph is a connected vertex cover in the original graph and the other way around. Notice that due to the refinement, the resulting graph is an induced grid graph even in the two-dimensional case.

Connected Dominating Set. We use a classical reduction by Clark, Colbourn, and Johnson [16] from Planar Connected Vertex Cover to Grid Connected

DOMINATING SET (see Theorem 5.1 in [16]). We apply this reduction to our VERTEX COVER construction from section 4.2. We get an induced grid graph embedded in an $O(n)\times O(n)$ grid. We can divide the construction into constant size variable and clause gadgets, and wire gadgets of size proportional to their length, and use these gadgets for the higher-dimensional reduction, similar to what we did at the end of section 4.1.

FEEDBACK VERTEX SET. We observe that subdividing an edge in a FEEDBACK VERTEX SET instance leads to an equivalent instance. Take our VERTEX COVER construction from section 4.2, and add a triangle to each edge, that is, for each edge uv we add a new unique vertex w_{uv}, and the edges uw_{uv} and uw_{uv}. These triangles ensure that at least one endpoint of the original edge has to be in the feedback vertex set. Moreover, the original graph has a vertex cover of size k if and only if the new graph has a feedback vertex set of size k. This results in a planar graph of degree at most 6. Our wires become triangle chains (of degree at most 4), and using subdivisions we can realize this wire gadget as an induced grid graph.

For vertices of degree more than 4, we use the degree reduction gadget by Speckenmeyer [47], which gives us constant size planar graphs that can be put in place of vertices of degree 5 and 6. These planar graphs can be drawn in an $O(1) \times O(1)$ grid, which can be turned into an induced grid graph of constant size using subdivisions. We introduce a refinement so that we can insert these gadgets as necessary.

4.5. Connected feedback vertex set. The goal of this section is to prove the following.

THEOREM 4.6.

(i) CONNECTED FEEDBACK VERTEX SET *has no* $2^{o(\sqrt{n})}$ *algorithm in unit-disk graphs, unless ETH fails.*

(ii) *Let* $d \geqslant 3$. *Then* CONNECTED FEEDBACK VERTEX SET *has no* $2^{o(n^{1-1/d})}$ *algorithm in induced* d*-dimensional grid graphs, unless ETH fails.*

Proof. We use our CONNECTED VERTEX COVER construction as a starting point, but replace each leaf added in Observation 4.4 with a *leaf cycle*: for a vertex w with leaf ℓ we add another vertex ℓ' and connect $w\ell$ and $\ell\ell'$ to form a cycle of length three. There exists a minimum connected feedback vertex set S which contains all the vertices w that have such a leaf cycle attached. Note that the set W of vertices with attached leaf cycles is by itself a feedback vertex set so, similarly to Remark 4.5, a minimum connected feedback vertex set S is just the smallest set containing W that induces a connected subgraph.

To prove (i), note that the construction without leaf cycles is a grid graph, and it has a natural unit-disk representation. We can extend this representation at each vertex $w \in W$ by adding two small perturbations of the disk of w, making sure that neighboring vertices $w, w' \in W$ get disjoint disks. This results in a unit-disk representation of our construction.

To prove (ii), we use the same global structure, starting from our d-dimensional CONNECTED VERTEX COVER construction, but we need to introduce some further changes. We replace our triangular leaf cycles with cycles of length 4, that is, we subdivide each edge $\ell\ell'$ with a new vertex; let G^d denote the resulting graph. We use cube wiring to realize G^d as a noninduced grid graph in $\mathbb{R}^3$ and in higher dimensions. Since our original connected vertex cover construction had maximum degree 4, the graph G^d has maximum degree 5, which is only attained on vertices that have a leaf cycle attached. Let us call an induced path in this construction which has a leaf cycle on all of its internal vertices a *leaf path*. We argue that in $\mathbb{R}^3$, the graph G^3

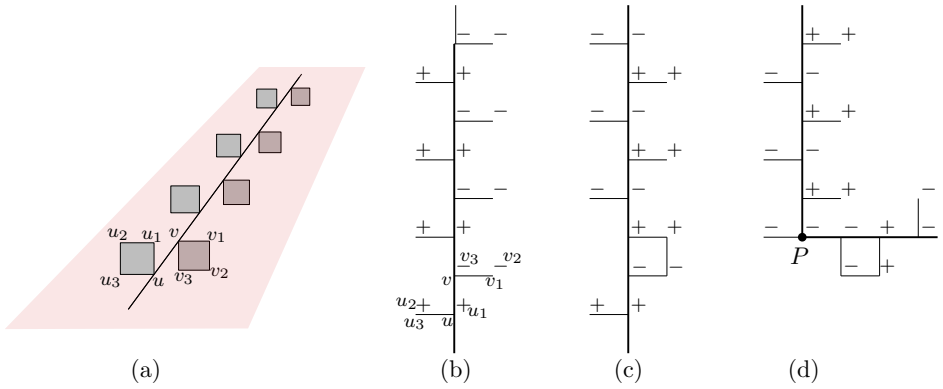

FIG. 13. (a) *and* (b): *A path with leaf cycles. The cycles on the left are above, and the cycles on the right are below, the plane* $z = 0$. (c): *A tweak along a path with leaf cycles.* (d): *Introducing a turn in a leaf path.*

can be tweaked to get an induced grid graph; the same tweaks can be used in higher dimensions as well.

We can avoid unwanted induced edges along leaf paths by putting the leaf cycles along each path on alternating sides (see Figure 13(a)). In order to visualize our gadgets in figures, we usually just draw the intersection of the gadget with a plane, which we can assume to be the plane $z = 0$. If the grid point directly above some vertex in this plane is in the gadget, e.g., we have $(x, y, 1)$ in the gadget, then we put the sign $+$ near the point $(x, y, 0)$. Similarly, the sign $-$ near $(x, y, 0)$ denotes that $(x, y, -1)$ is included in the gadget. For example, the depiction in Figure 13(a) is equivalent to Figure 13(b).

If we cannot place the leaf cycles alternately along a path for some reason, then we can introduce a tweak as shown in Figure 13(c). Note that the tweak does not change the structure or size of a minimum feedback vertex set. Furthermore, we can also introduce turns on leaf paths; we show a turn at a vertex P in Figure 13(d).

Finally, we need gadgets to deal with potential unwanted induced edges around vertices of degree 4 without leaf cycles, and around vertices of degree 5.

Due to the construction, a vertex of degree 5 has a leaf cycle and is the common endpoint of three leaf paths. We replace such vertices with the gadget on the left in Figure 14. Notice that the gadget does not contain new vertices, only some extra edges. All the marked vertices (the vertices that are in W) can be assigned to vertex-disjoint cycles. For the central vertex $P = (0, 0, 0)$ and its selected neighbors, the cycles are

$$\begin{array}{cccc} (0,0,0), & (0,0,1), & (0,-1,1,), & (0,-1,0); \\ (1,0,0), & (1,0,-1), & (1,1,-1), & (1,1,0); \\ (0,1,0), & (0,1,1), & (-1,1,1), & (-1,1,0); \\ (-1,0,0), & (-1,0,-1), & (-1,-1,-1), & (-1,-1,0). \end{array}$$

The gadget without the selected vertices is cycle-free. Therefore, there is a minimum connected feedback vertex set containing the selected vertices, i.e., the gadget has the same role as in the original construction, and the feedback vertex set itself remains unchanged.

For vertices of degree 4 without a leaf cycle we introduce the gadget on the right of Figure 14. Again, the marked vertices (the vertices from W) can be assigned to

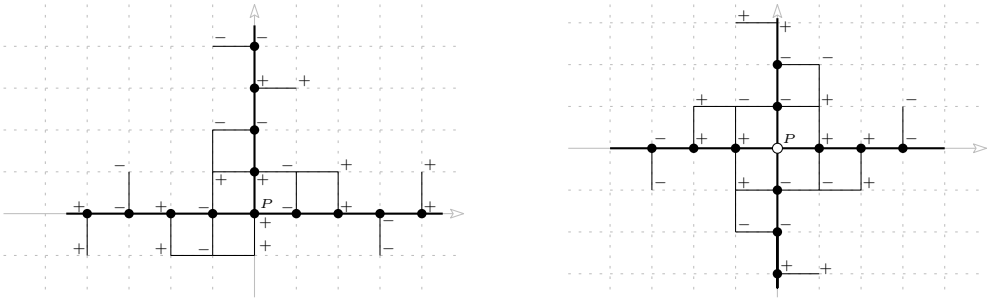

FIG. 14. *Left: Gadget for replacing degree five skeleton vertices. Right: Gadget for replacing degree five normal vertices.*

vertex disjoint cycles, and their removal gives a cycle-free graph; from the gadget, we will be left with the vertex P and four induced paths of length 6.

Using a further refinement, the above gadgetry can be integrated into the construction to create an induced d-dimensional grid graph that is at most a constant times larger than the original graph. Therefore a $2^{o(n^{1-1/d})}$ algorithm for CONNECTED FEEDBACK VERTEX SET would give a $2^{o(n^{1-1/d})}$ algorithm for FEEDBACK VERTEX SET, which would violate ETH, thus finishing the proof. ∎

4.6. Hamiltonian cycle, Hamiltonian path, and Euclidean traveling salesman problem (TSP). We prove the following theorem first.

THEOREM 4.7. *There is no $2^{o(n^{1-1/d})}$ algorithm for* HAMILTONIAN CYCLE *or* HAMILTONIAN PATH *in induced d-dimensional grid graphs, unless ETH fails.*

Proof. We can essentially use the construction by Itai, Papadimitriou, and Szwarcfiter [31] for HAMILTONIAN CYCLE in grid graphs, but in order to get a tight bound, we need to be somewhat careful with how we handle the underlying grid embedding. The writeup assumes that the reader is familiar with [31] and [44]. Our proof is a reduction from $(3,3)$-SAT that uses several steps. Similarly to the other problems, we start with the planar case.

Given a $(3,3)$-CNF formula ϕ, we start by drawing its incidence graph in the grid the following way. We place the variable vertices horizontally on the top, and the clauses vertically on the left of the figure, each edge drawn in a shape "⌟." Next, we apply the construction of Plesník [44] for the NP-completeness of DIRECTED HAMILTONIAN CYCLE in planar digraphs where the sum of in- and out-degrees of each vertex is 3 to this specific drawing; see Figure 15. (Note that the gadgetry is similar to, but slightly different from, the one given by Garey, Johnson, and Tarjan [25].) In this gadgetry, each variable and its negation is assigned a pair of parallel arcs, and the truth value is determined by the Hamiltonian cycle (the arc contained in the Hamiltonian cycle is exactly one of the two arcs). These parallel arcs are connected by XOR-gadgets, which are essentially four arcs, alternately oriented. The clauses are represented by three or two pairs of parallel arcs, depending on the number of literals inside. These arcs are attached to triple-OR and normal OR gadgets. The opposing arc of each literal is connected to an arc of the corresponding variable with a XOR gadget, that is, if variable x occurs as a positive literal in clause c, then we enforce the condition that either x is true and the negation of the literal in c is false, or x is false and the negation of the literal is true. Such XOR-gadgets can also cross each other using a crossing gadget. It is easy to see that for a $(3,3)$-CNF formula ϕ on

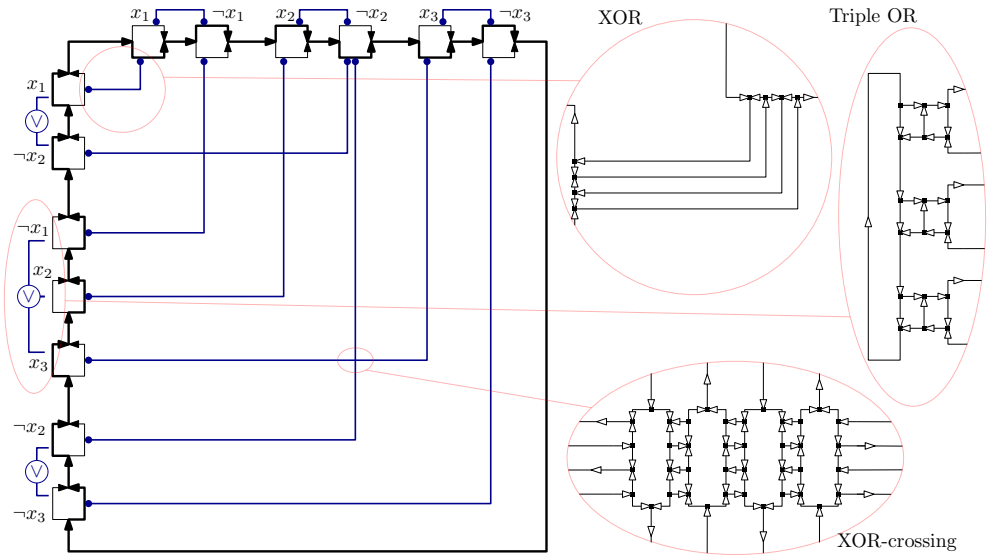

FIG. 15. *The construction by Plesník for the formula* $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$, *drawn in an* $O(n) \times O(n)$ *grid.*

n variables, the obtained planar digraph G_1 has size $O(n^2)$ and, moreover, that it is drawn in an $O(n) \times O(n)$ grid.

The next reduction step is to HAMILTONIAN CYCLE in planar undirected bipartite graphs (see also [31]); one can just replace each vertex v of G_1 with two vertices, v_{in} and v_{out}, connected by an edge, and for each arc uv of G_1, we add the edge $u_{out}v_{in}$ to the new graph G_2. Note that by introducing a 2-refinement in the drawing of G_1, we can add the new vertices and change the edges accordingly; therefore, we get a drawing of G_2 in an $O(n) \times O(n)$ grid. Using this graph, we can follow the proof by Itai, Papadimitriou, and Szwarcfiter [31] from this point onwards: we can make the above drawing of G_2 into a *parity preserving embedding* in a 3-refinement. This is a grid drawing where the left and right side of the bipartite graph are mapped to even and odd grid points, respectively. Next, they take a 9-refinement of the underlying grid, and substitute vertices with the nine vertices covered by a 2×2 grid square. Each edge becomes a *tentacle*, connecting two such grid squares. A tentacle is a width 2 grid path (see Figure 16), which allows one to model the Hamiltonian cycle passing the edge by "snaking" through it, or to do a long detour, which happens for tentacles that correspond to edges that are not in the original Hamiltonian cycle in G_2. The final graph we get is an induced grid graph that is now guaranteed to fit in an $O(n) \times O(n)$ grid. This concludes the proof of the lower bound in two dimensions.

For higher dimensions, we can reuse the variable, clause, and wire (XOR) gadgets we gained in the two-dimensional construction. Notice that the XOR-crossing gadgets are not necessary. The one additional thing to take care of is that we need to place the variable gadgets and the clause gadgets along a cycle, that is, we need to run a tentacle through these gadgets; essentially, we need to run this strip through our point sets P and Q in the cube wiring. To this end, one just needs to take a Hamiltonian path on the variable gadgets in the bottom facet, and connect its ends to the ends of the Hamiltonian path drawn on the clause gadgets in the top facet. We illustrate the approach in three dimensions in Figure 17.

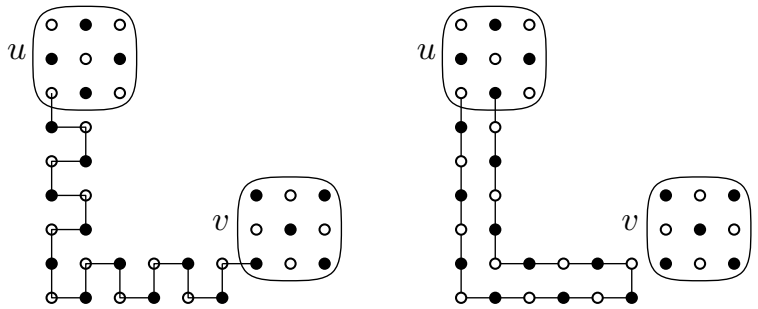

FIG. 16. *A Hamiltonian cycle passing from vertex u to v through a tentacle (left), and making a detour on the same tentacle (right).*

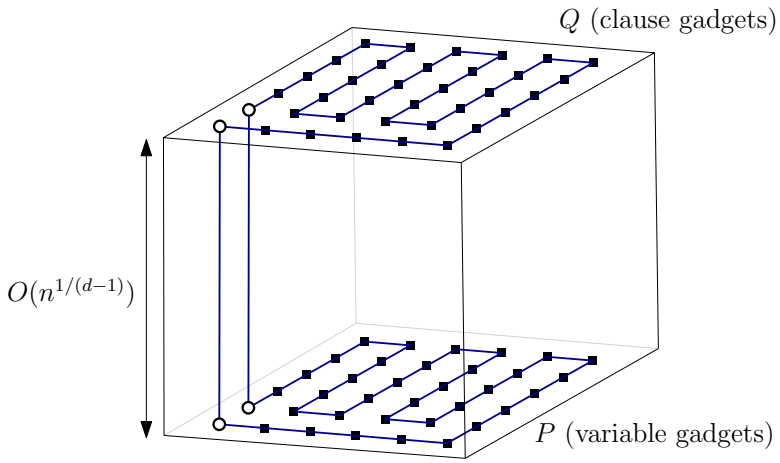

FIG. 17. *Adding a strip through all the variable and clause gadgets. This a schematic picture; the points represent gadgets, and the blue segments correspond to tentacles.*

To show the same bound for HAMILTONIAN PATH, observe that there are edges in our HAMILTONIAN CYCLE construction that are contained in all Hamiltonian cycles. Such an edge can be drawn as a simple path in the grid, instead of a tentacle as for other edges. By removing an inner vertex v of such a path, the neighbors of v will have degree 1, therefore, a Hamiltonian path is forced to have these vertices as endpoints. Consequently, we have gained an equivalent instance. ∎

The following is an important corollary.

COROLLARY 4.8. *There is no* $2^{o(n^{1-1/d})}$ *algorithm for* EUCLIDEAN TSP *in* $\mathbb{R}^d$, *unless ETH fails.*

Proof. Let G be the induced grid graph constructed above for Hamiltonian cycle, and let $P \subseteq \mathbb{R}^d$ be the set of grid points realizing its vertex set. We claim that G has a Hamiltonian cycle if and only if P has a TSP tour of length n. Indeed, a Hamiltonian cycle in G induces a tour of length n on P, since edges of the induced grid graph G correspond to pairs of vertices at distance 1. Moreover, if there is a tour of length n on P, then there must be a Hamiltonian cycle in G, since a tour of length n cannot afford to use edges of length greater than 1 (as there are no edges of length smaller than 1, and pairs of points at distance exactly one are always connected in G). ∎

5. Conclusion. We have presented an algorithmic and lower bound framework for obtaining $2^{\Theta(n^{1-1/d})}$ algorithms and matching conditional lower bounds for several problems in geometric intersection graphs. We find the following questions intriguing:

- Is it possible to obtain clique decompositions without geometric information? Alternatively, how hard is it to color the complement of a small diameter geometric intersection graph of fat objects? If we could approximate the size of a minimum clique decomposition in a greedy partition class within a constant factor, then all our algorithms (except those for Hamiltonian Cycle and Hamiltonian Path) would become robust.
- Many of our applications require the low degree property (i.e., the fact that $G_{\mathcal{P}}$ has bounded degree). Is the low degree property really essential for these applications? Would having low average degree be sufficient?
- Is it possible to modify the framework to work without the similar size assumption? In the case of Dominating Set, it is already known that the similar size assumption is necessary [6].

Finally, it would be interesting to explore the potential consequences of this framework for parameterized and approximation algorithms. Work in this direction has already begun [21].

REFERENCES

[1] J. Alber and J. Fiala, *Geometric separation and exact solutions for the parameterized independent set problem on disk graphs*, J. Algorithms, 52 (2004), pp. 134–151, https://doi.org/10.1016/j.jalgor.2003.10.001.

[2] B. Aronov, M. de Berg, E. Ezra, and M. Sharir, *Improved bounds for the union of locally fat objects in the plane*, SIAM J. Comput., 43 (2014), pp. 543–572, https://doi.org/10.1137/120891241.

[3] J. Baste and D. M. Thilikos, *Contraction-Bidimensionality of Geometric Intersection Graphs*, in 12th International Symposium on Parameterized and Exact Computation (IPEC 2017), D. Lokshtanov and N. Nishimura, eds., Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, Vol. 89, 2018, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 5:1–5:13, https://doi.org/10.4230/LIPIcs.IPEC.2017.5.

[4] M. de Berg, H. L. Bodlaender, S. Kisfaludi-Bak, and S. Kolay, *An ETH-tight exact algorithm for Euclidean TSP*, in Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, M. Thorup, ed., IEEE, Piscataway, NJ, 2018, pp. 450–461, https://doi.org/10.1109/FOCS.2018.00050.

[5] M. de Berg, H. L. Bodlaender, S. Kisfaludi-Bak, D. Marx, T. C. van der Zanden, *A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs*, in STOC 2018, ACM, New York, 2018, pp. 574–586, https://doi.org/10.1145/3188745.3188854.

[6] M. de Berg and S. Kisfaludi-Bak, *Lower bounds for dominating set in ball graphs and for weighted dominating set in unit-ball graphs*, in Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday, Lecture Notes in Comput, Sci. 12160, Springer, Cham, Switzerland, 2020, pp. 31–48, https://doi.org/10.1007/978-3-030-42071-0_5.

[7] M. de Berg, S. Kisfaludi-Bak, and G. J. Woeginger, *The complexity of dominating set in geometric intersection graphs*, Theoret. Comput. Sci., 769 (2019), pp. 18–31, https://doi.org/10.1016/j.tcs.2018.10.007.

[8] C. Biró, É. Bonnet, D. Marx, T. Miltzow, and P. Rzażewski, *Fine-grained complexity of coloring unit disks and balls*, J. Comput. Geom., 9 (2018), pp. 47–80, https://doi.org/10.20382/jocg.v9i2a4.

[9] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998), pp. 1–45, https://doi.org/10.1016/S0304-3975(97)00228-4.

[10] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof, *Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth*, Inform. and Comput., 243 (2015), pp. 86–111, https://doi.org/10.1016/j.ic.2014.12.008.

[11] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk, *A $c^k n$ 5-approximation algorithm for treewidth*, SIAM J. Comput., 45 (2016), pp. 317–378, https://doi.org/10.1137/130947374.

[12] H. L. Bodlaender and F. V. Fomin, *Tree decompositions with small cost*, Discrete Appl. Math., 145 (2005), pp. 143–154, https://doi.org/10.1016/j.dam.2004.01.008.

[13] H. L. Bodlaender and R. H. Möhring, *The pathwidth and treewidth of cographs*, SIAM J. Discrete Math., 6 (1993), pp. 181–188, https://doi.org/10.1137/0406014.

[14] H. L. Bodlaender and U. Rotics, *Computing the treewidth and the minimum fill-in with the modular decomposition*, Algorithmica, 36 (2003), pp. 375–408, https://doi.org/10.1007/s00453-003-1026-5.

[15] H. Breu and D. G. Kirkpatrick, *Unit disk graph recognition is NP-hard*, Comput. Geom., 9 (1998), pp. 3–24, https://doi.org/10.1016/S0925-7721(97)00014-X.

[16] B. N. Clark, C. J. Colbourn, and D. S. Johnson, *Unit disk graphs*, Discrete Math., 86 (1990), pp. 165–177, https://doi.org/10.1016/0012-365X(90)90358-O.

[17] B. Courcelle, J. A. Makowsky, and U. Rotics, *Linear time solvable optimization problems on graphs of bounded clique-width*, Theory Comput. Syst., 33 (2000), pp. 125–150, https://doi.org/10.1007/s002249910009.

[18] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, Cham, Switzerland, 2015.

[19] B. Escoffier, L. Gourvès, and J. Monnot, *Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs*, J. Discrete Algorithms, 8 (2010), pp. 36–49, https://doi.org/10.1016/j.jda.2009.01.005.

[20] F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, and M. 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, LIPIcs Leibniz Int. Proc. Inform. 80, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern, Germany, 2017, 65, https://doi.org/10.4230/LIPIcs.ICALP.2017.65.

[21] F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, and M. Zehavi, *ETH-tight algorithms for long path and cycle on unit disk graphs*, in 36th International Symposium on Computational Geometry, SOCG 2020, LIPIcs Leibniz Int. Proc. Inform. 164, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 44, https://doi.org/10.4230/LIPIcs.SoCG.2020.44.

[22] F. V. Fomin, D. Lokshtanov, and S. Saurabh, *Bidimensionality and geometric graphs*, in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, SIAM, Philadelphia, 2012, pp. 1563–1575.

[23] B. Fu, *Theory and application of width bounded geometric separators*, J. Comput. System Sci., 77 (2011), pp. 379 – 392, https://doi.org/10.1016/j.jcss.2010.05.003.

[24] M. R. Garey and D. S. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 826–834, https://doi.org/10.1137/0132071.

[25] M. R. Garey, D. S. Johnson, and R. E. Tarjan, *The planar Hamiltonian circuit problem is NP-complete*, SIAM J. Comput., 5 (1976), pp. 704–714, https://doi.org/10.1137/0205049.

[26] M. C. Golumbic and U. Rotics, *On the clique-width of some perfect graph classes*, Internat. J. Found. Comput. Sci, 11 (2000), pp. 423–443, https://doi.org/10.1142/S0129054100000260.

[27] S. Har-Peled and K. Quanrud, *Approximation algorithms for polynomial-expansion and low-density graphs*, SIAM J. Comput., 46 (2017), pp. 1712–1744, https://doi.org/10.1137/16M1079336.

[28] J. E. Hopcroft and R. M. Karp, *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput., 2 (), pp. 225–231, https://doi.org/10.1137/0202019.

[29] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, J. Comput. System Sci., 62 (2001), pp. 367–375, https://doi.org/10.1006/jcss.2000.1727.

[30] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, J. Comput. System Sci., 63 (2001), pp. 512–530, https://doi.org/10.1006/jcss.2001.1774.

[31] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, *Hamilton paths in grid graphs*, SIAM J. Comput., 11 (1982), pp. 676–686, https://doi.org/10.1137/0211056.

[32] H. Ito and M. 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 2010, 2010, pp. 120–127.

[33] R. J. Kang and T. Müller, *Sphere and dot product representations of graphs*, Discrete Comput. Geom., 47 (2012), pp. 548–568, https://doi.org/10.1007/s00454-012-9394-8.

[34] S. Kisfaludi-Bak, *ETH-Tight Algorithms for Geometric Network Problems*, PhD thesis, Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, Eindhoven, The Netherlands, 2019, https://research.tue.nl/en/publications/eth-tight-algorithms-for-geometric-network-problems.

[35] S. Kisfaludi-Bak and T. C. van der Zanden, *On the exact complexity of Hamiltonian cycle and q-colouring in disk graphs*, in Proceedings 10th International Conference on Algorithms and Complexity, CIAC 2017, D. Fotakis, A. Pagourtzis, and V. T. Paschos, eds., Lecture Notes in Computer Science 10236, Springer, Cham, Switzerland, 2017, pp. 369–380, https://doi.org/10.1007/978-3-319-57586-5_31.

[36] D. Kőnig, *Gráfok és mátrixok*, Mat. Fiz. Lapok, 38 (1931), pp. 116–119 (in Hungarian).

[37] D. Lichtenstein, *Planar formulae and their uses*, SIAM J. Comput., 11 (1982), pp. 329–343, https://doi.org/10.1137/0211025.

[38] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189, https://doi.org/10.1137/0136016.

[39] R. J. Lipton and R. E. Tarjan, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627, https://doi.org/10.1137/0209046.

[40] D. Marx, *The square root phenomenon in planar graphs*, in Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, 2013, Part II, Springer, Berlin, 2013, p. 28, https://doi.org/10.1007/978-3-642-39212-2_4.

[41] D. Marx and M. Pilipczuk, *Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams*, in Proceedings of the 23rd Annual European Symposium on Algorithms, ESA 2015, Lecture Notes Comput. Sci., 9294, Springer, Heidelberg, Germany, 2015, pp. 865–877, https://doi.org/10.1007/978-3-662-48350-3_72.

[42] D. Marx and A. Sidiropoulos, *The limited blessing of low dimensionality: When $1-1/d$ is the best possible exponent for d-dimensional geometric problems*, in Proceedings of the 30th Annual Symposium on Computational Geometry, SOCG 2014, ACM, New York, 2014, pp. 67–76, https://doi.org/10.1145/2582112.2582124.

[43] S. Oum, *Rank-width: Algorithmic and structural results*, Discrete Appl. Math., 231 (2017), pp. 15–24, https://doi.org/10.1016/j.dam.2016.08.006.

[44] J. Plesník, *The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two*, Inform. Process. Lett., 8 (1979), pp. 199–201, https://doi.org/10.1016/0020-0190(79)90023-1.

[45] N. Robertson and P. D. Seymour, *Graph minors.* XIII. *The disjoint paths problem*, J. Combin. Theory, Ser. B, 63 (1995), pp. 65–110, https://doi.org/10.1006/jctb.1995.1006.

[46] W. D. Smith and N. C. Wormald, *Geometric separator theorems and applications*, in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS 1998, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 232–243, https://doi.org/10.1109/SFCS.1998.743449.

[47] E. Speckenmeyer, *Untersuchungen zum Feedback-vertex-set-Problem in ungerichteten Graphen*, PhD thesis, University of Paderborn, Paderborn, Germany, 1983, http://d-nb.info/831085894.

[48] C. D. Thompson and H. T. Kung, *Sorting on a mesh-connected parallel computer*, in Proceedings of the 8th Annual ACM Symposium on Theory of Computing, STOC 1976, ACM, New York, 1976, pp. 58–64, https://doi.org/10.1145/800113.803632.

[49] C. D. Thompson and H. T. Kung, *Sorting on a mesh-connected parallel computer*, Comm. ACM, 20 (1977), pp. 263–271, https://doi.org/10.1145/359461.359481.

[50] C. A. Tovey, *A simplified NP-complete satisfiability problem*, Discrete Appl. Math., 8 (1984), pp. 85–89, https://doi.org/10.1016/0166-218X(84)90081-7.

[51] F. van den Eijkhof, H. L. Bodlaender, and M. Koster, *Safe reduction rules for weighted treewidth*, Algorithmica, 47 (2007), pp. 139–158, https://doi.org/10.1007/s00453-006-1226-x.

[52] G. J. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, Proceedings of the 5th International Workshop on Combinatorial Optimization, Lecture Notes in Comput. Sci. 2750, Springer, Berlin, 2003, pp. 185–208, https://doi.org/10.1007/3-540-36478-1_17.

[53] T. C. van der Zanden, *Theory and Practical Applications of Treewidth*, PhD thesis, Universiteit Utrecht, Department of Information and Computing Sciences, Utrecht, The Netherlands, 2019, https://dspace.library.uu.nl/handle/1874/381134.