Fine-Grained Complexity of Coloring Unit Disks and Balls*

Csaba Biró[1], Édouard Bonnet[2], Dániel Marx[3], Tillmann Miltzow[4], and Paweł Rzążewski[5]

1 **Department of Mathematics, University of Louisville, Louisville, KY, USA**
csaba.biro@louisville.edu
2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
edouard.bonnet@dauphine.fr
3 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
dmarx@cs.bme.hu
4 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
t.miltzow@gmail.com
5 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary; and**
Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
p.rzazewski@mini.pw.edu.pl

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 unit 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 unit 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 $\exp\left(o(n^{\frac{1+\alpha}{2}})\right) = \exp\left(o(\sqrt{n\ell})\right)$, unless the ETH fails.

* Supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

33rd International Symposium on Computational Geometry (SoCG 2017).
Editors: Boris Aronov and Matthew J. Katz; Article No. 18; pp. 18:1–18:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

- 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 $\exp\left(n^{\frac{d-1+\alpha}{d}-\epsilon}\right) = \exp\left(O(n^{1-1/d-\epsilon}\ell^{1/d})\right)$ for any $\epsilon > 0$, unless the ETH fails.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases unit disk graphs, unit ball graphs, coloring, exact algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.18

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 [27, 21, 1], 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 [16], 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, 6, 5, 15, 11, 12, 7, 9, 8, 28, 14, 10, 17, 18, 2, 24, 25, 21]. More generally, for d-dimensional geometric problems, running times of the from $2^{O(n^{1-1/d})}$ or $n^{O(k^{1-1/d})}$ appear naturally, and Marx and Sidiropoulos [22] 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 unit 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 unit 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, unit disks graphs can contain arbitrary large cliques, and hence the 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 unit disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. What happens between these two extremes of constant number of colors and $\Theta(n)$ colors? 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 unit 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 $2^{o(n^{\frac{1+\alpha}{2}})} = 2^{o(\sqrt{n\ell})}$, 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. [13], 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 proof is not very specific to disks and can be easily adapted to, say, axis-parallel unit 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 (of arbitrary lengths) 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.*

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 [22]. 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 seems to be $n^{1-1/d}$ 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 unit 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 $2^{n^{\frac{d-1+\alpha}{d}-\epsilon}}$ for any $\epsilon > 0$, unless the ETH fails.*

Techniques. The upper bounds of Theorems 1 and 3 follow fairly easily using standard techniques. Clearly, the problem of coloring unit disks with ℓ colors makes sense 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 unit disks, then it is known that there is a balanced separator of size $O(\sqrt{n\ell})$ [23, 27, 26]. 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

[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!

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.

For the lower bound, 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.

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 forming 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 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.

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 [22], 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 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 $\{1, 2, \ldots, n\}$. 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 two positive integers g, d, we denote by $R[g, d]$ the *d-dimensional 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*.

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 size of the instance $I = (G, k, \zeta, \mathcal{C})$ of d-GRID 3-SAT is the total number of variables, i.e., $g^d k$.

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 $\rho : v \in V(G) \mapsto \{p_1^v, p_2^v, \ldots, p_\ell^v\} \in ([\ell]^d)^\ell$ mapping each cell v to a set of ℓ points in $[\ell]^d$.
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, say, $w = v + e_i$ (for some $i \in [d]$), and $p \in \rho(v)$ and $q \in \rho(w)$ receive the same color, then $p[i] \leqslant q[i]$ where $a[i] := a \cdot e_i$ is the i-th coordinate of a?

Here the size of the instance is the total number of points, i.e., $|V(G)|\ell \leqslant g^d \ell$.

3 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 4).

First we reduce 3-SAT to 2-GRID 3-SAT.

▶ **Theorem 4.** *For any $0 \leqslant \alpha \leqslant 1$ there is no algorithm solving* 2-GRID 3-SAT *with total size n and $k = \Theta(n^\alpha)$ variables per cell in time $2^{o(\sqrt{nk})} = 2^{o(n^{\frac{1+\alpha}{2}})}$, unless the ETH fails.*

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 5.** *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

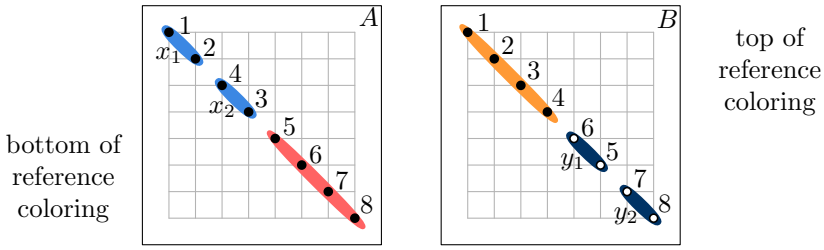


Figure 1 Cells of even parity contain the bottom half of the reference coloring as in cell A and cells of odd parity contain the top part of the reference coloring, as in cell B.

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.

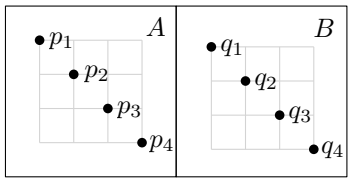
Standard cells. A *standard cell* is a cell where the points $p_1, \ldots, p_\ell$ are on the main diagonal, that is $p_i = (i, i)$ for every $i \in [\ell]$ (see cells A and B of Figure 2a). When we talk about the ordering of the points in a standard cell, we always mean the left-to-right (or equivalently, 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 $\bar{R}$.

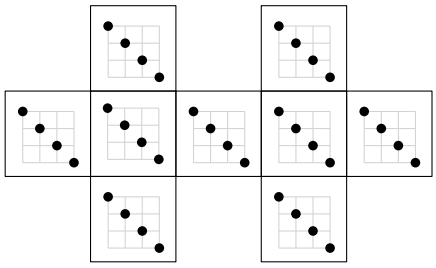
Variable-assignment cells. For each cell $v = (i, j) \in V(G)$, we introduce in F a standard cell $A(v) = (\delta i, \delta j)$, where δ is a large constant. The cells $A(v)$ for $v \in V(G)$ are responsible for encoding the truth assignment of variables in $\zeta(v)$. Therefore we call them *variable-assignment 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 $2^{\ell/4}$ 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 even, then for each $i \in [k]$, we interpret the coloring in the following way:

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



(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 2 Construction and usage of wires.

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

$p_{2k+2i-1} \mapsto 2i-1$, $p_{2k+2i} \mapsto 2i$as setting the variable$v_i$to true;
$p_{2k+2i-1} \mapsto 2i$, $p_{2k+2i} \mapsto 2i-1$as setting the variable$v_i$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 2a; 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 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. 2b). We say two cells are the same, if the point configuration and their coloring must be necessarily the same.

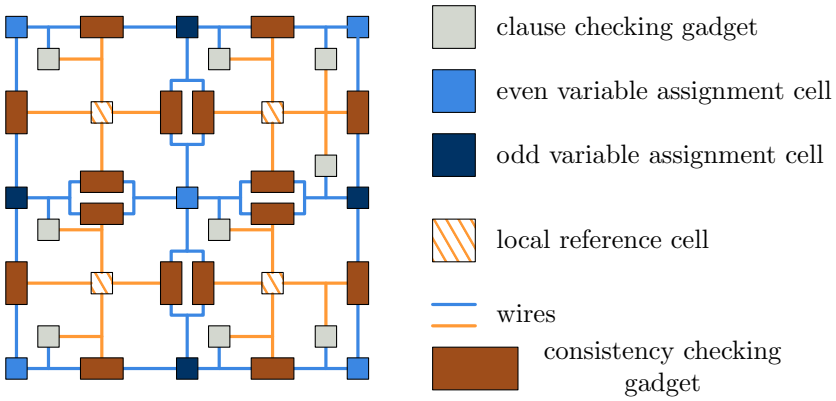


Figure 3 Illustration of the instance J. Each blue square represents a cell $A(v)$ corresponding to the cell v of I (light blue cells represent even cells and dark blue ones represent odd cells). The orange squares are local reference cells, which contain the reference coloring. Gray and brown squares 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 3 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 attain 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) we introduce two consistency gadgets, one for each face incident with vw. This gadget is responsible for ensuring the consistency on three different levels:

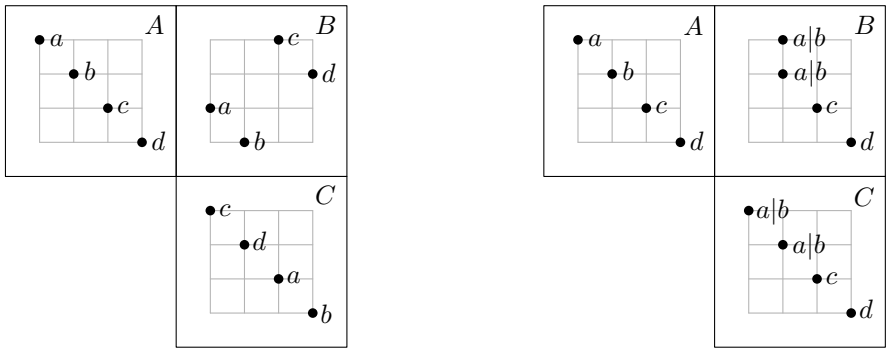
- to force all equality constraints $\mathcal{C}(v, w)$ to be satisfied,
- to ensure that each of $A(v)$ and $A(w)$ receives one of the standard colorings,
- to ensure that the local reference cell contains exactly the reference coloring.

This gadget also requires access to the reference coloring, so we join it with the appropriate local reference cell (see Fig. 3).

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. However, since wires allow us to create an exact copy of A, we can attach any constant number of gadgets to 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 size of 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 4a). The gadget is realized as follows. Let σ be



(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 4 Permutation gadget (left) and forgetting gadget (right), attached to cells A and C.

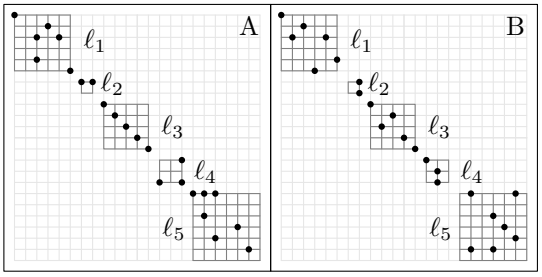
our target permutation. To the right of a standard cell A, we put a cell B. We place the points in B at the positions of 1's in the permutation matrix of σ. Below the cell B, we put a standard cell C. Is is straightforward to verify that in any feasible coloring of those three cells, for every $i \in [\ell]$, 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).

Forgetting color assignment. Besides permuting points and colors, it is also possible to forget the color assignment of some points. Figure 4b 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.

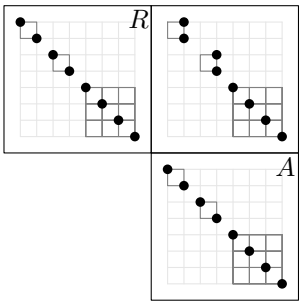
Parallelism. As we may have hinted in the previous paragraph, subparts of a given cell can act independently. In particular, this means that we can choose to forget any subset of information but preserve the rest. It is important to note that this is a more general phenomenon. Let $\ell_1, \ldots, \ell_t$ 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 5a. 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 5a).

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. 5b). 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.



(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 5 Boxes in adjacent cells with the same box-structure act independently from each other.

Clause gadget. We detail how a disjunction of three literals is encoded (see the left part of Figure 6). 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 $1, 3, 5, 2, 4, 6$, using the permutation gadget. Consider now cell U. It has one point at position $(3, 3)$ 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 $2i-1$ to the left of the box (say, the third column) and a point in the row $2i$ to the right of the box (the sixth column); if the variable appears negatively, we do the opposite: we place in cell B a point in the row $2i-1$ to the right of the box (sixth column) and a point in row $2i$ to the left of the box (third column). By construction, the colors to the right are not available to the point p. Therefore, the point p (and henceforth the whole set of cells) can be colored if and only if at least one literal is set to true 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

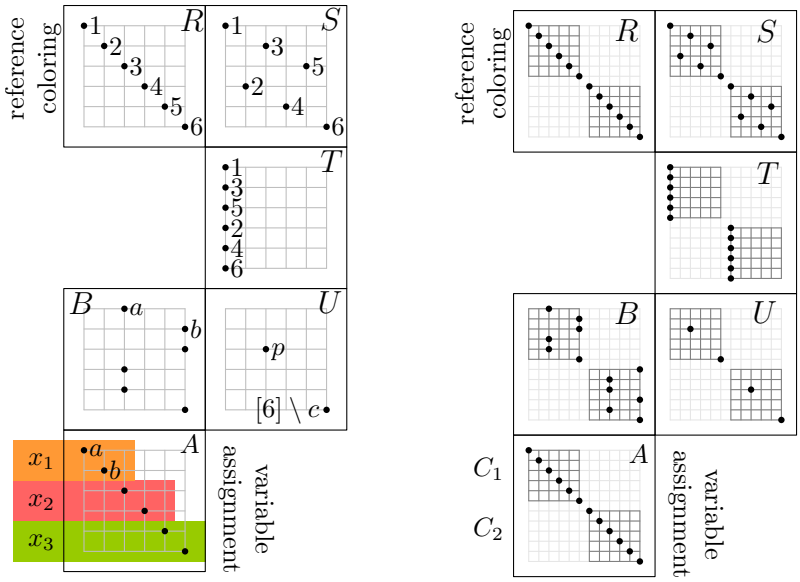


Figure 6 Illustration of the clause-checking gadget. To the left, one clause $x_1 \vee \neg x_2 \vee x_3$ is represented. To the right, two clauses are checked in parallel.

$2|C_1|+1, 2|C_1|+2, \ldots, 2|C_1|+2|C_2|$ 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 6. 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.

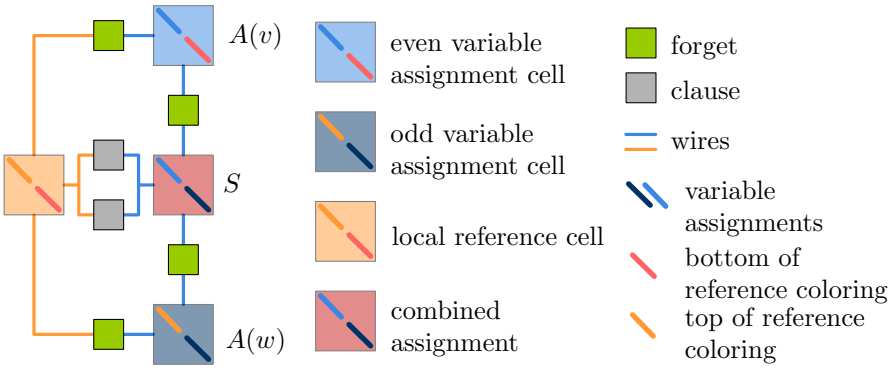


Figure 7 Overview of the consistency gadget. The clause gadgets serve to realize the equality constraints $\mathcal{C}(v, w)$.

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.

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 $p_1, p_2, \ldots, p_\ell$, the points of $A(w)$ by $q_1, q_2, \ldots, q_\ell$, and the points by R by $r_1, r_2, \ldots, r_\ell$ (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 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 one 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 $s_1, s_2, \ldots, s_\ell$ 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. The whole consistency gadget is displayed schematically in Figure 7.

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

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 the 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. 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 forget gadget, we know that for every coloring φ, we have: $\varphi(r_f) = \varphi(q_f) = \varphi(r'_f)$ for all $f \in [2k]$, and $\varphi(r_f) = \varphi(p_f) = \varphi(r'_f)$ for all $f \in [4k] \setminus [2k]$. This proves the claim. ◀

▶ **Claim 7.**

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 6 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 $n' = g^2k$ is the total size of 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 4, contradicts the ETH. ◀

Now, to prove the lower bound in Theorem 1, we need to show a reduction from Partial 2-grid Coloring to the problem of coloring unit disk graphs. This reduction is fairly standard and uses a well-known approach [20, Theorems 1 and 3].

4 Higher Dimensional Lower Bounds

The following result is a generalization of Theorem 4 to higher dimensions.

▶ **Theorem 8.** *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 total size n and $k = \Theta(n^\alpha)$ variables per cell in time $2^{n^{\frac{d-1+\alpha}{d}-\epsilon}} = 2^{O(n^{1-1/d-\epsilon}k^{1/d})}$, unless the ETH fails.*

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

▶ **Theorem 9.** *For any integer $d \geqslant 3$, and reals $0 \leqslant \alpha \leqslant 1$ and $\epsilon > 0$, there is no $2^{n^{1-1/d-\epsilon}\ell^{1/d}}$ 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 final step in proving the lower bound in Theorem 3 is reducing Partial d-grid Coloring to ℓ-Coloring of intersection graph of d-dimensional unit balls. It is very similar to the one in Theorem 1 (see also [22, Theorem 3.1.]).

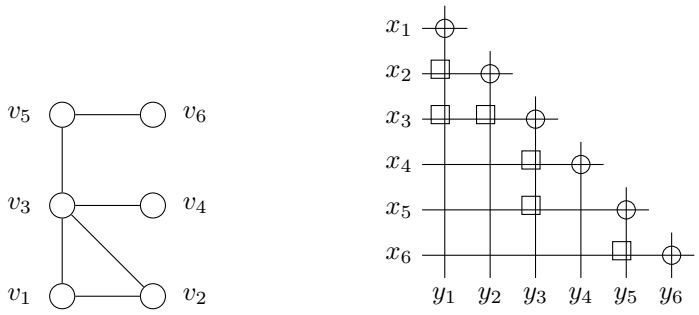


Figure 8 A graph G (left) and a high-level construction of G' (right). Circles denote equality gadgets and squares denote inequality gadgets.

5 Segments

In this section, we show that fatness is indeed necessary to obtain subexponential-time algorithm for coloring. We prove that a subexponential algorithm for coloring intersection graphs of segments (i.e., convex non-fat objects) with 6 colors would contradict the ETH.

Our construction works even if we use only horizontal or vertical segments. This class is known as 2-Dir. Note that if all segments are parallel, the intersection graph is an interval graph and, as such, can be colored in polynomial time. Moreover, we can even assume that the representation of the input graph is given. This is an important assumption, since the recognition of 2-Dir graphs is NP-complete (see Kratochvíl and Matoušek [19]).

Sketch of Proof of Theorem 2. 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 [3]). We construct a 2-Dir graph G', such that G is 3-colorable if and only if G' is 6-colorable.

Let the vertex set of G be $V = \{v_1, v_2 \ldots, v_n\}$. 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 8). Using appropriate gadgets we ensure that each x_i can only receive colors $\{1, 2, 3\}$, while each y_i can only receive colors $\{4, 5, 6\}$.

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. $f(x_i) + 3 \neq f(y_j)$ for all $i > j$ such that $v_i v_j$ is an edge of G.

This is achieved by using constant-size *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 8). 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 8).

The number of vertices of G' is $n' = \Theta(n)$, so the theorem follows. ◀

References

1 Jochen Alber and Jirí Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004. doi:10.1016/j.jalgor.2003.10.001.

2 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions). In *SODA 2014 Proc.*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.

3 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub W. Pachocki, and Arkadiusz Socała. Tight lower bounds on graph embedding problems. *CoRR*, abs/1602.05016, 2016. URL: http://arxiv.org/abs/1602.05016.

4 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004. doi:10.1137/S0895480103433410.

5 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-Center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

6 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.

7 Erik D. Demaine and Mohammad Taghi Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *GD 2014 Proc.*, pages 517–533, 2004. doi:10.1007/978-3-540-31843-9_57.

8 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

9 Erik D. Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008. doi:10.1007/s00493-008-2140-4.

10 Frederic Dorn, Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In *STACS 2010 Proc.*, pages 251–262, 2010. doi:10.4230/LIPIcs.STACS.2010.2459.

11 Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008. doi:10.1016/j.cosrev.2008.02.004.

12 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

13 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/j.jcss.2014.04.015.

14 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011. doi:10.1016/j.ipl.2011.05.016.

15 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.

16 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

17 Philip N. Klein and Dániel Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. In *ICALP 2012 Proc.*, pages 569–580, 2012. doi:10.1007/978-3-642-31594-7_48.

18 Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA 2014 Proc.*, pages 1812–1830, 2014. doi:10.1137/1.9781611973402.131.

19 J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

20 Dániel Marx. Efficient approximation schemes for geometric problems? In *ESA 2005 Proc.*, pages 448–459, 2005. doi:10.1007/11561071_41.

21 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In Nikhil Bansal and Irene Finocchi, editors, *ESA 2015 Proc.*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015. doi:10.1007/978-3-662-48350-3_72.

22 Dániel Marx and Anastasios 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 Thirtieth Annual Symposium on Computational Geometry*, SOCG 2014 Proc., pages 67:67–67:76, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582124.

23 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, January 1997. doi:10.1145/256292.256294.

24 Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for Steiner Tree on planar graphs. In *STACS 2013 Proc.*, pages 353–364, 2013. doi:10.4230/LIPIcs.STACS.2013.353.

25 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *FOCS 2014 Proc.*, pages 276–285. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.37.

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

27 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 Proc., pages 232–243, Washington, DC, USA, 1998. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=795664.796397.

28 Dimitrios M. Thilikos. Fast sub-exponential algorithms and compactness in planar graphs. In *ESA 2011 Proc.*, pages 358–369, 2011. doi:10.1007/978-3-642-23719-5_31.