Fine-grained complexity of coloring unit disks and balls*

Csaba Biró,[†] *Édouard Bonnet,*[‡] *Dániel Marx,*[§] *Tillmann Miltzow,*[¶] *Paweł Rzążewski,*[‖]

Abstract. On planar graphs, many classic algorithmic problems enjoy a certain "square root phenomenon" and can be solved significantly faster than what is known to be possible on general graphs: for example, Independent Set, 3-Coloring, Hamiltonian Cycle, Dominating Set can be solved in time $2^{O(\sqrt{n})}$ on an n-vertex planar graph, while no $2^{o(n)}$ algorithms exist for general graphs, assuming the Exponential Time Hypothesis (ETH). The square root in the exponent seems to be best possible for planar graphs: assuming the ETH, the running time for these problems cannot be improved to $2^{o(\sqrt{n})}$. In some cases, a similar speedup can be obtained for 2-dimensional geometric problems, for example, there are $2^{O(\sqrt{n}\log n)}$ time algorithms for Independent Set on unit disk graphs or for TSP on 2-dimensional point sets.

In this paper, we explore whether such a speedup is possible for geometric coloring problems. On the one hand, geometric objects can behave similarly to planar graphs: 3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on the intersection graph of n disks in the plane and, assuming the ETH, there is no such algorithm with running time $2^{o(\sqrt{n})}$. On the other hand, if the number ℓ of colors is part of the input, then no such speedup is possible: Coloring the intersection graph of n unit disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. More precisely, we exhibit a smooth increase of complexity as the number ℓ of colors increases: If we restrict the number of colors to $\ell = \Theta(n^{\alpha})$ for some $0 \leqslant \alpha \leqslant 1$, then the problem of coloring the intersection graph of n disks with ℓ colors

- can be solved in time $\exp\left(O(n^{\frac{1+\alpha}{2}}\log n)\right) = \exp\left(O(\sqrt{n\ell}\log n)\right)$, and
- cannot be solved in time $\exp\left(o(n^{\frac{1+\alpha}{2}})\right) = \exp\left(o(\sqrt{n\ell})\right)$, even on unit disks, unless the ETH fails.

More generally, we consider the problem of coloring d-dimensional balls in the Euclidean space and obtain analogous results showing that the problem

*Supported by the European Research Council (ERC) Starting Grant PARAMTIGHT (no. 280152) and Consolidator Grant SYSTEMATICGRAPH (no. 725978).

[†] *Department of Mathematics, University of Louisville*, csaba.biro@louisville.edu

[‡] *Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France*, edouard.bonnet@dauphine.fr

[§] *Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)*, dmarx@cs.bme.hu

[¶] *Université libre de Bruxelles (ULB), Brussels, Belgium*, t.miltzow@gmail.com

[‖] *Faculty of Mathematics and Information Science, Warsaw University of Technology*, p.rzazewski@mini.pw.edu.pl

- can be solved in time $\exp\left(O(n^{\frac{d-1+\alpha}{d}}\log n)\right) = \exp\left(O(n^{1-1/d}\ell^{1/d}\log n)\right)$, and
- cannot be solved in time $\exp\left(O(n^{\frac{d-1+\alpha}{d}-\epsilon})\right) = \exp\left(O(n^{1-1/d-\epsilon}\ell^{1/d})\right)$ for any $\epsilon > 0$, even for unit balls, unless the ETH fails.

Finally, we prove that fatness is crucial to obtain subexponential algorithms for coloring. We show that existence of an algorithm coloring an intersection graph of segments using a constant number of colors in time $2^{o(n)}$ already refutes the ETH.

1 Introduction

There are many examples of 2-dimensional geometric problems that are NP-hard, but can be solved significantly faster than the general case of the problem: for example, there are $2^{O(\sqrt{n}\log n)}$ time algorithms for TSP on 2-dimensional point sets or for Independent Set on the intersection graph of unit disks in the plane [1, 23, 30], while only $2^{O(n)}$ time algorithms are known for these problems on general metrics or on arbitrary graphs. There is evidence that these running times are essentially best possible: under the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [19], the $2^{O(\sqrt{n}\log n)}$ time algorithms for these 2-dimensional problems cannot be improved to $2^{o(\sqrt{n})}$, and the $2^{O(n)}$ algorithms for the general case cannot be improved to $2^{o(n)}$. Thus running times with a square root in the exponent seems to be the natural complexity behavior of many 2-dimensional geometric problems. There is a similar "square root phenomenon" for planar graphs, where running times of the form $2^{O(\sqrt{n})}$, $2^{O(\sqrt{k})}\cdot n^{O(1)}$, or $n^{O(\sqrt{k})}$ are known for a number of problems [4, 7–15, 17, 18, 20, 21, 23, 27, 28, 31]. More generally, for d-dimensional geometric problems, running times of the form $2^{O(n^{1-1/d})}$ or $n^{O(k^{1-1/d})}$ appear naturally, and Marx and Sidiropoulos [24] showed that, assuming the ETH, this form of running time is essentially best possible for some problems.

In this paper, we explore whether such a speedup is possible for geometric coloring problems. Deciding whether an n-vertex graph has an ℓ-coloring can be done in time $\ell^{O(n)}$ by brute force, or in time $2^{O(n)}$ using dynamic programming. On planar graphs, we can decide 3-colorability significantly faster in time $2^{O(\sqrt{n})}$, for example, by observing that planar graphs have treewidth $O(\sqrt{n})$. Let us consider now the problem of coloring the intersection graph of a set of disks in the 2-dimensional plane, that is, assigning a color to each disk such that if two disks intersect, then they receive different colors. For a constant number of colors, geometric objects can behave similarly to planar graphs: 3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on the intersection graph of n disks in the plane and, assuming the ETH, there is no such algorithm with running time $2^{o(\sqrt{n})}$. However, while every planar graph is 4-colorable, disks graphs can contain arbitrary large cliques, and hence ℓ-colorability is a meaningful question for larger, non-constant, values of ℓ as well. We show that if the number ℓ of colors is part of the input and can be up to $\Theta(n)$, then, surprisingly, no speedup is possible: Coloring the intersection graph of n disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. Thus we understand the complexity of the problem when the number of colors is a constant or $\Theta(n)$, but what exactly happens between these two extremes? Our main 2-dimensional result exhibits a smooth increase of complexity as the number ℓ of colors increases.

Theorem 1. *For any fixed $0 \leqslant \alpha \leqslant 1$, the problem of coloring the intersection graph of n disks with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O(n^{\frac{1+\alpha}{2}} \log n)} = 2^{O(\sqrt{n\ell} \log n)}$, and*
- *cannot be solved in time $2^{o(n^{\frac{1+\alpha}{2}})} = 2^{o(\sqrt{n\ell})}$, even for unit disks, unless the ETH fails.*

Let us remark that when we express the running time as a function of *two* parameters (number n of disks and number ℓ of colors) it is not obvious what we mean by claiming that a running time is "best possible." In the statement of Theorem 1, we follow Fomin et al. [16], who studied the complexity of a two-parameter clustering problem in a similar way: We restrict the parameter ℓ to be $\Theta(n^\alpha)$ for some fixed α, and determine the complexity under this restriction as a univariate function of n.

The proofs of both statements in Theorem 1 are not very specific to disks and can be easily adapted to, say, axis-parallel squares or other fat objects. However, it seems that the requirement of fatness is essential for this type of complexity behavior as, for example, the coloring of the intersection graphs of line segments does not admit any speedup compared to the $2^{O(n)}$ algorithm, even for a constant number of colors.

Theorem 2. *There is no $2^{o(n)}$ time algorithm for* 6-Coloring *the intersection graph of line segments in the plane, unless the ETH fails.*

We actually show a stronger statement that proves the lower bound even if the segments have only two directions. Let us mention that very recently Theorem 2 was strengthened by a subset of authors, who modified the argument to show that the same lower bound holds even for 4-Coloring, while for 3-Coloring there exists a subexponential algorithm [2]. We include the proof of Theorem 2 for the sake of completeness.

We observe that both hardness proofs crucially use that the segments can be of different (non-constant) lengths. For unit segments, only a weaker lower bound was shown in [2]: assuming the ETH, there is no algorithm for 4-Coloring an intersection graph of unit segments in time $2^{o(n^{2/3})}$. The exact complexity of coloring the intersection graphs of unit line segments remains open.

How does the complexity change if we look at the generalization of the coloring problem into higher dimensions? It is known for some problems that if we generalize the problem from two dimensions to d dimensions, then the square root in the exponent of the running time changes to a $1 - 1/d$ power, which makes the running time closer and closer to the running time of the brute force as d increases [24]. This may suggest that the d-dimensional generalization of Theorem 1 should have $(n\ell)^{1-1/d}$ in the exponent instead of $\sqrt{n\ell}$. Actually, this is not exactly what happens:[1] the correct exponent appears to be

[1] The astute reader can quickly realize that $2^{O((n\ell)^{1-1/d})}$ is certainly not the correct answer when, say, $\ell = \Theta(n)$ and $d = 3$: then $2^{O((n\ell)^{1-1/d})} = 2^{O(n^{4/3})}$ is worse than the running time $2^{O(n)}$ possible even for general graphs!

$n^{1-1/d}$ times $\ell^{1/d}$. That is, as d increases, the running time becomes less and less sensitive to the number of colors and approaches $2^{O(n)}$, even for constant number of colors.

Theorem 3. *For any fixed $0 \leqslant \alpha \leqslant 1$ and dimension $d \geqslant 2$, the problem of coloring the intersection graph of n balls in the d-dimensional Euclidean space with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O\left(n^{\frac{d-1+\alpha}{d}} \log n\right)} = 2^{O\left(n^{1-1/d}\ell^{1/d} \log n\right)}$, and*
- *cannot be solved in time $2^{O\left(n^{\frac{d-1+\alpha}{d}-\epsilon}\right)}$ for any $\epsilon > 0$, even for unit balls, unless the ETH fails.*

Note that the lower bound of Theorem 3 is slightly weaker than the lower bound of Theorem 1: we only prove that the exponent of n cannot be improved by any positive $\epsilon > 0$.

Techniques. The upper bounds of Theorems 1 and 3 follow fairly easily using standard techniques. Clearly, the problem of coloring disks with ℓ colors is non-trivial only if every point of the plane is contained in at most ℓ disks: otherwise the intersection graph would contain a clique of size larger than ℓ and we would immediately know that there is no ℓ-coloring. On the other hand, if every point is contained in at most ℓ of the n disks, then it is known that there is a balanced separator of size $O(\sqrt{n\ell})$ [26, 29, 30]. By finding such a separator and trying every possible coloring on the disks of the separator, we can branch into $\ell^{O(\sqrt{n\ell})}$ smaller instances (here it is convenient to generalize the problem into the list coloring problem, where certain colors are forbidden on certain disks). This recursive procedure has the running time claimed in Theorem 1. We can use higher-dimensional separation theorems and a similar approach to prove the upper bound of Theorem 3. This part is explained in detail in Section 2.

For the lower bound of Theorem 1, the first observation is that instances with the following structure seem to be the hardest: the set of disks consists of g^2 groups forming a $g \times g$-grid and each group consists of ℓ pairwise intersecting disks such that disks in group (i, j) can intersect disks only from those other groups that are adjacent to (i, j) in the $g \times g$-grid. Note that this instance has $n = g^2\ell$ disks. As a sanity check, let us observe that the $g\ell$ disks in any given row have $\ell^{g\ell}$ possible different colorings, hence we can solve the problem by a dynamic programming algorithm that sweeps the instance row by row in time in $2^{O(g\ell \log \ell)} = 2^{O(\sqrt{n\ell} \log \ell)}$, which is consistent with the upper bound of Theorem 1. We introduce the PARTIAL d-GRID COLORING problem as a slight generalization of such grid-like instances where some of the $g \times g$ groups can be missing, see Figure 1 for an illustration.

To prove that instances of this form cannot be solved significantly faster, we reduce from a restricted version of satisfiability where g^2k variables are partitioned into g^2 groups of size k each such that these groups form a $g \times g$-grid and there are two types of constraints: clauses of size at most 3 where each variable comes from the same group and equality constraints forcing two variables from two adjacent groups to be equal. It is not very difficult to show that any 3-SAT instance with $O(gk)$ variables and $O(gk)$ clauses can be embedded into such a problem, hence the ETH implies that the problem cannot be solved in

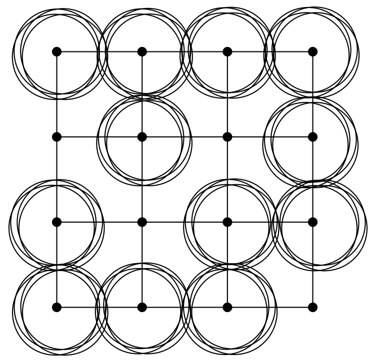


Figure 1: A grid of disks.

time $2^{o(gk)}$. We reduce such instances of 3-SAT to the coloring problem by representing each group of k variables with a group of $\ell = O(k)$ disks and make the following correspondence between truth assignments and colorings: if the i-th variable of the group is true, then we represent it by giving color $2i-1$ to the $(2i-1)$-st disk and color $2i$ to the $2i$-th disk, and we represent false by swapping these two colors. Then we implement gadgets that enforce the meaning of the clauses and the equality constraints. This way, we create an equivalent instance with $O(g^2)$ groups of $\ell = O(k)$ disks in each group, hence an algorithm with running time $2^{o(g\ell)} = 2^{o(gk)}$ would violate ETH, which is what we wanted to show.

A technical issue that arises in the reduction is that we want to consistently assign the same meaning to the colors everywhere throughout the created instance (e.g., in each group, we want to use colors $\{2i-1, 2i\}$ on the $(2i-1)$-st and $2i$-th disks of the group). This would be easy to do in the more general list coloring version of the problem where it is possible to distinguish colors using the lists. However, the colors are completely interchangeable in the ordinary coloring problem, for example, we can arbitrarily permute the colors in any proper coloring. An obvious approach is to use one of the groups as reference and define the color appearing on the i-th disk of the group to be interpreted as the i-th color. Then the challenge is to propagate this reference coloring to each and every gadget of the constructed instance. This may seem impossible at first: the "wires" propagating the reference coloring would need to cross the "wires" propagating streams of information between the gadgets. We get around this problem by splitting the reference coloring into two halves and propagating them separately.

The d-dimensional lower bound of Theorem 3 goes along the same lines, but we first prove a lower bound for a d-dimensional version of 3-SAT, where there are g^d groups of variables of size k each, arranged into a $g \times \cdots \times g$-grid. Based on earlier results by Marx and Sidiropoulos [24], we prove an almost tight lower bound for this d-dimensional 3-SAT by embedding a 3-SAT instance with roughly $g^{d-1}k$ variables and clauses into the d-dimensional $g \times \cdots \times g$-grid. Then the reduction from this problem to coloring unit balls in d-dimensional space is very similar to the 2-dimensional case.

2 Algorithms

The *ply* (also called *thickness*) of a family of sets is the maximum number of sets that cover a single point. Fix $d \geqslant 2$ and let $\mathcal{S}$ be a family of n d-dimensional convex objects. Note that the ply of $\mathcal{S}$ is at most the chromatic number of the intersection graph of $\mathcal{S}$. Indeed, the subfamily of objects covering a same point forms a clique.

The *diameter* of a geometric object S is the supremum of the distance between any pair of points of S. The *width* of S is the infimum of distances between two parallel hyperplanes H_1, H_2, such that S lies between H_1 and H_2. A family $\mathcal{S}$ of geometric objects is *B-fat* if for each $S \in \mathcal{S}$ it holds that

$$\frac{\text{diameter}(S)}{\text{width}(S)} \leqslant B.$$

The theorem below is a special case of Theorem 26 in the manuscript of Smith and Wormald [29], which was informally announced in the extended abstract presented at FOCS 1998 [30].

Theorem 4 (Smith, Wormald [29])**.** *For every $d \geqslant 1$ and $B \geqslant 0$, there exists a constant $c = c(d, B)$, such that for every B-fat collection $\mathcal{S}$ of n d-dimensional convex sets with ply at most ℓ, there exists a d-dimensional sphere Q, such that:*

(i) at most $\frac{d+1}{d+2}\,n$ elements of $\mathcal{S}$ are entirely inside Q,

(ii) at most $\frac{d+1}{d+2}\,n$ elements of $\mathcal{S}$ are entirely outside Q,

(iii) at most $cn^{1-1/d}\ell^{1/d}$ elements of $\mathcal{S}$ intersect Q.

Now, using a fairly standard divide-and-conquer approach we prove the main result of this section, which implies the upper bounds in Theorems 1 and 3.

Theorem 5. *Let G be an intersection graph of a B-fat collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of n d-dimensional convex objects. For any integer $\ell \leqslant n$ and lists $L\colon \mathcal{S} \to 2^{[\ell]}$, we can decide whether G can be properly colored with lists L in time $n^{O(n^{1-1/d}\ell^{1/d})} = 2^{O(n^{1-1/d}\ell^{1/d}\log n)}$, using polynomial space.*

Proof. First, we will exhaustively check if G contains an $(\ell+1)$-clique. If so, we can immediately terminate, as ℓ colors are clearly not sufficient to color G. We can do it in time:

$$n^{\ell+1} \cdot n^{O(1)} = n^{\ell} \cdot n^{O(1)} = 2^{\ell \log n} \cdot n^{O(1)} = 2^{O(n^{1-1/d}\ell^{1/d}\log \ell)}.$$

Indeed, $\ell \log n \leqslant n^{1-1/d}\ell^{1/d}\log \ell$ is equivalent to $\ell^{1-1/d}/\log \ell \leqslant n^{1-1/d}/\log n$; which holds when n is sufficiently large ($n \geqslant 8$) since $\ell \leqslant n$ and $x \to x^{1-1/d}/\log x$ is increasing for $x \geqslant 8$.

From now on, we can assume that there is no $(\ell+1)$-clique in G and thus the ply of $\mathcal{S}$ is at most ℓ. By Theorem 4, there exists a set $Q \subseteq \mathcal{S}$ of at most $cn^{1-1/d}\ell^{1/d}$ objects (or, equivalently, a subset of vertices of G) such that $\mathcal{S} \setminus Q$ is split into two parts $\mathcal{S}_1, \mathcal{S}_2$, each of size at most ${d+1}/{d+2}\; n$ and such that no object of $\mathcal{S}_1$ intersects an object of $\mathcal{S}_2$.

We can find such a set in an exhaustive way in time:

$$n^{cn^{1-1/d}\ell^{1/d}} \cdot n^{O(1)} = 2^{O(n^{1-1/d}\ell^{1/d}\log n)}.$$

Now, for every coloring of Q with lists L, we can try to extend this coloring to $\mathcal{S}_1$ and $\mathcal{S}_2$ recursively, using the standard divide-and-conquer approach (note that the lists of objects in $\mathcal{S} \setminus Q$ are updated according to the coloring of Q). This gives us the total running time

$$T(n) \leqslant 2^{O(n^{1-1/d}\ell^{1/d}\log \ell)} \cdot 2\, T\left(\frac{d+1}{d+2}\, n\right).$$

Solving this recursion we obtain the running time $2^{O(n^{1-1/d}\ell^{1/d}\log \ell)}$.

Thus, the total running time of the algorithm is $2^{O(n^{1-1/d}\ell^{1/d}\log n)}$. Observe that the space used is polynomial. □

Since finding a proper geometric representation of many types of intersection graphs is NP-hard [32] (or even ∃ℝ-hard [3]), we are often interested in designing *robust algorithms*. An algorithm is robust if its input is a graph (without a geometric representation), and the algorithm either gives a correct answer, or reports that the input graph is not an intersection graph.

We point out that the above coloring procedure is robust. If G is not an intersection graph of fat convex objects, then the algorithm either gives the correct answer (if G happens to have appropriate separators), or we can correctly report that the input is invalid (the exhaustive search step fails to find any separator).

Note that the running time could be slightly improved to $2^{O(n^{1-1/d}\ell^{1/d}\log \ell)}$ should we have a faster algorithm for finding separators. It is worth noting that such (polynomial) algorithms exist for d-dimensional balls, cubes, and many other shapes [26, 30]. In particular, we obtain the following result for disks in a plane.

Corollary 6. *Given a set $\mathcal{S}$ of n disks in the plane, the existence of a ℓ-coloring of an intersection graph of $\mathcal{S}$ can be decided in time $2^{O(\sqrt{n\ell}\log \ell)}$, using polynomial space.*

3 Intermediate problems

In this section, we introduce two technical problems, which will serve as an intermediate step in our hardness reductions. Let us start with some notation and definitions. For an integer n, we denote by $[n]$ the set $\{1, 2, \dots, n\}$. For a set S, we denote by 2^S the family of all subsets of S. For a fixed dimension d and $i \in [d]$, we denote by e_i the d-dimensional vector, whose i-th coordinate is equal to 1 and all remaining coordinates are equal to 0. For a point $p \in \mathbb{N}^d$ and $i \in [d]$, by $p[i]$ we denote the i-th coordinate of p, i.e. $p \cdot e_i$, where '$\cdot$' denotes the inner product of vectors. For two positive integers g, d, we denote by $R[g, d]$ the *d-dimensional g-grid*, i.e., a graph whose vertices are all vectors from $[g]^d$, and two vertices are adjacent if they differ on exactly one coordinate, and exactly by one (on that coordinate). In other words, a and a' are adjacent if $a = a' \pm e_i$ for some $i \in [d]$.

We will often refer to vertices of a grid as *cells*. Moreover, if the value of g is either clear from the context or unimportant, we will call $R[g, d]$ simply a d-dimensional grid.

The first problem is called d-GRID 3-SAT and can be seen as a 3-SAT embedded in a grid.

Problem: d-GRID 3-SAT

Input: A d-dimensional grid $G = R[g, d]$, a positive integer k, a function $\zeta : v \in V(G) \mapsto \{v_1, v_2, \ldots, v_k\}$ mapping each cell v to k fresh boolean variables, and a set $\mathcal{C}$ of constraints of two kinds:

clause constraints: for a cell v, a set $\mathcal{C}(v)$ of pairwise variable-disjoint disjunctions of at most 3 literals on $\zeta(v)$;

equality constraints: for adjacent cells v and w, a set $\mathcal{C}(v, w)$ of pairwise variable-disjoint constraints of the form $v_i = w_j$ (with $i, j \in [k]$).

Question: Is there an assignment of the variables such that all constraints are satisfied?

Not all variables need to appear in some constraint.

The second technical problem is called PARTIAL d-GRID COLORING.

Problem: PARTIAL d-GRID COLORING

Input: An induced subgraph G of the d-dimensional grid $R[g, d]$, a positive integer ℓ, and a function $\rho : v \in V(G) \mapsto \{p^v_1, p^v_2, \ldots, p^v_\ell\} \in ([\ell]^d)^\ell$ mapping each cell v to a set of ℓ points in $[\ell]^d$. It is possible that some points p^v_i and p^v_j are superimposed, i.e., they have exactly the same coordinates – they are still counted as different points.
Question: Is there an ℓ-coloring of all the points such that:

- two points in the same cell get different colors;
- if v and w are adjacent in G with $w = v + e_i$ (for some $i \in [d]$), and $p \in \rho(v)$ and $q \in \rho(w)$ receive the same color, then $p[i] \leqslant q[i]$?

The above definition is fairly technical but its underlying idea is simple. Let us first think of PARTIAL 2-GRID COLORING, the case when $d = 2$. We want to see a grid of unit disks (see Figure 1) as the centers of the disks in a discretized and normalized space (say, *inner grids*) where adjacencies between two contiguous cells (of the *outer grid*) is determined by exactly one coordinate within the inner grids. The forthcoming Section 4.1 and Figure 2 show the direct correspondence between a grid of unit disks and an instance of PARTIAL 2-GRID COLORING.

4 Two-Dimensional Lower Bounds

In this section, we discuss how to obtain a lower bound for the complexity of coloring unit disk graphs. We do it using a three-step reduction and the intermediate problems introduced in the previous section. Thanks to introducing these two intermediate steps, our construction is easy to generalize to higher dimensions (see Section 5).

We start with the last step of the reduction chain as it is the most direct. Furthermore it explains and motivates the introduction of Partial d-grid Coloring, or rather here Partial 2-grid Coloring, its special case in dimension 2. We will use the following theorem, whose proof can be found in Section 4.3.

Theorem 8. *For any $0 \leqslant \alpha \leqslant 1$, there is no $2^{o(\sqrt{n\ell})}$ algorithm solving* Partial 2-grid Coloring *on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell (that is n/ℓ cells), unless the ETH fails.*

4.1 Reduction from Partial 2-grid Coloring to ℓ-Coloring of unit disk graphs

Proof of the third and last step of the lower bound of Theorem 1. There is a transparent reduction from Partial 2-grid Coloring to ℓ-Coloring on unit disk graphs. We follow, for instance, Theorems 1 and 3 in [22]. In that paper, a reduction is given from a problem called Grid Tiling to Independent Set on unit disk graphs. The same reduction applies from Partial 2-grid Coloring, which can be seen as a coloring variant of Grid Tiling, to ℓ-Coloring on unit disk graphs.

Consider an instance of Partial 2-grid Coloring with n points in total, whose points are are in $[\ell]^2$ in each cell. One turns every point $(x, y) \in [\ell]^2$ of every cell at position (i, j) into a disk centered at $((2\ell^2 + 0.1)i + x, (2\ell^2 + 0.1)j + y)$. The common radius of all the disks is set to ℓ^2, and we set the number of colors to ℓ (see Figure 2). This way, the fact that two disks coming from adjacent cells along the x-axis (resp. y-axis) intersect is only determined by their x-coordinate (resp. y-coordinate). Indeed, the disks are big enough compared to the cells containing the points so that in the region where the disks of adjacent cells may intersect, their boundaries are close to horizontal or vertical straight lines (see the red rectangle in Figure 2). A formal explanation is detailed in Theorem 14.34 of [6]. Now Theorem 1 follows directly from Theorem 8. □

Remark 1. *Note that we do not actually require that the relation of the number n of disks and the number ℓ of colors is $\ell = \Theta(n^\alpha)$ for some α. The claim holds also for other functions $\ell = \ell(n) = O(n)$, e.g. $\ell = \Theta(\log n)$.*

Then we detail the first step of the reduction chain.

4.2 Reduction from 3-Sat to 2-grid 3-Sat

Theorem 7. *For any $0 \leqslant \alpha \leqslant 1$ there is no algorithm solving* 2-grid 3-Sat *with n variables in total and $k = \Theta(n^\alpha)$ variables per cell in time $2^{o(\sqrt{nk})} = 2^{o\left(n^{\frac{1+\alpha}{2}}\right)}$, unless the ETH fails.*

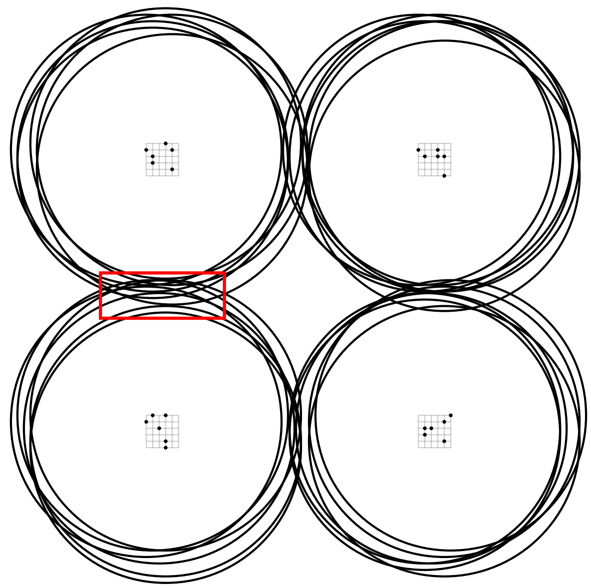


Figure 2: Illustration of how the disks are spaced out. In the region marked by the red rectangle, where disks of two adjacent cells may intersect, the boundary of each disk is close to a horizontal straight line. So, two disks do *not* intersect if and only if the y-coordinate of the center in the top cell is at most the y-coordinate of the center in the bottom cell.

Proof. The ETH together with the Sparsification Lemma [19] implies that there is no $2^{o(N+M)}$ algorithm to decide satisfiability of a 3-Sat formula with N variables and M clauses.

Let Φ be a 3-Sat formula with the variable set Var and the clause set $\mathcal{C}$. There is a simple polynomial-time procedure to modify Φ so that each variable appears at most 3 times. Indeed, first note that we can assume that no variable appears more than once within a clause. For each variable v appearing $\Delta > 3$ times, we introduce Δ new variables $v_1, v_2, \dots, v_\Delta$ and substitute each appearance of v with a different v_i. We also add clauses $(v_1 \vee \neg v_2), (v_2 \vee \neg v_3), \dots, (v_{\Delta-1} \vee \neg v_\Delta), (v_\Delta \vee \neg v_1)$ to $\mathcal{C}$. This chain of clauses enforces that all v_i's have the same truth-value in any satisfying assignment. Note that each newly introduced variable has exactly 3 occurrences. We repeat this until there are no variables with more than 3 occurrences. Thus we introduced at most $3|\mathcal{C}|$ new variables and $3|\mathcal{C}|$ new clauses. So we can assume that each variable of Φ appears at most three times, let $N := |Var|$ and $M := |\mathcal{C}|$. Clearly $M = \Theta(N)$.

We choose $k = \Theta(N^{2\alpha/(1+\alpha)})$ (actual constants will follow from the description below). Now we want to cover the set of variables by $g = 7\lceil 6M/k \rceil = \Theta(k^{(1-\alpha)/2\alpha})$ groups $\mathcal{V}_1, \dots, \mathcal{V}_g$ such that the following conditions hold:

- for each clause C, there exists a group $\mathcal{V}_i$, such that all variables of C belong to $\mathcal{V}_i$; we say this group contains the clause C;

- if two clauses C, C' share a variable, then they are contained in different groups;

- each group contains at most $k/2$ variables.

To form these groups, we first construct a partition $\mathcal{P}_0$ of the clauses into $g' = g/7 = \lceil 6M/k \rceil$ groups of size at most $\lfloor M/g' \rfloor \leqslant k/6$. As each variable occurs in at most three clauses, and thus every clause shares some variable with at most six other clauses, we can easily define a second refined partition $\mathcal{P}_1$ of the clauses into $g = 7g'$ groups, such that no two clauses that share a variable are contained in the same group. We denote these groups by $\mathcal{C}_1, \ldots, \mathcal{C}_g$. Now, we set $\mathcal{V}_i$ to be exactly the set of variables contained in the set of clauses $\mathcal{C}_i$. As each clause has at most three variables, each group $\mathcal{V}_i$ contains at most $3 \cdot k/6 = k/2$ variables.

Now, we construct an instance $I(\Phi)$ of 2-Grid 3-Sat. Let $G = R[g, 2]$ with k variables in

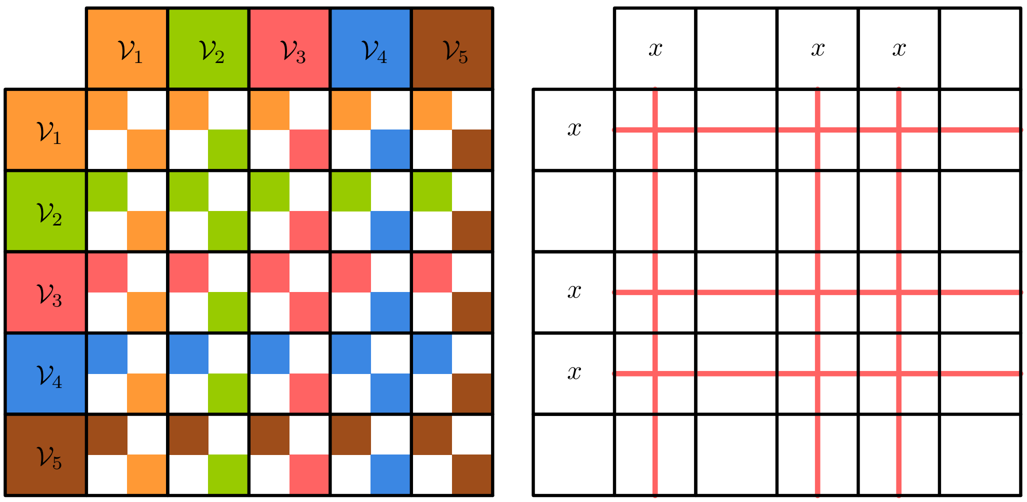


Figure 3: Allocation of the variables. Each color corresponds to a set of variables. Note that for any variable x, the set of cells containing x is connected.

each cell. Note that the number of variables in $I(\Phi)$ is $g^2 k = \Theta(k^{1/\alpha})$.

The cell (i, j) should contain the information about truth assignment of $\mathcal{V}_i \cup \mathcal{V}_j$. Note that if a variable appears in both $\mathcal{V}_i$ and $\mathcal{V}_j$, it will be represented just once in the cell (i, j). As each group $\mathcal{V}_i$ contains at most $k/2$ variables, each cell has enough space to accommodate all this information. To make all cells contain exactly k variables, we can add some dummy variables, which will not appear in any constraints. The total number of dummy variables added is at most $g^2 \cdot k$, so the total number n of variables is $\Theta(k^{1/\alpha})$. Observe that the variable group $\mathcal{V}_i$ is contained exactly in the i-th row and i-th column of G. Now, we add each clause $C \in \mathcal{C}_i$ to the set of clause constraints of the cell (i, i). Finally, we need to make sure that all copies of a single variable have the same truth assignment. Observe that for each variable x the cells containing x form a connected set (see Fig. 3). Therefore the consistency can be ensured using equality constraints. Note that since no variable appears more than once in a single cell, the equality constraints related to each edge of G are variable-disjoint.

It is easy to see that Φ is satisfiable if and only if $I(\Phi)$ is satisfiable. Furthermore $N = O(gk) = O(\sqrt{g^2k^2}) = O(\sqrt{nk})$. This implies that a $2^{o(\sqrt{nk})}$ algorithm for 2-Grid 3-Sat refutes the ETH. $\square$

4.3 Reduction from 2-grid 3-Sat to Partial 2-grid Coloring

The next step is reducing 2-GRID 3-SAT to PARTIAL 2-GRID COLORING. This step is the most important part of the proof.

Theorem 8. *For any $0 \leqslant \alpha \leqslant 1$, there is no $2^{o(\sqrt{n\ell})}$ algorithm solving* PARTIAL 2-GRID COLORING *on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell (that is n/ℓ cells), unless the ETH fails.*

Proof. We present a reduction from 2-GRID 3-SAT to PARTIAL 2-GRID COLORING. Let $I = (G, k, \zeta, \mathcal{C})$ be an instance of 2-GRID 3-SAT, where $G = R[g, 2]$ and each cell contains k variables. We think of G as embedded in the plane in a natural way, with edges being horizontal or vertical segments. We construct an equivalent instance $J = (F, \ell, \rho)$ of PARTIAL 2-GRID COLORING with $|V(F)| = \Theta(|V(G)|) = \Theta(g^2)$ and $\ell := 4k$ points per cell, where F is an induced subgraph of $R[g', 2]$ with $g' = \Theta(g)$.

First, we will explain the most basic building blocks of our construction, i.e., standard cells, reference cell, variable-assignment cells, local reference cells, and wires. Then we are ready to give an overview of the whole reduction. We finish with an elaborate explanation of more complicated gadgets and proof of their correctness.

Standard cells. A *standard cell* is a cell where the points $p_1, \ldots, p_\ell$ are on the main diagonal, that is $p_i = (i, i)$ for every $i \in [\ell]$ (see cells A and B of Figure 5a). When we talk about the ordering of the points in a standard cell, we always mean the left-to-right (or equivalently, top-to-bottom) ordering. Standard cells will be used for the basic pieces of the construction, i.e., variable-assignment cells, local reference cells, and wires (see below).

Reference coloring. Later in the construction we will choose one standard cell $\bar{R}$, which will be given a special function. We will refer to the coloring of $\bar{R}$ as the *reference coloring*. For each $i \in [\ell]$, we define the color i to be the color used for the point p_i in $\bar{R}$. Now, saying that a point somewhere else has color i, has an absolute meaning; it means using the same color as used for point p_i in $\bar{R}$.

Variable-assignment cells. For each cell $v = (i, j) \in V(G)$, we introduce in F a standard cell $A(v) = (\delta i, \delta j)$, where δ is a large constant (the coordinates of cells refer to their position in $R[g', 2]$, which is a supergraph of F). The cells $A(v)$ for $v \in V(G)$ are responsible for encoding the truth assignment of variables in $\zeta(v)$. Therefore we call them *variable-assignment cells*. We will partition variable-assignment cells into two types. The cell $A(v)$ for $v = (i, j)$ of I is called *even* if $i + j$ is even. Otherwise $A(v)$ is *odd*. Note that if v and w are adjacent cells in I, then $A(v)$ and $A(w)$ have different parity.

As each variable-assignment cell contains $\ell = 4k$ points, there are $\ell! = 2^{O(\ell \log \ell)}$ ways to color these points with ℓ colors. We will only make use of $2^{\ell/4} = 2^k$ colorings among those. In our construction, we will make sure that each variable-assignment cell receives one of the *standard colorings*. If the cell $A(v)$ is even, the coloring φ of $A(v)$ is standard if $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i - 1, 2i\}$ for $i \in [k]$ and $\varphi(p_i) = i$ for $i \in [4k] \setminus [2k]$. If the cell $A(v)$ is odd, its standard colorings φ are the ones with $\varphi(p_i) = i$ for $i \in [2k]$ and $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i - 1, 2i\}$ for $i \in [2k] \setminus [k]$. The choice of the particular standard coloring for the points in $A(v)$ defines the actual assignment of variables in $\zeta(v)$. If $A(v)$ is

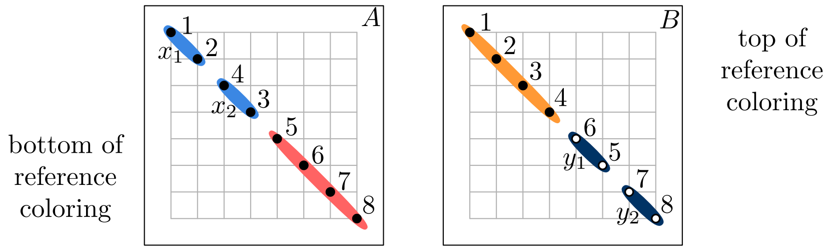


Figure 4: Variable-assignment cells of even parity contain the bottom half of the reference coloring as in cell A and cells of odd parity contain the top part of the reference coloring, as in cell B.

even, then for each $i \in [k]$, we interpret the coloring in the following way:

$$p_{2i-1} \mapsto 2i-1\ ,\ p_{2i} \mapsto 2i \text{ as setting the variable } v_i \text{ to true;}$$
$$p_{2i-1} \mapsto 2i\ ,\ p_{2i} \mapsto 2i-1 \text{ as setting the variable } v_i \text{ to false.}$$

If $A(v)$ is odd, for each $i \in [k]$, we interpret it in that way:

$$p_{2k+2i-1} \mapsto 2i-1\ ,\ p_{2k+2i} \mapsto 2i \text{ as setting the variable } v_i \text{ to true;}$$
$$p_{2k+2i-1} \mapsto 2i\ ,\ p_{2k+2i} \mapsto 2i-1 \text{ as setting the variable } v_i \text{ to false.}$$

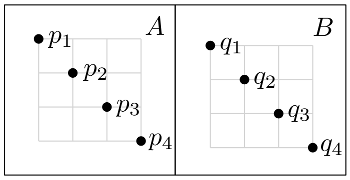
Observe that in even (odd, respectively) cells $A(v)$ the assignment of variables is only encoded by the coloring of the first (last, respectively) $2k$ points in $A(v)$. The colors of the remaining points are exactly the same as in the reference coloring, so each cell contains exactly one half of the reference coloring.

Local reference cells. For all $i, j \in [g-1]$, we introduce a new standard cell $R(i,j) := (\delta i + \delta/2, \delta j + \delta/2)$, called a *local reference cell*. Moreover, we set the reference $\bar{R}$ to be $R(1,1)$. In the construction, we will ensure that the coloring of each local reference cell is exactly the same, i.e., is exactly the reference coloring.

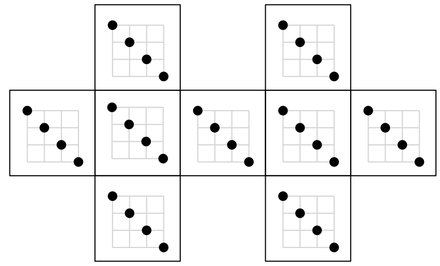
Consider the variable-assignment cell $A(v)$ for $v = (i,j)$. We say that a local reference cell $R(i',j')$ is associated with $A(v)$, if $j - j' \in \{0,1\}$ and $i - i' \in \{0,1\}$. Note that each variable-assignment cell has one, two, or four associated local reference cells. Moreover, if v, w are adjacent cells of I, then $A(v)$ and $A(w)$ share at least one associated local reference cell.

Wires. If two standard cells are adjacent, then they must be colored in the same way; thus having a path of standard cells, allows us to transport the information from one cell to another. Let us prove that claim. Let A and B be two adjacent standard cells, such that A is left of B (see Figure 5a; the argument is similar if the cells are vertically adjacent).

Let $p_1, \ldots, p_\ell$ be the points of the cell A and $q_1, \ldots, q_\ell$ be the points of the cell B. Note that the color of q_1 is necessarily equal to the color of p_1, because the x-coordinates of points $p_2, p_3, \ldots, p_\ell$ exceed the x-coordinate of q_1. Inductively, we can show that for every $i \geqslant 2$, the color of q_i is the same as the color of p_i. Indeed, the colors used for $p_{i+1}, p_{i+2}, \ldots, p_\ell$ are not available for q_i, because these points are too close to q_i. On the



(a) If two standard cells are adjacent, they must have the same coloring.



(b) Wires can be used to create many copies of the same cell.

Figure 5: Construction and usage of wires.

other hand, by the inductive assumption, all colors used on $p_1, p_2, \ldots, p_{i-1}$ are already used for points $q_1, q_2, \ldots, q_{i-1}$. Thus the only possible choice for the color of q_i is the color of p_i.

Observe that the use of wires allows us to create many copies of the same cell (see Fig. 5b). We say two cells are the same, if the point configuration and their coloring must be necessarily the same.

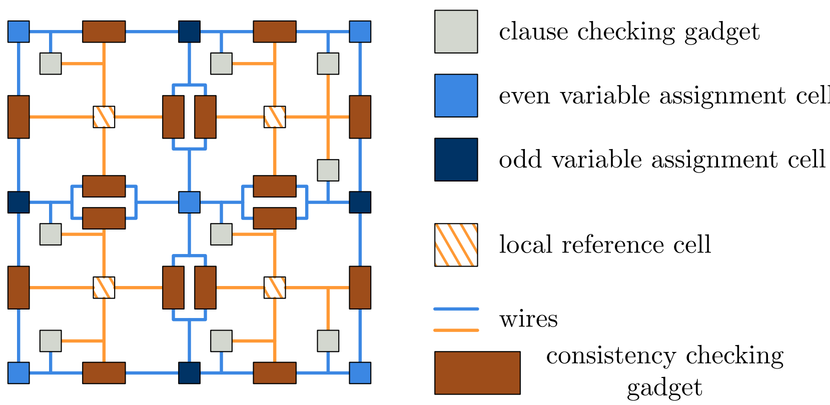


Figure 6: Illustration of the instance J. Each blue square represents a cell $A(v)$ corresponding to the cell v of I (light blue cells represent even cells and dark blue ones represent odd cells). The orange squares are local reference cells, which contain the reference coloring. Gray and brown squares represent, respectively, clause-checking and consistency gadgets.

Overview of the construction Before we move on to describe more complicated gadgets, we explain the overview of the construction. Figure 6 presents the arrangement of the cells in F. For each variable-assignment cell $A(v)$, we introduce a *clause-checking gadget*, which is responsible for ensuring that all clauses in $\mathcal{C}(v)$ are satisfied. This gadget requires an access to the reference coloring, which can be attained from the local reference cells (we can choose any of the local reference cells associated with $A(v)$). For each edge vw of G, we introduce a *consistency gadget*. In fact, for inner edges of G (i.e., the ones not incident with the outer face[2]) we introduce two consistency gadgets, one for each face incident with vw.

[2]by a face we mean a region of the plane bounded by edges of G

This gadget is responsible for ensuring the consistency on three different levels:

- to force all equality constraints $\mathcal{C}(v, w)$ to be satisfied,
- to ensure that each of $A(v)$ and $A(w)$ receives one of the standard colorings,
- to ensure that the local reference cell contains exactly the reference coloring.

This gadget also requires access to the reference coloring, so we join it with the appropriate local reference cell (see Fig. 6).

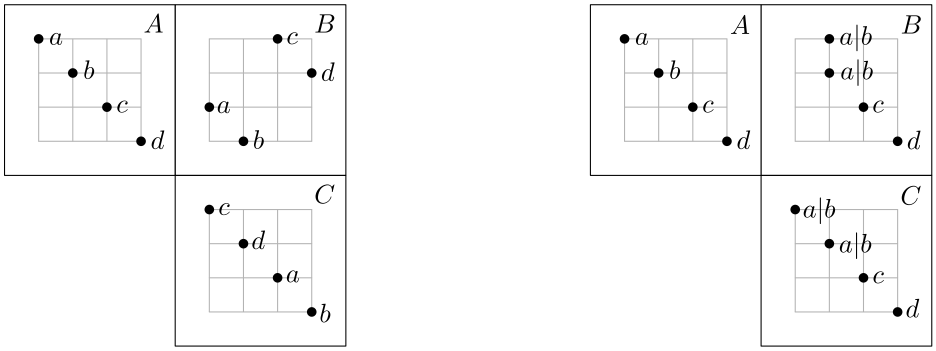
To join the variable-assignment cells and local reference cells with appropriate gadgets, we will use wires. Notice that each cell A can interact with at most four other cells, which may not be enough, if we want to attach several gadgets to A (see e.g. the middle variable assignment cell in Figure 6). However, since wires allow us to create an exact copy of A, we can attach any constant number of gadgets to copies of A, adding only a constant number of additional cells. Moreover, we can do it in a way that ensures that no two gadgets interact with each other (anywhere but on A). Thus, when we say that we attach some gadget to a cell, we will not discuss how exactly we do this.

Every gadget uses only a constant number of cells. Thus, making the constant δ large enough and using wires, we can make sure that different gadgets do not interact with each other (except for the shared cells). The total number of points in the construction is clearly increased only by a constant factor.

Permuting points and colors. Recall that when describing wires, we have not used the second coordinate of the points $p_1, \ldots, p_\ell$ and $q_1, \ldots, q_\ell$. In fact, those coordinates can be chosen at our convenience, and the argument supporting the claim in the paragraphs on the wires would still work. Combining this observation horizontally and vertically, we can force any permutation of the colors (see Figure 7a). The gadget is realized as follows. Let σ be our target permutation. To the right of a standard cell A, we put a cell B. For each $i \in [\ell]$, we add a point $(i, \sigma(i))$ to B, so there are ℓ points in total. Below the cell B, we put a standard cell C. We observe that in any feasible coloring of those three cells, for every $i \in [\ell]$, the points p_i and $q_{\sigma(i)}$ have the same color, where p_i (resp. q_i) is the point in (i, i) in the cell A (resp. cell C). Although permutation gadgets are more complicated than wires, the formal argument of correctness is identical as in the case of wires, as the propagation of colors between neighboring cells depends on only one coordinate.

Forgetting color assignment. Besides permuting points and colors, it is also possible to forget the color assignment of some points. Figure 7b shows a forgetting gadget attached to standard cells A and C. In the cell A we have the coloring from left to right a, b, c, d. In the cell C, the first two points can be colored either a, b or b, a. In particular, if A is an even variable-assignment cell, then by looking at C we cannot distinguish anymore whether the variable was set to true or to false. Thus, using a forgetting gadget attached to two standard cells, we may force equality of colors of some corresponding points, while giving some freedom of choosing the others. This concept will be used in the next paragraph.

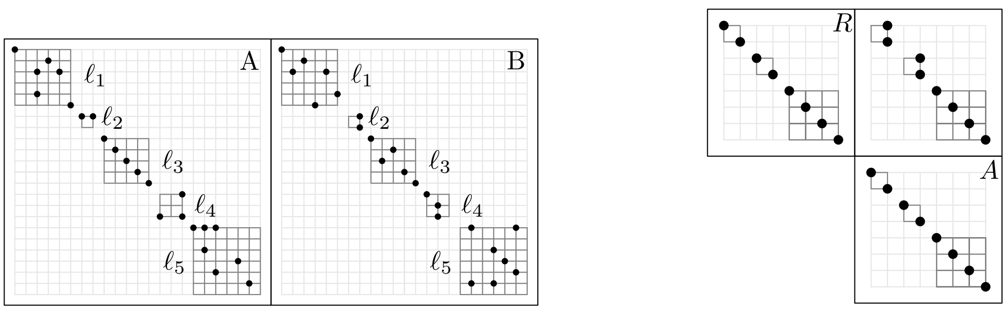
Parallelism. As we may have hinted in the previous paragraph, subparts of a given cell can act independently. In particular, this means that we can choose to forget any subset



(a) The coloring of C is the coloring of A with the permutation $\sigma = (3, 4, 1, 2)$ applied.

(b) In the cell C, colors a and b are now interchangeable.

Figure 7: Permutation gadget (left) and forgetting gadget (right), attached to cells A and C.



(a) The sets of colors used within corresponding boxes of A and B are equal.

(b) If R contains the reference coloring, then A receives one of standard colorings (for an even cell).

Figure 8: Boxes in adjacent cells with the same box-structure act independently from each other.

of information but preserve the rest. It is important to note that this is a more general phenomenon. Let $\ell_1, \ldots, \ell_t$ be positive integers summing up to ℓ. Consider an arrangement of cells where the points of each cell are all contained in the same square boxes of side lengths respectively $\ell_1, \ldots, \ell_t$, along the diagonal as shown in Figure 8a. For each $h \in [t]$, the h-th box (of side length ℓ_h) contains exactly ℓ_h points.

One may observe that a slight generalization of the argument given in the paragraph on wires shows that if A and B are adjacent cells with the same box-structure, i.e., each has points grouped in t boxes of sizes $\ell_1, \ldots, \ell_t$, then for each $h \in [t]$, the set of colors used on points in h-th box in A is exactly the same as the set of colors used in h-th box in B (see Figure 8a).

We point out that the combination of this observation and the forgetting gadget attached to a local reference cell and a variable-assignment cell A can be used to ensure that A receives one of the standard colorings (see Fig. 8b). The construction of the forgetting gadget varies depending on the parity of A. In general the gadget preserves the colors of $2k$ points containing the copy of one half of the reference coloring, and allows any permutation of colors within two-element boxes representing the variables. We will use a similar approach to check several clauses in parallel within the same group of a constant number of cells.

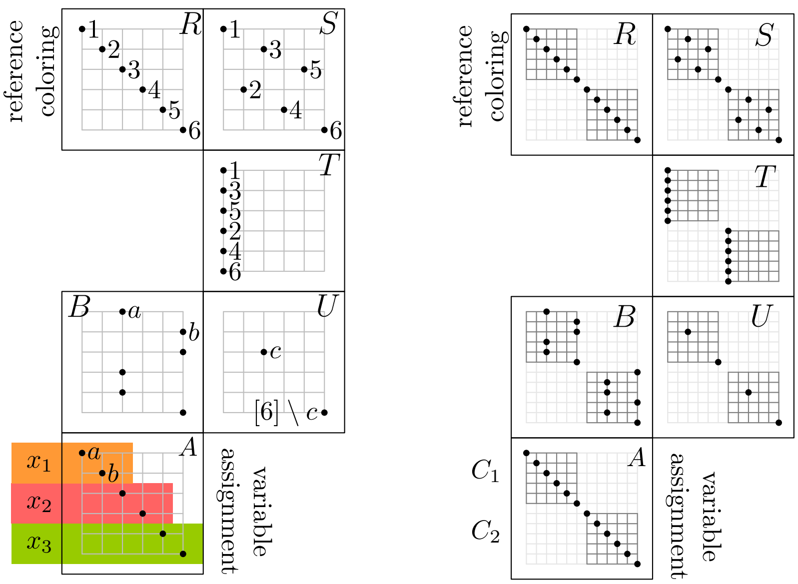


Figure 9: Illustration of the clause-checking gadget. To the left, one clause $x_1 \vee \neg x_2 \vee x_3$ is represented. To the right, two clauses are checked in parallel.

Clause gadget. We detail how a disjunction of three literals is encoded (see the left part of Figure 9). Clauses with fewer literals are just a simplification of what comes next. First, we will explain how to express a clause C, whose variables x_1, x_2, x_3 are contained in a (6×6)-box of a variable-assignment cell A. In the next paragraph we will show how to check several variable-disjoint clauses in one constant-size gadget. In general, in what follows, one should think of the coordinates that we will specify as coordinates within a box part of the cell, rather than as coordinates in the cell. The same applies to the colors, we should always look at the set of colors appearing in the particular box. Obviously, the clause-checking gadget needs to interact with variable-assignment encoding the values of x_1, x_2, x_3. For simplicity of notation assume that x_1 is encoded by coloring points p_1, p_2 with colors $1, 2$; x_2 is encoded by coloring points p_3, p_4 with colors $3, 4$ and; x_3 is encoded by coloring points p_5, p_6 with colors $5, 6$. Our clause-checking gadget needs also an access to the reference coloring contained in the cell R. This is necessary to be able to distinguish between colors e.g. 1 and 2, and thus between setting x_1 to true or to false.

First consider cells S, T, and U. The cell R contains the reference coloring and we force the order of the colors in cell T to be from top to bottom $1, 3, 5, 2, 4, 6$, similarly as we did in a permutation gadget. Consider now cell U. It has one point at position $p = (3, 3)$ and 5 points superimposed at position $(6, 6)$. Now, because of cell T, the point p can only have a color $c \in \{1, 3, 5\}$. All the other colors should be given to the 5 superimposed points.

Then, consider cells A and B.

The cell A contains the variable assignment. Recall that for each variable we use two points. If a variable occupying rows $2i-1$ and $2i$ in the cell A occurs positively in C, then we place in cell B a point in row $2i-1$ in the third column of the box and a point in the row $2i$ in the sixth column; if the variable appears negatively, we do the opposite: we place in cell B a point in the row $2i-1$ in the sixth column and a point in row $2i$ in the third column. Note that the colors of points in B are forced, looking from top to bottom they are the same as in A. By construction, the colors in the sixth column are not available to the point p. Therefore, the point p can be colored if and only if one of colors 1,3,5 appears in the third column of B, i.e., one of literals is true. Since the remaining points in U can receive any distinct colors, we conclude that the whole set of cells constituting the gadget can be colored if and only if the clause is satisfied by the truth assignment.

Checking clauses in parallel. Consider the cell v of 2-GRID 3-SAT. Let $C_1, \ldots, C_f$ be the clauses of $\mathcal{C}(v)$ and recall that these clauses are pairwise variable-disjoint. Let σ be a permutation of points in $A(v)$, such that the $2|C_1|$ points encoding the variables of C_1 appear on positions $1, 2, \ldots, 2|C_1|$, the $2|C_1|$ points encoding the variables of C_2 appear on positions $2|C_1|+1, 2|C_1|+2, \ldots, 2|C_1|+2|C_2|$ and so on. The points encoding variables which do not appear in any clause from $\mathcal{C}(v)$ and the points which do not encode any variable (i.e., the points carrying a half of the reference coloring) appear on the last position, in any order.

We introduce a new standard cell A, and using a permutation gadget we ensure that it contains the copies of points of $A(v)$ in the permutation σ. In the same way we introduce a standard cell R, which contains the reference coloring with the permutation σ applied. An illustration on how two clauses can be checked simultaneously is shown on the right part of Figure 9. Observe that since the clauses in $\mathcal{C}(v)$ are pairwise variable-disjoint, one clause-checking gadget is enough to ensure the satisfiability of all clauses in $\mathcal{C}(v)$.

Thus, for each cell $A(v)$ and its associated local reference cell R, we introduce a clause-checking gadget corresponding to the clauses in $\mathcal{C}(v)$, and join it with $A(v)$ and R.

Equality check. Let A be a cell of J and let the points p_{2i-1}, p_{2i} (p_{2j-1}, p_{2j} for $2i < 2j-1$) in the cell A encode the variable x (y, respectively). Suppose we want to make sure that always $x = y$. This is equivalent to saying that in any proper coloring φ, we have $\varphi(p_{2i-1}) + 1 = \varphi(p_{2i})$ whenever $\varphi(p_{2j-1}) + 1 = \varphi(p_{2j})$.

Such an equivalence of two variables can be expressed by two clauses $C_1 = x \vee \neg y$ and $C_2 = \neg x \vee y$. Thus, if we have an access to the reference coloring, we can ensure the equivalence using the clause-checking gadget. Observe that C_1 and C_2 are not variable-disjoint, so in fact we need to use two clause-checking gadgets. However, two clause-checking gadgets are enough to ensure the equivalence of any set of pairwise-disjoint pairs of variables represented in the single cell. Observe that A does not have to be a variable-assignment cell (i.e., does not have to carry a half of the reference coloring). In fact, we will use the equality checks for cells where each pair of points p_{2i-1}, p_{2i} corresponds to some variable, encoded in an analogous way as in variable-assignment cells.

Consistency gadget. The last gadget, called the consistency gadget, will join every three cells $A(v), A(w), R$, where $A(v)$ and $A(w)$ are variable-assignment cells corresponding to adjacent cells v and w of I, and a R is a local reference cell associated with both $A(v)$ and $A(w)$. This gadget is responsible for ensuring that colorings of these three cells are consistent, that is:

- each cell $A(v)$, $A(w)$ is colored with a standard coloring,
- the equality constraints $\mathcal{C}(v, w)$ in the 2-GRID 3-SAT instance I are satisfied,
- R has exactly the reference coloring.

For schematic picture of the gadget, refer to Figure 10. Suppose that $A(v)$ is even, $A(w)$ is odd, and v is above w in I (all other cases are symmetric). We denote the points of $A(v)$ by $p_1, p_2, \dots, p_\ell$, the points of $A(w)$ by $q_1, q_2, \dots, q_\ell$, and the points by R by $r_1, r_2, \dots, r_\ell$ (going from top-left to bottom-right). First, we introduce two forgetting gadgets and attach one of them to R and $A(v)$, and the other one to R and $A(w)$. The first gadget forgets the top half of the reference coloring, i.e., it ensures that in every coloring φ we have

- $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{\varphi(r_{2i-1}), \varphi(r_{2i})\}$ for $i \in [k]$,
- $\varphi(p_{2i-1}) = \varphi(r_{2i-1})$ and $\varphi(p_{2i}) = \varphi(r_{2i})$ for $i \in [2k] \setminus [k]$.

The second gadget forgets the bottom half of the reference coloring, i.e., it ensures that in every coloring φ we have

- $\varphi(q_{2i-1}) = \varphi(r_{2i-1})$ and $\varphi(q_{2i}) = \varphi(r_{2i})$ for $i \in [k]$,
- $\{\varphi(q_{2i-1}), \varphi(q_{2i})\} = \{\varphi(r_{2i-1}), \varphi(r_{2i})\}$ for $i \in [2k] \setminus [k]$.

We also introduce a new standard cell S. Let $s_1, s_2, \dots, s_\ell$ be the points in S. With two additional forgetting gadgets, one attached to S and $A(v)$, and the other one attached to S and $A(w)$, we ensure that in every coloring φ we have:

- $\varphi(s_{2i-1}) = \varphi(p_{2i-1})$ and $\varphi(s_{2i}) = \varphi(p_{2i})$ for $i \in [k]$,
- $\varphi(s_{2i-1}) = \varphi(q_{2i-1})$ and $\varphi(s_{2i}) = \varphi(q_{2i})$ for $i \in [2k] \setminus [k]$.

Note that the cell S contains the information about the values of all variables in $\zeta(v)$ (first $2k$ points) and in $\zeta(w)$ (second $2k$ points). Now consider the set of equality constraints $\mathcal{C}(v, w)$, recall that each of them is of the form $v_i = w_j$. Thus we want to ensure that in every coloring φ, we have $\varphi(s_{2i-1}) + 1 = \varphi(s_{2i})$ if and only if $\varphi(s_{2k+2j-1}) + 1 = \varphi(s_{2k+2j})$. We can easily do it by performing the equality check on S, using two clause gadgets and R as a reference coloring.

Is is straightforward to observe that if I is satisfiable, then J can be colored with ℓ colors, in a way described above. The opposite implication follows from the claims below.

Claim 1. The coloring of each $R(i, j)$ for $i, j \in [g - 1]$ is exactly the same as the coloring of $\bar{R} = R(1, 1)$.

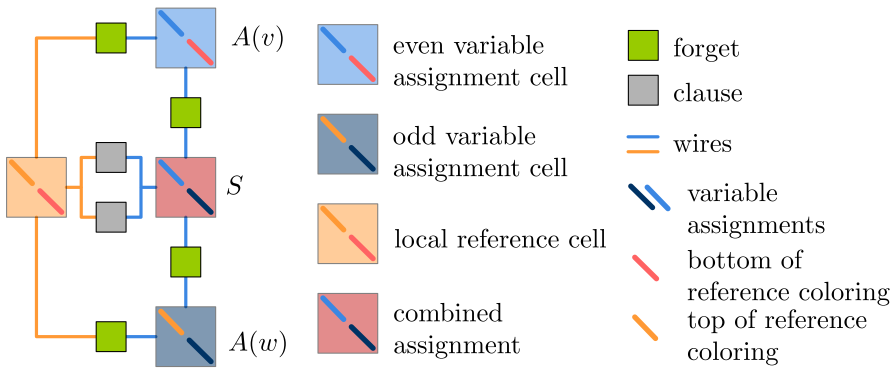


Figure 10: Overview of the consistency gadget. The clause gadgets serve to realize the equality constraints $\mathcal{C}(v, w)$.

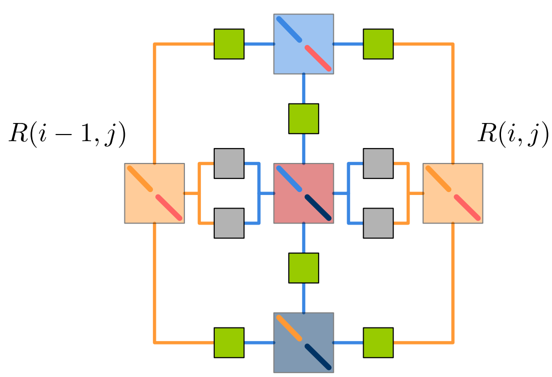


Figure 11: Two consistency gadgets ensure propagation of the reference coloring.

Proof. To show this, we will prove that the coloring of $R(i,j)$ is the same as the coloring of $R(i-1,j)$ for each $2 \leqslant i \leqslant g-1$ and $j \in [g-1]$. The case for $R(i,j-1)$ is analogous, and the claim follows inductively.

Let $v=(i,j)$ and $w=(i,j+1)$ be cells of I. Note that v and w are adjacent and $A(v)$ and $A(w)$ are associated with both $R(i-1,j)$ and $R(i,j)$. Without loss of generality assume that v is even and w is odd, see Fig. 11 for illustration.

Intuitively, the consistency gadget corresponding to the edge vw and the left face bounded by vw (we shall call it the left consistency gadget) is responsible for copying one half of the coloring of $R(i-1,j)$ to $A(v)$ and the other half to $A(w)$. Analogously, the consistency gadget corresponding to vw and the right face bounded by vw (we shall call it the right consistency gadget) is responsible for copying these halves to $R(i,j)$, making sure that the coloring of $R(i,j)$ is exactly the same as the coloring of $R(i-1,j)$. More formally, for $f \in [\ell]$, by p_f, q_f, r_f, and r'_f we denote, respectively, the points of $A(v)$, $A(w)$, $R(i-1,j)$, and $R(i,j)$. By the correctness of the left consistency gadget, we know that for every coloring φ, we have:

- $\varphi(r_f) = \varphi(q_f)$ for all $f \in [2k]$,
- $\varphi(r_f) = \varphi(p_f)$ for all $f \in [4k] \setminus [2k]$.

Analogously, by the correctness of the right consistency gadget, we know that for every coloring φ, we have:

- $\varphi(q_f) = \varphi(r'_f)$ for all $f \in [2k]$,
- $\varphi(p_f) = \varphi(r'_f)$ for all $f \in [4k] \setminus [2k]$.

This shows that $\varphi(r_f) = \varphi(r'_f)$ for every coloring φ and every $f \in [\ell]$, which proves the claim. □

Claim 2. The following statements are true.

1. The coloring of each $A(v)$ is one of the standard colorings.
2. For each pair of adjacent cells v,w of I, all local constraints $\mathcal{C}(v,w)$ are satisfied.
3. For each cell v of I, all constraints $\mathcal{C}(v)$ are satisfied.

The claim follows directly from Claim 1 and the correctness of forget, clause-checking, and consistency gadgets.

Now, observe that the total number of points in F is $n = O(g^2\ell) = O(n')$, where $n' = g^2 k$ is the total number of variables in I. Thus, the existence of an algorithm solving J in time $2^{o(\sqrt{n\ell})}$ could be used to solve I in time $2^{o(\sqrt{n'k})}$, which, by Theorem 7, contradicts the ETH. □

5 Higher Dimensional Lower Bounds

Recall that in the hardness proof of 2-grid 3-Sat and Partial 2-grid Coloring (see Theorems 7 and 8) we started with a 3-Sat instance with N variables and $\Theta(N)$ clauses, we formed $g = \Theta(N/k)$ groups, each containing $O(k)$ variables, and we arranged them on in such a way that every pair of groups met in a separate grid cell. This required $O(g^2)$ grid cells.

Suppose we want to try a similiar approach to obtain a tight (i.e., matching the upper bound in Theorem 5 lower bound in $d \geqslant 3$ dimensions. We observe that the naive approach of creating $\Theta(N/k)$ groups is not enough. Indeed, a standard computation shows that the bound in Theorem 5 is attained for the grid $R[g, d]$ with $g = \Theta((N/k)^{1/(d-1)})$ and k variables/points per cell. Thus we have to refine our reduction from 3-Sat to d-grid 3-Sat.

5.1 Embeddings

For integers $g, d \geqslant 1$, we denote by $H[g, d]$ the d-dimensional *Hamming grid*, i.e., a graph whose vertices are all points from $[g]^d$, and two vertices are adjacent if their Hamming distance is exactly one (in other words, they differ on exactly one coordinate).

An *embedding* of a graph F into a graph G is a mapping $f: V(F) \to 2^{V(G)}$, such that:

- for each $v \in V(F)$, the set $f(v)$ is connected in G,
- for each edge uv of G, the sets $f(u)$ and $f(v)$ *touch*, i.e., either they have a non-empty intersection or there is an edge in G joining a vertex from $f(u)$ to a vertex of $f(v)$.

The *depth* of an embedding f is the maximum number of vertices of F mapped to sets containing the same vertex of G, that is $max\{|S| : S \subseteq V(F) \text{ and } \bigcap_{v \in S} f(v) \neq \emptyset\}$.

Observe that if f is an embedding of G into F with depth D, and f' is an embedding of F into H with depth D', then the composition $f' \circ f$ of f and f' is an embedding of F into H with depth $D \cdot D'$.

Now we will present a series of results about graph embeddings. We start with embedding arbitrary graphs into Hamming grids.

Theorem 9 (Marx, Sidiropoulos [25])**.** *Let $d \geqslant 2$. For every graph G with m edges, no isolated vertices, and with maximum degree Δ, there is an embedding f from G to $H[g, d-1]$ having depth $O(d^2\Delta)$, where $g = O(m^{1/(d-1)} \cdot \frac{\log m}{\log \log m})$. Moreover, such an embedding can be found in deterministic polynomial time.*

The next step will be embedding a Hamming grid into another, smaller Hamming grid.

Lemma 10. *For every $g, d \geqslant 1$ and every $k = O(g^d)$, there exists $g' = O(g/k^{1/d})$ and an embedding f of $H[g, d]$ into $H[g', d]$ with depth $O(k)$. Moreover, this embedding can be found in deterministic polynomial time.*

Proof. Let $s = \lfloor k^{1/d} \rfloor$ and $g' = \lceil g/s \rceil = O(g/k^{1/d})$. Let $v = (a_1, a_2, \ldots, a_d)$ be a vertex of $H[g,d]$. We define f by mapping v to the singleton containing $(1+\lfloor a_1/s \rfloor, 1+\lfloor a_2/s \rfloor, \ldots, 1+\lfloor a_d/s \rfloor)$. Note that the number of vertices of $H[g,d]$ mapped to a single vertex is $s^d = O(k)$ It is straightforward to verify that f is an embedding. □

Finally, Hamming grids can be embedded in grids.

Theorem 11 (Marx, Sidiropoulos [25])**.** *For every $d, g \geqslant 1$ there is an embedding f from $H[g, d-1]$ to $R[g,d]$ having depth at most d. Moreover, such an embedding can be found in deterministic polynomial time.*

Now, by combining the above results, we show how we can efficiently embed an incidence graph of a 3-Sat formula into the grid.

Lemma 12. *Let Φ be a 3-Sat formula over the variable set $Var = \{x_1, x_2 \ldots, x_N\}$, such that each variable appears in at most $\Delta \geqslant 3$ clauses, and let $k = O(N)$ be an integer. There exists $g = O(\Delta(\Delta N/k)^{1/(d-1)} \cdot \frac{\log(\Delta N)}{\log\log(\Delta N)})$ and a mapping φ of variables of Φ to subsets of vertices of $R[g,d]$, such that:*

(i) for every $x \in Var$, the set $\varphi(x)$ is connected,

(ii) for every $v \in V(R[g,d])$, the number of variables x such that $v \in \varphi(x)$ is $O(\Delta d^3 k)$,

(iii) for every clause C, there exists a vertex $v(C) \in V(R[g,d])$ such that $v(C) \in \bigcap_{x \in C} \varphi(x)$ (if there is more than one such vertex, we set $v(C)$ to be any of them);

(iv) if for two clauses C, C' it holds that $v(C) = v(C')$, then C and C' are variable-disjoint.

Moreover, such a mapping can be found in polynomial time.

Proof. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ be the set of clauses of Φ. Consider an incidence graph G of Φ, i.e., the bipartite graph with the vertex set $Var \cup \mathcal{C}$, and the edge set $\{xC \colon x \in Var, C \in \mathcal{C}$, and $x \in C\}$. Note that the maximum degree of G is Δ.

By Theorem 9, we can find an embedding f from G to $H[g', d-1]$ for $g' = O((\Delta N)^{1/(d-1)} \cdot \frac{\log(\Delta N)}{\log\log(\Delta N)})$, with depth $O(d^2\Delta)$. Now, by Lemma 10, there exists an embedding f' of $H[g', d-1]$ into $H[g'', d-1]$ for $g'' = O(g'/k^{1/(d-1)}) = O((\Delta N/k)^{1/(d-1)} \cdot \frac{\log(\Delta N)}{\log\log(\Delta N)})$ with depth $O(k)$. By Theorem 11, there is an embedding f'' of $H[g'', d-1]$ into $R[g'', d]$ with depth at most d.

Let $b = 3\Delta + 1$. Next, consider the following depth-1 embedding f''' of $R[g'', d]$ into $R[g,d]$, where $g = bg'' = O(\Delta(\Delta N/k)^{1/(d-1)} \cdot \frac{\log(\Delta N)}{\log\log(\Delta N)})$. For $a = (a_1, a_2, \ldots, a_d) \in V(G[g'', d])$, we define $f'''(a) = \bigcup_{q=1}^{d} \bigcup_{p=0}^{b-1} \{(ba_1, ba_2, \ldots, ba_d) + pe_q\}$.

The composition φ' of f, f', f'', and f''' is an embedding of G into $R[g,d]$ with depth $O(d^3 \Delta k)$.

By the properties of f''', we observe that for every clause C the set $\varphi'(C)$ contains a vertex of the form ba, where $a = (a_1, a_2, \ldots, a_d)$ for integers $a_1, a_2, \ldots, a_d$. We set $v'(C)$ to be such a vertex (if there is more than one, we pick an arbitrary one).

Consider a clause C and let $v'(C) = ba$ for $a = (a_1, a_2, \ldots, a_d)$. Let $\mathcal{C}'$ be the set of clauses C', such that $v'(C') = ba$. We want to partition $\mathcal{C}'$ into at most $b - 1 = 3\Delta$ groups $\mathcal{C}'_1, \mathcal{C}'_2, \ldots, \mathcal{C}'_{b-1}$, such that the clauses in one group are variable-disjoint. We can easily do it with a greedy algorithm – note that each clause may share a variable with at most $3(\Delta - 1) < 3\Delta$ other clauses. Now, for every clause C of $\mathcal{C}'_i$, for $i \in [b-1]$, we will extend φ', by including $v(C) := ba + ie_1 + e_2$ in $\varphi'(C)$. It is not hard to verify that φ' is still an embedding of G into $R[g, d]$ with depth $O(d^3\Delta k)$.

Finally, for every $x \in Var$, we define a mapping φ of variables of Φ to subsets of vertices of $R[g, d]$ in the following way: $\varphi(x) = \varphi'(x) \cup \bigcup_{C:\, x \in C} \varphi'(C)$. Note that each $\varphi(x)$ is connected, since $\varphi'(x)$ and every $\varphi'(C)$ is connected, and $\varphi'(x)$ and $\varphi'(C)$ touch whenever $x \in C$.

Moreover, recall that the depth of φ' is $O(d^3\Delta k)$. Since the number of variables mapped by φ to any fixed vertex of $R[g'', d]$ is at most three times larger, so it is $O(d^3\Delta k)$. The last two properties follow from the observation that $v(C)$ belongs to $\varphi(x)$ for every $x \in C$. □

Now we are ready to prove the following theorem, which is a generalization of Theorem 7 to higher dimensions.

Theorem 13. *For any integer $d \geqslant 3$ and reals $\epsilon > 0$ and $0 \leqslant \alpha \leqslant 1$, there is no algorithm solving d-GRID 3-SAT with n variables in total and $k = \Theta(n^{\alpha})$ variables per cell in time $2^{O\left(n^{\frac{d-1+\alpha}{d} - \epsilon}\right)} = 2^{O(n^{1-1/d-\epsilon}k^{1/d})}$, unless the ETH fails.*

Proof. Let Φ be a 3-SAT instance with N variables and $\Theta(N)$ clauses, and let each variable appear in at most $\Delta = 3$ clauses. By the ETH and the Sparsification Lemma [19], there is no algorithm deciding the satisfability of Φ in time $2^{o(N)}$.

Let $k = \Theta\left(\left(\frac{N^{1/(d-1)}\log N}{\log\log N}\right)^{d/(1/(d-1)+1/\alpha)}\right)$, $g = O((N/k)^{1/(d-1)} \cdot \frac{\log N}{\log\log N})$, and let φ be the mapping of variables of Φ to the subsets of vertices of $R[g, d]$ given by Lemma 12 (recall that $\Delta = 3$).

Now we construct an instance $I(\Phi)$ just as we did in the proof of Theorem 7. For every cell v of $R[g, d]$ we add variables $\varphi^{-1}(v)$. For each clause C, we add the clause constraint to the cell $v(C)$. Moreover, using equality constraints, we ensure that all copies of the same variable get the same truth assignment (recall that each set $\varphi(x)$ is connected). It is clear that $I(\Phi)$ is a satisfiable instance of d-GRID 3-SAT if and only if Φ is satisfiable.

The number of cells in $I(\Phi)$ is $g^d = O\left((N/k)^{d/(d-1)}\left(\frac{\log N}{\log\log N}\right)^d\right)$. The total number of variables is thus

$$n = g^d k = O\left((N/k)^{d/(d-1)} k \cdot \left(\frac{\log N}{\log\log N}\right)^d\right) = O\left((N^d/k)^{1/(d-1)} \cdot \left(\frac{\log N}{\log\log N}\right)^d\right),$$

and a direct computation shows that $k = \Theta(n^\alpha)$.

Suppose we have an algorithm solving d-Grid 3-Sat in time $2^{O\left(n^{\frac{d-1+\alpha}{d}-\epsilon}\right)}$ for some $\epsilon > 0$. Applying it to $I(\Phi)$ gives a total running time $\exp\left(O(n^{\frac{d-1+\alpha}{d}-\epsilon})\right) = 2^{o(N)}$. So we can use this algorithm to solve Φ in time $2^{o(N)}$, thus refuting the ETH. □

5.2 Reduction from d-grid 3-Sat to Partial d-grid Coloring

After establishing the hardness of d-Grid 3-Sat, we can proceed to showing the hardness of Partial d-grid Coloring.

Theorem 14. *For any integer $d \geqslant 3$, and reals $0 \leqslant \alpha \leqslant 1$ and $\epsilon > 0$, there is no $2^{O\left(n^{1-1/d-\epsilon}\ell^{1/d}\right)}$ algorithm solving* Partial d-grid Coloring *on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell, unless the ETH fails.*

The proof of Theorem 14 is a consequence of Theorem 13 and of the gadgets constructed in Section 4. The reduction is now from d-Grid 3-Sat and we only have to very slightly adapt the construction. The overall picture is the grid-like structure of Figure 6 extended to dimension d. From an instance I of d-Grid 3-Sat produced by the reduction of Theorem 13 on the grid $R[g, d]$ with k variables per cell, we build an equivalent instance J of Partial d-grid Coloring on a subgraph of $R[g', d]$ with $g' = \Theta(g)$ with $\ell := 4k$ points (and colors), in the following way. A cell of I is again called *even* (resp. *odd*) if its coordinates sum up to an even integer (resp. odd integer). For each even/odd cell of I, we have a corresponding even/odd *standard* cell in J (in the grid $R[g', d]$). We define similarly a *standard cell* as a cell in which the ℓ points $p_1, \ldots, p_\ell$ are in the main diagonal, i.e., $p_i = (i, i, \ldots, i) \in [\ell]^d$ for all $i \in [\ell]$. Observe that, as in the 2-dimensional case, two adjacent standard cells have to be colored in the same way (see Figure 12).

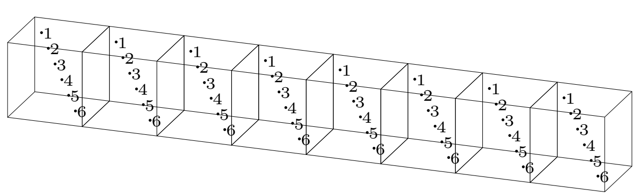


Figure 12: A wire in dimension 3. The coloring of any cell forces the same coloring in the others.

In each even/odd cell of J, the truth assignment of the k variables is encoded the same way as in the 2-dimensional case. Each of those cells are wired via a constant number (four is enough) of adjacent standard cells to one clause gadget (responsible for ensuring the satisfiability of the clauses on those very variables). One can notice that, planarity not

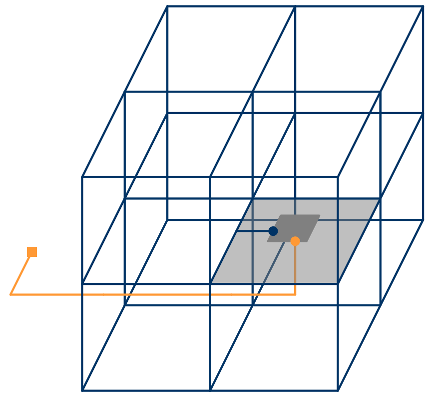


Figure 13: Wiring the reference coloring (depicted in blue), to the place where it is needed.

being an obstacle anymore, bringing the reference coloring to the clause gadgets is no longer a delicate issue. We can therefore simplify a bit our reduction for $d \geqslant 3$.

First, we do not require the local reference coloring cells of the planar case. Now, we only have one global reference coloring cell, and we wire this cell to every clause gadget. We can do it, as all gadgets are are embedded in a two-dimensional subspace, so we can always use extra dimensions to find place for wires (see Fig. 13).

Secondly, we no longer need two consistency gadgets between two adjacent even and odd cells. Now, between every even cell (resp. every odd cell) and each of its $2d$ neighboring odd cells (resp. even cells), we have one consistency gadget. Each even/odd cell is attached to $2d$ wires as represented in Figure 14 in dimension 3.

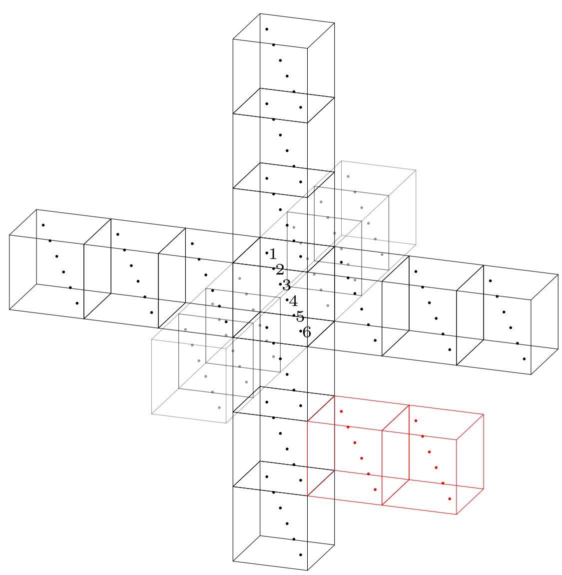


Figure 14: The $2d$ wires leaving a, say, even cell to reach each one of the $2d$ consistency gadgets shared with the $2d$ adjacent odd cells. The cells in red are part of the wires to the clause gadget.

Every consistency and clause gadget is embedded into a plane (subset of cells in an affine subspace of dimension 2) supported by say, e_1 and e_2, the first two vectors of the canonical basis. The wires which should be plugged to the corresponding gadget are naturally guided towards the plane (see Figure 15 where we give the example of the clause gadget). This construction can be realized with $g' = 100g$. The soundness follows from the 2-dimensional case.

It is noteworthy that we are not using the extra dimensions for those crucial gadgets. The higher dimensional space is mainly needed and used in Section 5.1 to get Theorem 13.

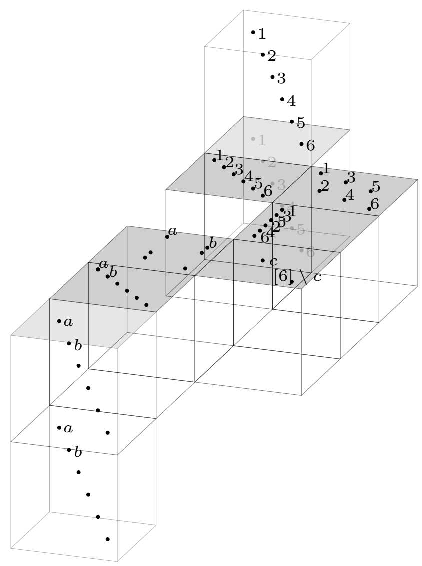


Figure 15: The clause gadget in dimension 3. The wires meet in a plane where the information is projected to 2 dimensions. The core of the gadget is then identical to the 2-dimensional case.

The final step in proving the lower bound in Theorem 3 is reducing Partial d-grid Coloring to ℓ-Coloring of an intersection graph of d-dimensional unit balls. It is very similar to the one in Theorem 1 (see also [24, Theorem 3.1.]).

5.3 Reduction from Partial d-grid Coloring to ℓ-Coloring of unit d-ball graphs

Proof of the third and last step of the lower bound of Theorem 3. There is a transparent reduction from Partial d-grid Coloring to ℓ-Coloring on intersection graphs of d-dimensional balls. Recall that the points of an instance of Partial d-grid Coloring are in $[\ell]^d$ in each cell, and that the cells created by the reduction of Theorem 14 are in $[g']^d$ with $g' = \Theta(g)$.

One turns every point $(x_1, \ldots, x_d) \in [\ell]^d$ of every cell at position $(i_1, \ldots, i_d) \in [g']^d$ into a d-dimensional ball centered at $((2(d-1)\ell^2 + 0.1)i_1 + x_1, \ldots, (2(d-1)\ell^2 + 0.1)i_d + x_d)$. The common radius of all the balls is set to $(d-1)\ell^2$, and we set the number of colors to ℓ. The correctness of this reduction is similar to the 2-dimensional case and is detailed in [25, Theorem 3.1.]. □

6 Segments

First we will present the hardness proof for the list coloring problem, and then we will show how to modify it to obtain the result for 6-coloring. The segments in our construction will be axis-parallel, the class of intersection graphs of such segments in denoted by 2-Dir. In the description, we will identify the vertices of the intersection graph with the segments.

Theorem 15. *There is no algorithm working in time* $2^{o(n)}$ *for the list 6-coloring of* 2-Dir *graphs with* n *vertices, unless the ETH fails.*

Proof. We reduce from 3-coloring of graphs with maximum degree at most 4. Let G be a graph with n vertices and $m = \Theta(n)$ edges. It is a folklore result that, assuming the ETH, there is no algorithm solving this problem in time $2^{o(n)}$ (see for instance Lemma 1 in [5]).

Let the vertex set of G be $V = \{v_1, v_2 \ldots, v_n\}$. We construct a 2-Dir graph G' with lists L of colors from the set $\{1, 2, 3, 4, 5, 6\}$, such that G is 3-colorable if and only if G' is list-colorable with respect to the lists L.

For each vertex v_i we introduce two segments: a horizontal one, called x_i, and a vertical one, called y_i, so that they form a half of a $n \times n$ grid (see Figure 16). When i increases, x_i becomes longer and y_i shorter. One may observe that the intersection graph induced by those segments is *not* grid-like. We set the lists of each x_i to $\{1, 2, 3\}$ and the lists of each y_i to $\{4, 5, 6\}$.

Each color $c \in \{1, 2, 3\}$ will be identified with the color $c + 3$. Thus, we want to ensure that in any feasible 6-coloring f of G' we have:

1. $f(x_i) + 3 = f(y_i)$ for all $i \in [n]$,
2. $f(x_i) + 3 \neq f(y_i)$ for all $i > j$ such that $v_i v_j$ is an edge of G.

This is achieved by using *equality gadgets* and *inequality gadgets*. At the crossing point of x_i and y_i, we put an equality gadget (represented by a circle on Figure 16). Moreover, for each edge $v_i v_j$ of G, we put an inequality gadget at the crossing point of x_i and y_j, $i > j$ (represented by a square on Figure 16).

The equality gadget consists of 9 segments, arranged as depicted on Figure 17. Consider the equality gadget and suppose x_i gets the color 1. Then a_1 receives color 4, and b_1 and c_1 colors 5 and 6, respectively. Thus the only choice for the color for y_i is 4. This can be extended to remaining segments of the gadget e.g. by coloring a_2 with color 2, a_3 with color 3, b_2, c_2 with color 5, and b_3, c_3 with color 6. The other cases are symmetric.

The inequality gadget consists of 7 segments, arranged as depicted on Figure 18. So now consider an inequality gadget and suppose the color of x_i is 1. Then p_1 and p_2 get colors 5 and 6, respectively. Thus the only choice for x' is 4, which prevent y_j from receiving color 4. This coloring can be extended to remaining segments by coloring q_1, q_2 with color 2 and r_1, r_2 with color 3. The other cases are again symmetric.

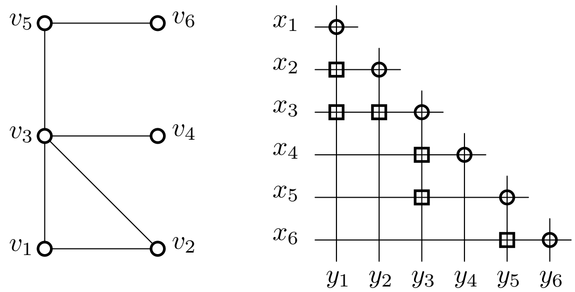
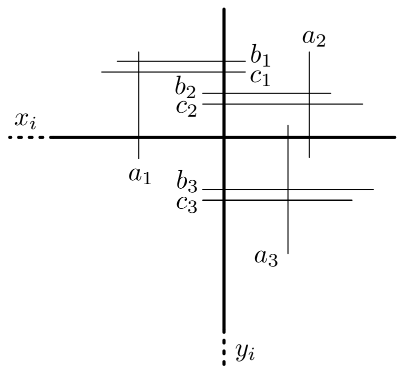


Figure 16: A graph G (left) and a high-level construction of G' (right). Circles denote equality gadgets and squares denote inequality gadget

This proves that G' has a coloring with lists L if and only if G is 3-colorable.



vertex	list
x_i	$1, 2, 3$
y_i	$4, 5, 6$
a_1	$1, 4$
b_1	$4, 5$
c_1	$4, 6$

vertex	list
a_2	$2, 5$
b_2	$4, 5$
c_2	$5, 6$
a_3	$3, 6$
b_3	$4, 6$
c_3	$5, 6$

Figure 17: Equality gadget.

The number of vertices of G' is $n' = \underbrace{2n}_{x_i, y_i} + \underbrace{9n}_{\text{equality}} + \underbrace{7m}_{\text{inequality}} = \Theta(n)$.

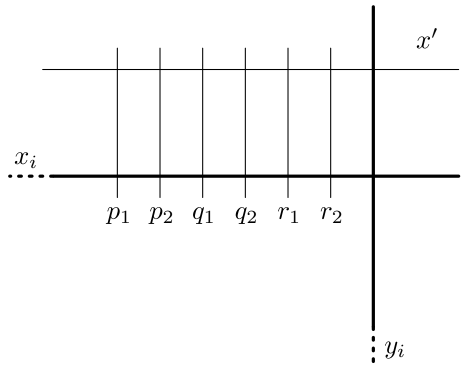
Now suppose we can find a list coloring of G' in time $2^{o(n')}$. This yields an algorithm for 3-coloring of G in time $2^{o(n')} = 2^{o(n)}$, which in turn contradicts the ETH. $\square$

To obtain Theorem 2, we modify the construction above to simulate the lists of available colors.

Theorem 2. *There is no $2^{o(n)}$ time algorithm for* 6-Coloring *the intersection graph of line segments in the plane, unless the ETH fails.*

Proof. We modify the construction from the proof of Theorem 15. We first introduce six overlapping segments $R_1, R_2, \ldots, R_6$, whose coloring will serve as a reference coloring. Since these segments are pairwise intersecting, each of them receives a different color. We will denote by $i \in \{1, 2, \ldots, 6\}$ the color assigned to R_i.

Now, for each segment v of G', we want to simulate the list $L(v)$ from the instance of list 6-coloring constructed in the proof of Theorem 15. For every color $i \notin L(v)$, we want



vertex	list
x_i	$1,2,3$
y_i	$4,5,6$
x'	$4,5,6$
p_1	$1,5$
p_2	$1,6$

vertex	list
q_1	$2,4$
q_2	$2,6$
r_1	$3,4$
r_2	$3,5$

Figure 18: Inequality gadget.

to introduce a segment s_i intersecting v, which will always receive color i.

To achieve this, we first need to transport the reference coloring to every gadget. We will do it using bundles of overlapping segments, the overall high-level idea is depicted in Figure 19. We make sure that a bundle consisting of overlapping segments colored 1,2,3 intersects all y_i's, and a bundle consisting of overlapping segments colored 4,5,6 intersects all x_i's. Observe that this already simulates the lists for every x_i, y_i ($i \in [n]$).

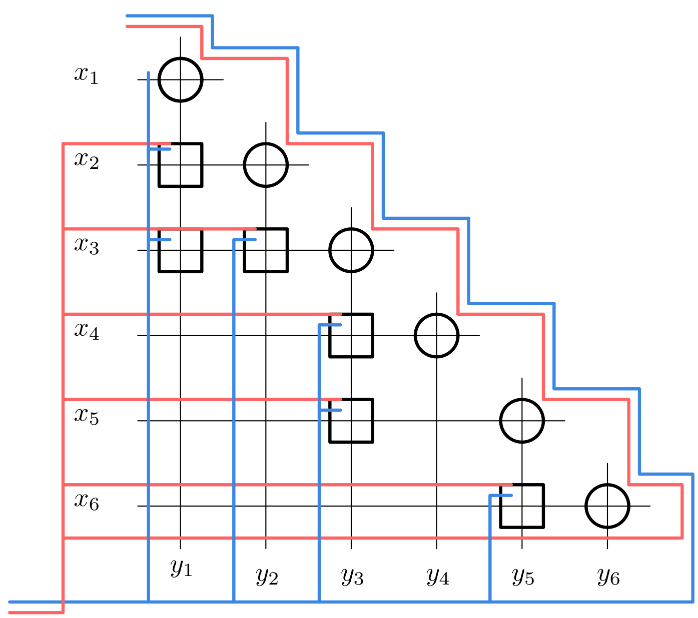


Figure 19: Reference coloring is transported to every gadget. Red and blue lines denote, respectively, triples of overlapping segments with colors 1,2,3, and 4,5,6. Parallel lines depicted close to each other are in fact overlapping. Segments $R_1, R_2, \ldots, R_6$ are positioned in the lower left corner of the picture.

Such a construction relies on a constant-size gadget, which allows us to split, join, or turn the reference coloring. In other words, we want to be able to split a bundle into two perpendicular bundles, carrying the same information (splitting, see e.g. red bundles in the left of Figure 19), join two bundles carrying distinct sets of colors into a single one (joining, this happens next to the inequality gadgets), or change the orientation of a bundle from a horizontal to vertical (turning, see the top-right bundles in Figure 19). We always need to make sure that the color of each overlapping segment in the bundle is uniquely determined.

The construction of the gadget for splitting a bundle of six segments is depicted in Figure 20. Turning a bundle or splitting a bundle of three colors can be easily realized by finishing unnecessary segments just after leaving the gadget. Also note that joining is actually the same as splitting. In order to join two perpendicular bundles, each carrying three colors, we add three segments to each of bundles (this will make sure that they receive the colors not appearing in the bundle), and attaching the extended bundles to a split gadget.

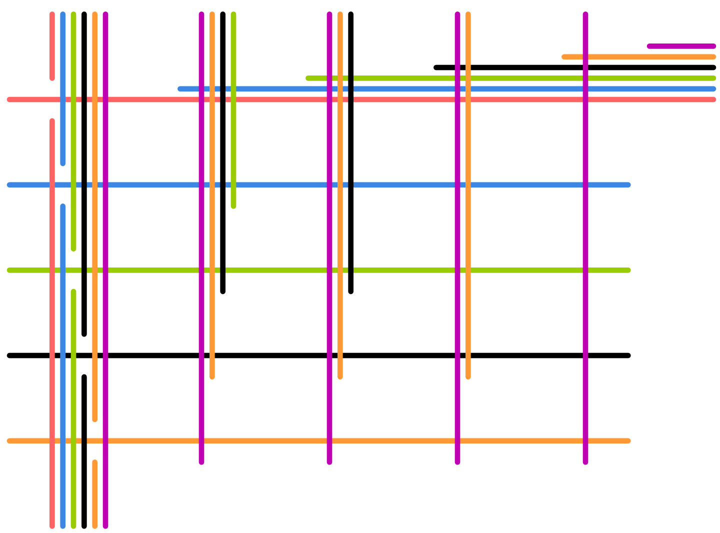


Figure 20: Split gadget for 6 colors. The parallel segments depicted close to each other are in fact overlapping. Observe that the depicted 6-coloring is the only possible (up to the permutation of colors).

Note that the number of segments in this gadget is constant. Now, the only thing left is to connect every segment in every gadget to appropriate segments carrying the reference coloring. This can easily be done using a constant number of additional segments per gadget (see Figure 21).

The total size of the construction increases by a constant factor, as we introduce $O(n)$ constant-size split gadgets. Thus an algorithm for 6-coloring the constructed 2-Dir graph in time $2^{o(n')}$ could be used to 3-color the input graph G in time $2^{o(n)}$, contradicting the ETH. $\square$

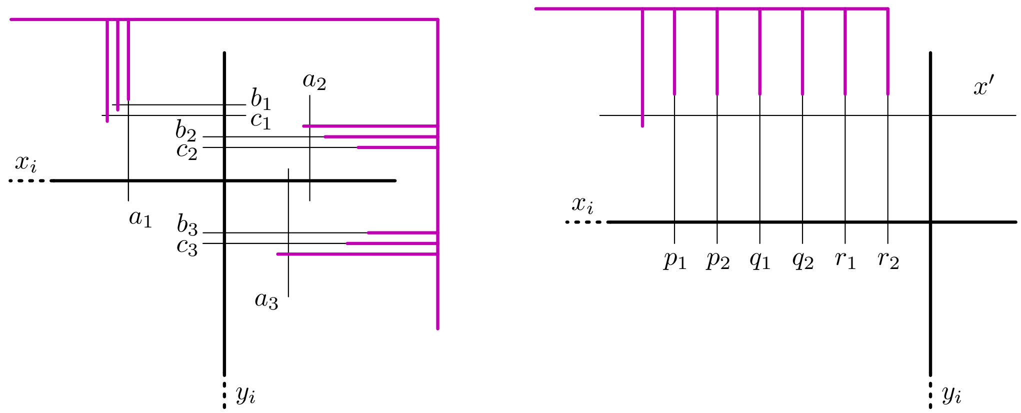


Figure 21: Simulation of lists for vertices in equality and inequality gadgets. Violet lines denote tuples of overlapping segments, carrying the reference coloring. We finish unwanted segments just after leaving the turning gadgets.

References

[1] J. Alber and J. Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004.

[2] É. Bonnet and P. Rzążewski. Optimality program in segment and string graphs. *CoRR*, abs/1712.08907, 2017.

[3] J. Cardinal. Computational geometry column 62. *SIGACT News*, 46(4):69–78, 2015.

[4] R. H. Chitnis, M. Hajiaghayi, and D. Marx. Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions). In *SODA 2014 Proc.*, pages 1782–1801, 2014.

[5] M. Cygan, F. V. Fomin, A. Golovnev, A. S. Kulikov, I. Mihajlin, J. W. Pachocki, and A. Socała. Tight lower bounds on graph embedding problems. *CoRR*, abs/1602.05016, 2016.

[6] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[7] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004.

[8] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Fixed-parameter algorithms for (k, r)-Center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005.

[9] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

[10] E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.

[11] E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

[12] E. D. Demaine and M. T. Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *GD 2014 Proc.*, pages 517–533, 2004.

[13] F. Dorn, F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In *STACS 2010 Proc.*, pages 251–262, 2010.

[14] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.

[15] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[16] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014.

[17] F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011.

[18] F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006.

[19] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[20] P. N. Klein and D. Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. In *ICALP 2012 Proc.*, pages 569–580, 2012.

[21] P. N. Klein and D. Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA 2014 Proc.*, pages 1812–1830, 2014.

[22] D. Marx. Efficient approximation schemes for geometric problems? In *ESA 2005 Proc.*, pages 448–459, 2005.

[23] D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In N. Bansal and I. Finocchi, editors, *ESA 2015 Proc.*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015.

[24] 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. SOCG 2014 Proc., pages 67:67–67:76, New York, NY, USA, 2014. ACM.

[25] 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. *CoRR*, abs/1612.01171, 2016.

[26] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, Jan. 1997.

[27] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen. Subexponential-time parameterized algorithm for Steiner Tree on planar graphs. In *STACS 2013 Proc.*, pages 353–364, 2013.

[28] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *FOCS 2014 Proc.*, pages 276–285. IEEE Computer Society, 2014.

[29] W. D. Smith and N. C. Wormald. Geometric separator theorems. available online at https://www.math.uwaterloo.ca/~nwormald/papers/focssep.ps.gz.

[30] W. D. Smith and N. C. Wormald. Geometric separator theorems and applications. FOCS 1998 Proc., pages 232–243, Washington, DC, USA, 1998. IEEE Computer Society.

[31] D. M. Thilikos. Fast sub-exponential algorithms and compactness in planar graphs. In *ESA 2011 Proc.*, pages 358–369, 2011.

[32] E. J. van Leeuwen and J. van Leeuwen. *Convex Polygon Intersection Graphs*, pages 377–388. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.