Subexponential Parameterized Algorithms for Graphs of Polynomial Growth[*]

Dániel Marx[1] and Marcin Pilipczuk[2]

1   Institute for Computer Science and Control, Hungarian Academy of Sciences
    (MTA SZTAKI), Budapest, Hungary
    dmarx@cs.bme.hu
2   Institute of Informatics, University of Warsaw, Warsaw, Poland
    marcin.pilipczuk@mimuw.edu.pl

Abstract

We show that for a number of parameterized problems for which only $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ time algorithms are known on general graphs, subexponential parameterized algorithms with running time $2^{\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log^2 k)} n^{\mathcal{O}(1)}$ are possible for graphs of polynomial growth with growth rate (degree) δ, that is, if we assume that every ball of radius r contains only $\mathcal{O}(r^\delta)$ vertices. The algorithms use the technique of *low-treewidth pattern covering,* introduced by Fomin et al. [18] for planar graphs; here we show how this strategy can be made to work for graphs of polynomial growth.

Formally, we prove that, given a graph G of polynomial growth with growth rate δ and an integer k, one can in randomized polynomial time find a subset $A \subseteq V(G)$ such that on one hand the treewidth of $G[A]$ is $\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log k)$, and on the other hand for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is $2^{-\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log^2 k)}$. Together with standard dynamic programming techniques on graphs of bounded treewidth, this statement gives subexponential parameterized algorithms for a number of subgraph search problems, such as Long Path or Steiner Tree, in graphs of polynomial growth.

We complement the algorithm with an almost tight lower bound for Long Path: unless the Exponential Time Hypothesis fails, no parameterized algorithm with running time $2^{k^{1-\frac{1}{\delta}-\varepsilon}} n^{\mathcal{O}(1)}$ is possible for any $\varepsilon > 0$ and any integer $\delta \geq 3$.

1   Introduction

In recent years, research on parameterized algorithms had a strong focus on understanding the optimal form of dependence on the parameter k in the running time $f(k) n^{\mathcal{O}(1)}$ of parameterized algorithms. For many of the classic algorithmic problems on graphs, algorithms with running time $2^{\mathcal{O}(k)} n^{O(1)}$ exist, and we know that this form of running time is best

---

possible, assuming the Exponential-Time Hypothesis (ETH) [8, 22, 26]. This means that we have an essentially tight understanding of these problems when considering graphs in their full generality, but it does not rule out the possibility of improved algorithms when restricted to some class of graphs. Indeed, many of these problems become significantly easier on certain important graph classes. The most well-studied form of this improvement is the so-called "square root phenomenon" on planar graphs (and some if its generalizations): there is a large number of parameterized problems that admit $2^{O(\sqrt{k} \cdot \mathrm{polylog} k)} n^{O(1)}$ time algorithms on planar graphs [7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 23, 24, 29, 30, 31]. Many of these positive results can be explained by the theory of bidimensionality [11] and explicity or implicitly rely on the relation between treewidth and grid minors.

Very recently, a superset of the present authors showed a new technique to obtain subexponential algorithms in planar graphs for problems related to the SUBGRAPH ISOMORPHISM problem [18], such as the LONG PATH problem of finding a simple path of length k in the input graph. The approach of [18] can be summarized as follows: a randomized polynomial-time algorithm is showed that, given a planar graph G and an integer k, selects a random induced subgraph of treewidth sublinear in k in such a manner that, for every connected k-vertex subgraph H of G, the probability that H survives in the selected subgraph is inversely-subexponential in k. Such a statement, dubbed *low-treewidth pattern covering*, together with standard dynamic programming techniques on graphs of bounded treewidth, gives subexponential algorithms for a much wider range of SUBGRAPH ISOMORPHISM-type problems than bidimensionality; for example, while bidimensionality provides a subexponential algorithm for LONG PATH in undirected graphs, it seems that the new approach of [18] is needed for directed graphs.

The proof of the low treewidth pattern covering theorem of [18] involves a number of different partitioning techniques in planar graphs. In this work, we take one of these techniques – called *clustering procedure*, based on the metric decomposition tool of Linial and Saks [25] and the recursive decomposition used in the construction of Bartal's hierarchically well-separated trees (so-called HSTs) [3] – and observe that it is perfectly suited to tackle the so-called *graphs of polynomial growth*.

To explain this concept formally, let us introduce some notation. All graphs in this paper are unweighted, and the distance function $\mathrm{dist}_G(u, v)$ measures the minimum possible number of edges on a path from u to v in G. For a graph G, integer r, and vertex $v \in V(G)$ by $B_G(v, r)$ we denote the set of vertices $w \in V(G)$ that are within distance *less than r* from v in G, $B_G(v, r) = \{w \in V(G) : \mathrm{dist}_G(v, w) < r\}$, while by $\partial B_G(v, r)$ we denote the set of vertices within distance *exactly r*, that is, $\partial B_G(v, r) = \{w \in V(G) : \mathrm{dist}_G(v, w) = r\}$. We omit the subscript if the graph is clear from the context.

▶ **Definition 1.1** (polynomial growth, [4]). We say that a graph G (or a graph class $\mathcal{G}$) has *polynomial growth* of degree (growth rate) δ if there exists a universal constant C such that for (every graph $G \in \mathcal{G}$ and) every radius r and every vertex $v \in V(G)$ we have

$$|B(v, r)| \leq C \cdot r^\delta.$$

The algorithmic consequences (and some of its variants) of this definition have been studied in the literature in various contexts (see, for example, [2, 21, 4, 1]). A standard example of a graph of polynomial growth with degree δ is a δ-dimensional grid. Graph classes of polynomial growth include graphs of bounded doubling dimension (with unit-weight edges), a popular assumption restricting the growth of a metric space in approximation algorithms or routing in networks (cf. the thesis [5] of Chan or [1] and references therein).

Our main result is the following low treewidth pattern covering statement.

▶ **Theorem 1.2.** *For every graph class $\mathcal{G}$ of polynomial growth with growth rate δ, there exists a polynomial-time randomized algorithm that, given a graph $G \in \mathcal{G}$ and an integer k, outputs a subset $A \subseteq V(G)$ with the following properties:*
1. *the treedepth of $G[A]$ is $\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log k)$;*

2. *for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is $2^{-\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log^2 k)}$.*

Note that Theorem 1.2 uses the notion of *treedepth*, a much more restrictive graph measure than treewidth (cf. [28]), that in particular implies the same treewidth bound. Thus, together with standard dynamic programming techniques on graphs of bounded treewidth, Theorem 1.2 gives the following.

▶ **Corollary 1.3.** *There exist randomized parameterized algorithms with running time bound $2^{\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log^2 k)} n^{\mathcal{O}(1)}$ for* LONG PATH *and* STEINER TREE *parameterized by the size of the solution, when restricted to a graph class of polynomial growth with growth rate δ.*

In the corollary above we only listed the two most classic applications, refraining from repeating the lengthy discussion on the applications of low treewidth pattern covering statements that can be found in the introduction of Fomin et al. [18].

We complement the algorithmic statement of Theorem 1.2 with the following lower bound.

▶ **Theorem 1.4.** *If there exists an integer $\delta \geq 3$, a real $\varepsilon > 0$, and an algorithm that decides if a given subgraph of a δ-dimensional grid of side length n contains a Hamiltonian path in time $2^{\mathcal{O}(n^{\delta-1-\varepsilon})}$, then the ETH fails.*

Since a subgraph of a δ-dimensional grid of side length n has polynomial growth with degree at most δ and at most n^{δ} vertices, Theorem 1.4 shows that, unless the ETH fails, one cannot hope for a better term than $k^{1-\frac{1}{\delta}}$ in the low treewidth pattern covering statement as in Theorem 1.2.

2 Upper bound: proof of Theorem 1.2

In this section we prove Theorem 1.2. Without loss of generality, we assume $k \geq 4$.

Our main tool is a clustering procedure, or metric decomposition tool of [25], which can be informally described as follows. As long as the analysed graph G is not empty, we carve out a new cluster as follows. We pick any vertex $v \in V(G)$ as a center of the new cluster, and set its radius $r := 1$. Iteratively, with some chosen probablity p, we accept the current radius, and with the remaining probability $1 - p$ we increase r by one and repeat. That is, we choose r with geometric distribution with success probability p. Once a radius r is accepted, we set $B_G(v, r)$ as a new cluster, and delete $B_G(v, r) \cup \partial B_G(v, r)$ from G. In this manner, $B_G(v, r)$ is carved out as a separated cluster, at the cost of sacrificing $\partial B_G(v, r)$. A typical usage would be as follows: If one chooses p of the order of k^{-1}, then a simple analysis shows that every cluster has radius $\mathcal{O}(k \log n)$ w.h.p., while a fixed set $X \subseteq V(G)$ of size k is fully retained in the union of clusters with constant probability.

We apply the aforementioned clustering procedure in two steps. In the first one, we use $p \sim k^{-1}$ and the goal is to chop the graph into components of radius $\mathcal{O}(k \log k)$, which – by the polynomial growth property – are of polynomial size. The polynomial size bound is crucial for the second phase, when we consider every component independently, sparsifying it further using the clustering procedure with much higher cutoff probability, namely $p \sim k^{-\frac{1}{1+\delta}}$. These two steps are described in the subsequent two subsections.

We remark here that, because we rely only on the clustering procedure, and not the other arguments of [18], we do not need the assumption on the connectivity of the pattern $G[X]$. This assumption was essential for the planar case of [18].

2.1 Chopping the graph into parts of polynomial size

The goal of the first step is to delete a number of vertices from the graph so that on one hand every connected component of G has radius $\mathcal{O}(k \log k)$, and on the other hand the probability of deleting a vertex from an unknown vertex set $X \subseteq V(G)$ of size at most k is small. The proof of the following lemma is of the same nature as the clustering step in [18, Section 4.1 of the full version], with one subtlety: the obtained radii are of order $k \log k$ instead of $k \log n$. This improvement, crucial for the second step, heavily depends on the polynomial growth property.

▶ **Lemma 2.1.** *Let $\mathcal{G}$ be a graph class of polynomial growth with growth rate δ. There exists a constant $c_r > 0$ and a polynomial-time randomized algorithm that, given a graph $G \in \mathcal{G}$ and positive integer $k \geq 4$, outputs a subset $A \subseteq V(G)$ such that*
1. *every connected component of $G[A]$ is of radius at most $c_r k \log k$;*
2. *for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is at least $17/256$.*

Proof. For a sufficiently large constant $c_r > 0$ depending on the graph class $\mathcal{G}$, we perform the following iterative process. We start with $G_0 := G$ and $A_0 := \emptyset$. In i-th iteration ($i = 1, 2, 3, \ldots$), we consider the graph G_{i-1}. If the graph G_{i-1} is empty, we stop. Otherwise, we pick an arbitrary vertex $v_i \in V(G_{i-1})$ and pick a radius r_i according to the geometric distribution with success probability $1/k$, capped at value $R := c_r k \log k$ (i.e., if the choice of the radius is greater than R, we set $r_i := R$). For further analysis, we would like to look at the choice of the radius r_i as the following iterative process: we start with $r_i = 1$ and iteratively accept the current radius with probability $1/k$ or increase it by one and repeat with probability $1 - 1/k$, stopping unconditionally at radius R. Given v_i and r_i, we set $A_i := A_{i-1} \cup B_{G_{i-1}}(v_i, r_i)$ and $G_i := G_{i-1} - (B_{G_{i-1}}(v_i, r_i) \cup \partial B_{G_{i-1}}(v_i, r_i))$. That is, we remove from G_i all vertices within distance at most r_i from v_i, while retaining in A_i only those that are within distance less than r_i.

Clearly, as we remove a vertex from G_i at every step, the process stops after at most $|V(G)|$ steps. Let ι be the last index of the iteration. Consider the graph $G' := G[A_\iota]$. Recall that in the i-th step we put $B_{G_{i-1}}(v_i, r_i)$ into A_i, but remove not only $B_{G_{i-1}}(v_i, r_i)$ from G_{i-1} but also $\partial B_{G_{i-1}}(v_i, r_i) = N_{G_{i-1}}(B_{G_{i-1}}(v_i, r_i))$. Consequently, the vertex sets of the connected components of G' are exactly sets $B_{G_{i-1}}(v_i, r_i)$ for $1 \leq i \leq \iota$. Since the radii r_i are capped at value $R = c_r k \log k$, every connected component of G' has radius at most R.

We now claim the following.

▶ **Claim 2.2.** *For every $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq V(G')$ is at least $17/256$.*

Proof. Fix $X \subseteq V(G)$ of size at most k. Note that $X \not\subseteq V(G')$ only if at some iteration i, some vertex $x \in X$ is exactly within distance r_i from v_i in the graph G_{i-1}. We now bound the probability that this happens, split into two subcases: either $r_i = R$ or $r_i < R$.

Case 1: hitting a vertex within distance $r_i = R$.   Let $Y = \bigcup_{x \in X} B_G(x, R+1)$. Note that if $x \in X$ is exactly within distance $r_i \leq R$ from v_i in the graph G_{i-1}, then necessarily $v_i \in Y$. On the other hand, by the polynomial growth property,

$$|Y| \leq k \cdot C \cdot (R+1)^\delta = Ck(c_r k \log k + 1)^\delta = \mathcal{O}(k^{\delta+1} \log^\delta k).$$

We consider ourselves *lucky* if whenever $v_i \in Y$, we have $r_i < R$, that is, the process choosing r_i does not hit the cap of R for every center in Y. Note that, for a fixed iteration i, we have

$$\Pr(r_i = R) = \left(1 - \frac{1}{k}\right)^{R-1} = \left(1 - \frac{1}{k}\right)^{c_r k \log k - 1} \le k^{-0.1 \cdot c_r}.$$

Thus, for sufficiently large constant c_r (depending only on C and δ), we have that

$$\Pr(r_i = R) < (k \cdot |Y|)^{-1}.$$

We infer that, for such a choice of c_r, the probability that we are not lucky is at most $1/k$.

Case 2: hitting a vertex within distance $r_i < R$. It is convenient to think here of the choice of the radius r_i as an interative process that starts from $r_i = 1$, accepts the current radius with probability $1/k$, or increases it by one and repeats with probability $1 - 1/k$. For a fixed iteration i and a choice of v_i, consider a potential radius $r_i < R$ when there is a vertex $x \in X$ within distance exactly r_i from v_i in G_{i-1}. If we do not accept this radius (which happens with probability $1 - 1/k$), the vertex x is included in $B_{G_{i-1}}(v_i, r_i)$ and is surely included in G'. Consequently, in the whole process we care about not accepting a given radius only k times, at most once for every vertex $x \in X$. We infer that the probability that for some iteration i there is a vertex $x \in X$ within distance exactly r_i from v_i and $r_i < R$ is at most $1 - (1 - 1/k)^k$.

Considering both cases, by union bound, the probability that $X \subseteq V(G')$ is at least

$$1 - \left(1 - \left(1 - \frac{1}{k}\right)^k + \frac{1}{k}\right) = \left(1 - \frac{1}{k}\right)^k - \frac{1}{k} \ge \frac{17}{256}.$$

The last estimate uses the assumption $k \ge 4$. ◄

Claim 2.2 concludes the proof of Lemma 2.1. ◄

2.2 Handling a component of polynomial size

▶ **Lemma 2.3.** *Let $\mathcal{G}$ be a graph class of polynomial growth with growth rate δ. For every constant $c_r > 0$ there exists a constant $c > 0$ and a polynomial-time randomized algorithm that, given a positive integer k, and a connected graph $G \in \mathcal{G}$ of radius $c_r k \log k$, outputs a subset $A \subseteq V(G)$ such that*
1. *the treedepth of $G[A]$ is $\mathcal{O}(k^{1-\frac{1}{1+\delta}} \log k)$;*
2. *for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is at least $2^{-c \cdot |X| \cdot k^{-\frac{1}{1+\delta}} \cdot \log^2 k}$.*

We emphasize here the linear dependency on $|X|$ in the exponent of the probability bound. This dependency, similarly as in the analysis of [18], allows us to easily analyse independent runs of the algorithm on multiple connected components.

To prove Lemma 2.3, we again use the clustering procedure, but with a significantly higher cutoff probability, namely of the order of $k^{-\frac{1}{1+\delta}}$, as opposed to k^{-1} from the previous section. This yields clusters of sublinear size, namely of size roughly $k^{\frac{\delta}{1+\delta}}$. However, this comes with a cost: we can no longer claim that the solution X survives in the clustered graph with large probability, but – on average – $k^{\frac{\delta}{1+\delta}}$ vertices of X of size k will be deleted by the clustering clustering procedure. To recover from that, we crucially depend on the fact that the graph

has size polynomial in k: there is only a subexponential, namely $\binom{\text{poly}(k)}{k^{\frac{\delta}{1+\delta}}} = 2^{\mathcal{O}(k^{1-\frac{1}{1+\delta}}\log k)}$, number of choices for the removed vertices of X, and we can afford to guess them.

Let us make a quick comparison with the techniques of [18]. The usage of the clustering technique in Lemma 2.3 is significantly different than the one in [18, Section 4.1 of the full version]: we choose a higher cutoff probability, which leads to smaller radii, at the cost of allowing some vertices of the set X on the boundary (that need to be subsequently guessed). The charging argument used here (Claim 2.5) is inspired by the argument of [18, Claim 28 in the full version]. However, the reason why we obtain sublinear treedepth (Claim 2.4) and the consequent tradeoffs in the exponent are specific to our polynomial growth setting.

Let us now proceed with the formal arguments.

Proof of Lemma 2.3. The random process we employ is similar to the one of the previous section, but more involved. Let $c'_r > 0$ be a constant to be fixed later.

We start with $G_0 = G$, $A_0 = \emptyset$ and $B_0 = \emptyset$. In the i-th iteration of the process, we consider the graph G_{i-1}. If the graph G_{i-1} is empty, we stop. Otherwise, we pick an arbitrary vertex $v_i \in V(G_{i-1})$ and pick a radius r_i according to the geometric distribution with success probability $k^{-1/(1+\delta)} \log k$, capped at value $R' := c'_r k^{1/(1+\delta)}$ (i.e., as before, if the choice of the radius is greater than R', we set $r_i := R'$). In other words, we start with $r_i = 1$ and iteratively accept the current radius with probability $k^{-1/(1+\delta)} \log k$ or increase it by one and repeat with the remaining probability, stopping unconditionally at radius R'.

As before, we set $A_i := A_{i-1} \cup B_{G_{i-1}}(v_i, r_i)$ and $G_i := G_{i-1} - (B_{G_{i-1}}(v_i, r_i) \cup \partial B_{G_{i-1}}(v_i, r_i))$. However, now, as the radii are smaller, we may want to retain some vertices of $\partial B_{G_{i-1}}(v_i, r_i)$, as they can be part of the vertex set X; for this, we use the sets B_i. With probability $1 - 1/(k|V(G)|)$ we put $P_i = \emptyset$ and $B_i = B_{i-1}$. With the remaining probability, we proceed as follows. Uniformly at random, we choose a number $1 \leq \ell_i \leq k^{1-1/(1+\delta)} \log k$ and a set P_i of ℓ_i vertices of $\partial B_{G_{i-1}}(v_i, r_i)$ (or all of them, if there are less than ℓ_i vertices in this set). We put $B_i := B_{i-1} \cup P_i$.

Let i_0 be the index of the last iteration. If $|B_{i_0}| > k^{1-1/(1+\delta)} \log k$, then we output $A = \emptyset$. Otherwise, we output $A := A_{i_0} \cup B_{i_0}$. Let us now verify that A has the desired properties.

▶ **Claim 2.4.** *The treedepth of $G[A]$ is $\mathcal{O}(k^{\delta/(1+\delta)} \log k)$.*

Proof. The claim is trivial if $A = \emptyset$, so assume otherwise; in particular, $|B_{i_0}| \leq k^{1-1/(1+\delta)} \log k$. We use the following inductive definition of treedepth: the treedepth of an empty graph is 0, while for any graph G on at least one vertex we have that

$$\text{treedepth}(G) = \begin{cases} 1 + \min\{\text{treedepth}(G - v) : v \in V(G)\} & \text{if } G \text{ is connected} \\ \max\{\text{treedepth}(C) : C \text{ connected component of } G\} & \text{otherwise.} \end{cases}$$

Upon deleting from $G[A]$ the at most $k^{1-1/(1+\delta)} \log k$ vertices of B_{i_0}, we are left with $G[A_{i_0}]$. Similarly as in the previous section, every connected component of $G[A_{i_0}]$ is of radius at most $R' = c'_r k^{1/(1+\delta)}$. Consequently, every connected component of $G[A_{i_0}]$ is of size at most $C \cdot (c'_r)^\delta k^{\delta/(1+\delta)}$. The claim follows. ◀

▶ **Claim 2.5.** *For every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is at least $2^{-c|X|k^{-1/(1+\delta)} \log^2 k}$ for some constant $c > 0$ depending only on c_r, δ, and C.*

Proof. Fix a vertex set X. The claim is trivial for $X = \emptyset$ so assume otherwise. In particular, as $|X| \geq 1$, then we can estimate the desired probability as

$$2^{-c|X|k^{-1/(1+\delta)} \log^2 k} \leq 2^{-ck^{-1/(1+\delta)} \log^2 k} = 1 - \Omega\left(\frac{\log^2 k}{k^{1/(1+\delta)}}\right). \tag{1}$$

Consider a fixed iteration i, and the moment when, knowing v_i, we choose the radius r_i. Given G_{i-1} and v_i, we say that a radius r is *bad* if

$$\left| X \cap \partial B_{G_{i-1}}(v_i, r) \right| > \left(k^{-1/(1+\delta)} \log k \right) \cdot \left| X \cap B_{G_{i-1}}(v_i, r) \right|. \tag{2}$$

Let $1 \leq r^0 < r^1 < r^2 < \ldots < r^t$ be a sequence of bad radii. First, note that $X \cap \partial B(v_i, r^0) \neq \emptyset$, and thus $|X \cap B(v_i, r^1)| \geq 1$. Furthermore, as for every $j \geq 1$ we have $\partial B(v_i, r^j) \subseteq B(v_i, r^{j+1})$, we have

$$\left| X \cap B(v_i, r^{j+1}) \right| \geq \left(1 + k^{-1/(1+\delta) \log k} \right) |X \cap B(v_i, r^j)|.$$

Consequently,

$$\left| X \cap B(v_i, r^j) \right| \geq \left(1 + k^{-1/(1+\delta) \log k} \right)^{j-1}.$$

Since $|X| \leq k$, we infer that

$$t < 10 k^{1/(1+\delta)}. \tag{3}$$

We are interested in the following event $\mathbf{A}$: every chosen radius r_i is not bad and is smaller than R' (i.e., we did not hit the cap of R'). Recall the iterative interpretation of the choice of the radii r_i: we start with $r_i = 1$, accept the current radius with probability $k^{-1/(1+\delta)} \log k$, or increase r_i by one and repeat with the remaining probability. Thus, we are interested in the intersection of the following two events: we do not accept any bad radius, but we accept some good radius before the cap R'.

Whenever we do not accept a bad radius r, a vertex of $X \cap \partial B(v_i, r)$ is included in $B(v_i, r_i) \subseteq A_i$. Consequently, in the whole algorithm we encounter at most $|X|$ bad radii; each is independently accepted with probability $k^{-1/(1+\delta)} \log k$.

By (3), in a fixed iteration i there are at most $10 k^{1/(1+\delta)}$ bad radii. Consequently, if we count only acceptance of good radii, the probability that the radius r_i reaches the bound R' is at most

$$\left(1 - k^{-1/(1+\delta)} \log k \right)^{(c'_r - 10) k^{1/(1+\delta)}} \leq k^{-0.1 c'_r}.$$

Consequently, since $|V(G)| \leq C \cdot (c_r k \log k)^\delta$, by choosing c'_r large enough, we can ensure that the probability that there exists a radius r_i equal to R' is at most k^{-1}. Since the choices of acceptance of different radii are independent, we infer that the probability of the event $\mathbf{A}$ is at least

$$\left(1 - k^{-1} \right) \cdot \left(1 - k^{-1/(1+\delta)} \log k \right)^{|X|} \geq 2^{-c_1 |X| k^{-1/(1+\delta)} \log k}$$

for some positive constant c_1. Here, we have used (1) to estimate the first factor.

Assume that the event $\mathbf{A}$ happens, and let us fix one choice of v_i and r_i. Note that these choices determine the sets A_i and the graphs G_i; the only remaining random choices are whether to include some vertices into the sets B_i.

For an iteration i, define $X_i := X \cap \partial B_{G_{i-1}}(v_i, r_i)$. We are now considering the following event $\mathbf{B}$: in every iteration i we have $P_i = X_i$. Note that if $\mathbf{B}$ happens, then $X \subseteq A$. Thus, we need to estimate the probability of the event $\mathbf{B}$.

If $X_i = \emptyset$, then we guess so with probability $1 - 1/(k|V(G)|)$. As there are at most $|V(G)|$ iterations, with probability at least $1 - 1/k$ we will make correct decision in all iterations i for which $X_i = \emptyset$.

Consider now an iteration i for which $X_i \neq \emptyset$. Since the radius r_i is good, we have

$$\left| X \cap \partial B_{G_{i-1}}(v_i, r_i) \right| \leq k^{-1/(1+\delta)} \log k \left| X \cap B_{G_{i-1}}(v_i, r_i) \right|. \tag{4}$$

In particular, $|X \cap B_{G_{i-1}}(v_i, r_i)| \geq k^{1/(1+\delta)}/\log k$, and thus there are at most $k^{\delta/(1+\delta)} \log k$ such iterations. Furthermore,

$$\left| \bigcup_{i=1}^{i_0} X_i \right| \leq |X| k^{-1/(1+\delta)} \log k.$$

In every such iteration i, we need to correctly guess that X_i is nonempty $(1/(k|V(G)|)$ success probability), correctly guess $\ell_i = |X_i|$ (at least $1/k$ success probability) and correctly guess $P_i = X_i$ (at least $|V(G)|^{-|X_i|}$ success probability). All these choices are independent. Since $|V(G)|$ is bounded polynomially in k, the probability of the event **B** is at least

$$\left(1 - \frac{1}{k}\right) \cdot \prod_{i:X_i \neq \emptyset} \frac{1}{k|V(G)|} \cdot \frac{1}{k} \cdot \frac{1}{|V(G)|^{|X_i|}} \geq \left(1 - \frac{1}{k}\right) \cdot (|V(G)|^2 \cdot k)^{-|X| \cdot k^{-1/(1+\delta)} \log k}$$

$$\geq 2^{-c_2 |X| \cdot k^{-1/(1+\delta)} \log^2 k}$$

for some constant c_2 depending on c_r, δ, and C. This finishes the proof of the claim. ◄

Lemma 2.3 follows directly from Claims 2.4 and 2.5. ◄

2.3   Summary

Let us now wrap up the proof of Theorem 1.2, using Lemmata 2.1 and 2.3. We first apply the algorithm of Lemma 2.1 to the input graph G and integer k, obtaining a set $A_0 \subseteq V(G)$. Then, we apply the algorithm of Lemma 2.3 independently to every connected component C of $G[A_0]$, obtaining a set $A_C \subseteq C$; recall that every such component is of radius at most $R = c_r k \log k$. As the output A, we return the union of the returned sets A_C. Clearly, the treedepth bound holds. If we denote $X_C := X \cap C$ for a component C, we have that the probability that $X \subseteq A$ is at least

$$\frac{17}{256} \cdot \prod_C 2^{-c|X_C| k^{-1/(1+\delta)} \log^2 k} \geq \frac{17}{256} \cdot 2^{-ck^{1-1/(1+\delta)} \log^2 k}.$$

This finishes the proof of Theorem 1.2.

3   Lower bound: proof of Theorem 1.4

In this section we prove Theorem 1.4. The reduction is heavily inspired by the reduction for δ-dimensional Euclidean TSP by Marx and Sidiropolous [27]. In particular, our starting point is the same CSP pivot problem.

▶ **Theorem 3.1** ([27]). *For every fixed $\delta \geq 2$, there is a constant λ_δ such that for every constant $\varepsilon > 0$ an existence of an algorithm solving in time $2^{\mathcal{O}(n^{\delta-1-\varepsilon})}$ CSP instances with binary constraints, domain size at most λ_δ, and Gaifman graph being a δ-dimensional grid of side length n would refute ETH.*

Let us recall that a *binary CSP instance* consists of a *domain D*, a set V of *variables*, and a set E of *constraints*. Every constraint is a binary relation $\psi_{u,v} \subseteq D \times D$ that binds two variables $u, v \in V$. The goal is to find an assignment $\phi : V \to D$ that satisfies every
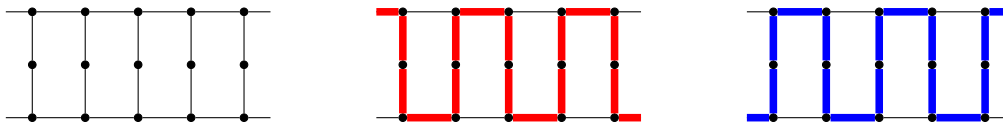
Figure 1 A 2-chain with two ways how a Hamiltonian path can traverse it, called henceforth *modes.*
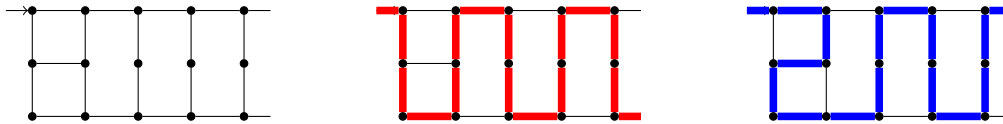
Figure 2 An endpoint of a 2-chain, allowing traversing the 2-chain in both modes.

constraint; a constraint $\psi_{u,v}$ is satisfied if $(\phi(u), \phi(v)) \in \psi_{u,v}$. The *Gaifman graph* of a binary CSP instance has vertex set V and an edge uv for every constraint $\psi_{u,v}$.

Similarly as in the case of [27], our goal is to take a given CSP instance as in Theorem 3.1 and turn it into a Hamiltonian path instance by local gadgets. That is, we are going to replace every variable of the CSP instance with a constant-size gadget (i.e., with size depending only on δ and λ_δ); the way the gadget is traversed by the Hamiltonian path indicates the choice of the value of the variable. The neighboring gadgets are wired up to ensure that the constraint binding them is satisfied.

More formally, let us fix an integer $\delta \geq 3$. The input of a reduction is a CSP instance as in Theorem 3.1: of domain size at most λ_δ and whose Gaifman graph is a δ-dimensional grid of size length n. The output is a subgraph of a δ-dimensional grid of side length cn for some constant c depending only on δ and λ_δ that has a Hamiltonian path if and only if the input CSP instance is satisfiable.

Let us fix a δ-dimensional graph of side length cn for some sufficiently large constant c to be defined later (we will see that $c = \Theta(\delta\lambda_\delta^2)$ suffices). We partition this grid into n^δ subgrids of side length c, each corresponding to a variable of the input CSP instance in a natural fashion.

3.1 2-chains

The base gadget of the construction is a 2-chain as presented on Figure 1. A direct check shows that there are two ways how a 2-chain can be traversed by a Hamiltonian path, as depicted on the figure.

Figure 2 shows a gadget present on both left and right endpoints of a 2-chain. As shown on the figure, it allows choosing how the 2-chain is traversed.

We will refer to the two depicted Hamiltonian paths of a 2-chain as *modes* of the chain. Given one of the horizontal edges of the 2-chain, a mode is *consistent* with this edge if the corresponding Hamiltonian path traverses the edge in question, and *inconsistent* otherwise.

We will attach various gadgets to 2-chains via one of the horizontal edges. To maintain the properties of the 2-chains, in particular the effectively two ways of traversing a 2-chain, we need to space out the attached gadgets. More formally, we partition every 2-chain into sufficiently long chunks (chunks of length 8 are more than sufficient), and allow gadgets to attach only to one of the two middle horizontal edges on one side of the chain (see Figure 3), with at most one gadget per chunk. A gadget is always attached to an edge e by adding two new vertices u and v near the edge e, in the same 2-dimensional plane as the 2-chain itself,
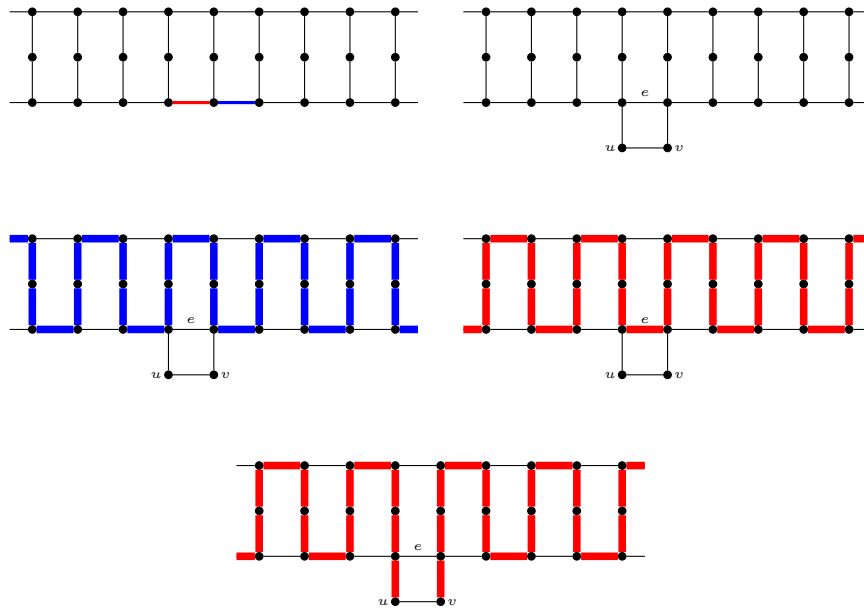
Figure 3 From top to bottom, left to right: a chunk on a 2-chain, with two attachment edges marked red and blue; a standard attachment of a gadget; three ways how a 2-chain with attached gadget can be traversed.

such that the endpoints of e, u, and v form a square. Properties of such an attachment can be summarized in the following straightforward claim.

▷ **Claim 3.2.** *Consider a chunk c on a 2-chain A, and a gadget attached to an edge e in c. Then every Hamiltonian path traverses c in one of the following three ways (see Figure 3):*

1. *as on Figure 1, inconsistently with e;*
2. *as on Figure 1, consistently with e;*
3. *as on Figure 1, consistently with e, but with the edge e replaced with an edge towards vertex u and towards vertex v.*

In particular, Claim 3.2 allows us to formally speak about a mode of a 2-chain, even if multiple gadgets are attached to it.

3.2 Placing 2-chains

For every variable of the input CSP instance, we create λ_δ 2-chains of length $L = \mathcal{O}(d\lambda_\delta)$ (to be determined later). They are positioned parallelly in the following fashion (see Figure 4): we choose an arbitrary 3-dimensional subspace of the δ-dimensional subgrid of sidelength c devoted to a particular variable, and place 2-chains such that the i-th 2-chain occupies vertices $\{0, 1, \ldots, L\} \times \{0, 1, 2\} \times \{i\}$. The edges indicated as attachment points for gadgets are on the one side of all chains.

All chains, for all variables, are wired up into a Hamiltonian path: for every variable, we connect the constructed 2-chains into a path in a straightforward fashion, we take an arbitrary Hamiltonian path of the original Gaifman graph of the input CSP instance (which is a δ-dimensional grid, and thus trivially admits a Hamiltonian path), and connect endpoints of the 2-chains in the same order using simple paths. This is straightforward to perform if we space out the variable gadgets enough.
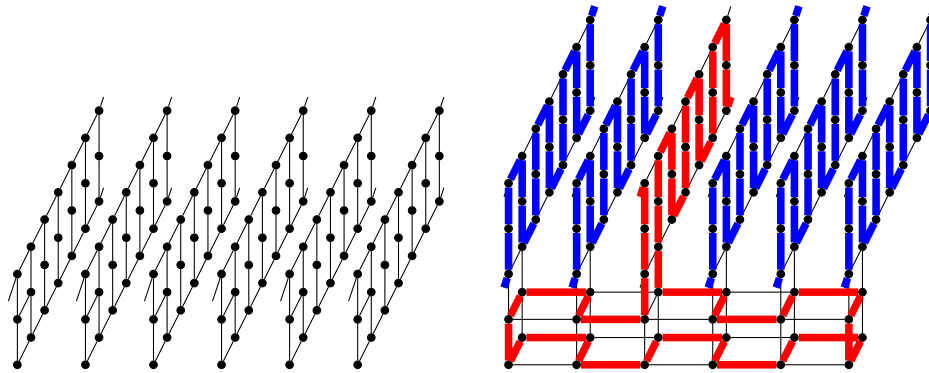
Figure 4 Left: Placing parallel 2-chains for a single variable x. Right: A tube gadget attached to the 2-chains, with intended Hamiltonian path.

Since all constructed 2-chains are isomorphic, we indicate one mode of a 2-chain as a *low mode*, and the other one as *high mode*. Our goal is to introduce gadgets that (i) ensure that for every variable, exactly one of the corresponding 2-chains is in high mode, indicating the choice of the value for this variable; (ii) for every two variables that are bound by a constraint, for every pair of values that is forbidden by the constraint, ensure that the two variables in question do not attain the values in question at the same time, that is, the corresponding two 2-chains are not both in high mode at the same time.

3.3 OR-checks

The construction of 2-chains allow us to implement a simple "OR" constraint on two 2-chains. Consider two 2-chains A and B, and two horizontal edges e_A and e_B on A and B, respectively. By attaching an OR-check to these edges we mean the following construction:

1. we create vertices u_A and v_A near e_A as well as u_B and v_B near e_B, as in the description of gadget attachment;
2. we connect u_A to u_B by a path and v_A to v_B by a path.

If the 2-chains are spaced enough, it is straightforward to implement the above construction such that the resulting graph is a subgraph of a d-dimensional grid.

Claim 3.2 allows us to observe the following.

▶ **Claim 3.3.** *If A is traversed in a way consistent with e_A, then one can modify the Hamiltonian path traversing A so that it visits the OR gadget: replace e_A with a path traversing first a path from u_A to u_B, the edge $u_B v_B$, and then the path from v_B to v_A. A symmetrical claim holds if B is traversed in a way consistent with e_B.*

In the other direction, there is no Hamiltonian path that traverses both A and B in a way inconsistent with e_A and e_B, respectively.

We now observe that, by attaching OR-checks in a straightforward manner, we can ensure that:

1. for every variable x, at most one 2-chain corresponding to x is in high mode (we wire up every pair of 2-chains with an OR-check forbidding two high modes at the same time);
2. for every two variables x and y that are bound by a constraint ψ, for every pair of values (α_x, α_y) that is forbidden by the constraint ψ, the α_x-th 2-chain of x and the α_y-th 2-chain of y are not in the high mode at the same time.

We are left with ensuring that for every variable x, at least one of the corresponding 2-chains is in the high mode. This is the aim of the next gadget.

3.4   Tube gadget

Fix a variable x. Without loss of generality, we can assume that the first chunk of every 2-chain for x has not been used by the OR-checks introduced previously. Let e_i be the attachment edge of the i-th 2-chain that is consistent with the high mode of the 2-chain; note that the edges e_i lie next to each other (see Figure 4).

We create a $2 \times 2 \times \lambda_\delta$ grid, called henceforth a *tube gadget*, placed near the edges e_i, such that every edge e_i can be attached to an edge of the grid in a standard way discussed earlier. See Figure 4 for an illustration.

Since a $2 \times 2 \times \lambda_\delta$ grid admits a Hamiltonian cycle that traverses every edge in one of the first two dimensions, if the i-th chain is traversed in high mode for some i, we can replace e_i on the Hamiltonian path with a traversal along the aforementioned Hamiltonian cycle. This observation, together with Claim 3.2, proves the following claim.

▶ **Claim 3.4.** *If there exists an index i such that the i-th 2-chain is traversed in high mode, then the Hamiltonian path of this 2-chain can be altered to visit every vertex of the $2 \times 2 \times \lambda_\delta$ grid.*

On the other hand, any Hamiltonian path of the entire graph needs to traverse at least one 2-chain in high mode, in order to visit the vertices of the $2 \times 2 \times \lambda_\delta$ grid.

3.5   Summary

The tube gadgets ensure that, for every variable, at least one corresponding 2-chain is in high mode. The first type of the attached OR-checks ensure that at most one such 2-chain is in high mode. Thus, effectively the gadgets introduced for a single variable x can be in one of λ_δ by choosing the 2-chain that is in high mode, which corresponds to the choice of the value for x in an assignment.

The second type of the attached OR-checks ensure that the values of the neighboring variables satisfy the constraint that binds them, completing the proof of the correctness of the reduction.

To conclude, let us observe that every 2-chain is attached to one tube gadget and $\mathcal{O}(\delta \lambda_\delta)$ OR-checks, and the whole gadget replacing a single variable takes part in $\mathcal{O}(\delta \lambda_\delta^2)$ OR-checks. Thus taking $L = \mathcal{O}(\delta \lambda_\delta^2)$ suffices. By leaving space of size $\mathcal{O}(\delta \lambda_\delta^2)$ between consecutive variable gadgets we can ensure more than enough space for all connections. This gives $c = \mathcal{O}(\delta \lambda_\delta^2)$, that is, the constructed graph is a subgraph of a d-dimensional grid of side length $\mathcal{O}(\delta \lambda_\delta^2 n)$, and admits a Hamiltonian path if and only if the input CSP instance is satisfiable. This finishes the proof of Theorem 1.4.

4   Conclusions

We have shown a low treewidth pattern covering statement for graphs of polynomial growth with subexponential term being $2^{k^{1-\frac{1}{1+\delta}}}$, where δ is the growth rate of the graph class. An almost tight lower bound shows that, assuming ETH, one should not hope for a better term than $2^{k^{1-\frac{1}{\delta}}}$.

Two natural questions arise. The first one is to close the gap between $\frac{1}{1+\delta}$ and $\frac{1}{\delta}$; we conjecture that our lower bound is tight, and the term $k^{1-\frac{1}{1+\delta}}$ in the running time bound

of Theorem 1.2 is only a shortfall of our algorithmic techniques. The second one is to derandomize the algorithms of this work and of [18]. The clustering step is the only step of the algorithm of [18] that we do not know how to derandomize, despite its resemblance to the construction of Bartal's HSTs [3] that was subsequently derandomized [6].

References

1   Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal*, page 75. IEEE Computer Society, 2006. `doi:10.1109/ICDCS.2006.72`.

2   Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In Phillip B. Gibbons and Paul G. Spirakis, editors, *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 49–55. ACM, 2005. `doi:10.1145/1073970.1073978`.

3   Yair Bartal. On approximating arbitrary metrices by tree metrics. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 161–168. ACM, 1998. `doi:10.1145/276698.276725`.

4   Vincent Blondel, Kyomin Jung, Pushmeet Kohli, and Devavrat Shah. Partition-merge: Distributed inference and modularity optimization. *CoRR*, abs/1309.6129, 2013. URL: `http://arxiv.org/abs/1309.6129`.

5   T. H. Hubert Chan. *Approximation Algorithms for Bounded Dimensional Metric Spaces.* PhD thesis, Carnagie Mellon University, 2007. Available at `http://i.cs.hku.hk/~hubert/thesis/thesis.pdf`.

6   Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge A. Plotkin. Approximating a finite metric by a small number of tree metrics. In *39th Annual Symposium on Foundations of Computer Science, FOCS'98, November 8-11, 1998, Palo Alto, California, USA*, pages 379–388. IEEE Computer Society, 1998. `doi:10.1109/SFCS.1998.743488`.

7   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*, pages 1782–1801, 2014. `doi:10.1137/1.9781611973402.129`.

8   Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

9   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`.

10 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`.

11 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.

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

13 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`.

14 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`.

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

16 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`.

17 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`.

18 Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering. In *FOCS*, pages 515–524, 2016. Full version available at `http://arxiv.org/abs/1604.05999`.

19 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`.

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

21 Ramakrishna Gummadi, Kyomin Jung, Devavrat Shah, and Ramavarapu S. Sreenivas. Computing the capacity region of a wireless network. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*, pages 1341–1349. IEEE, 2009. `doi:10.1109/INFCOM.2009.5062049`.

22 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`.

23 Philip N. Klein and Dániel Marx. Solving planar k-terminal cut in $O(n^{c\sqrt{k}})$ time. In *Proceedings of the 39th International Colloquium of Automata, Languages and Programming (ICALP)*, volume 7391 of *Lecture Notes in Comput. Sci.*, pages 569–580. Springer, 2012.

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

25 Nathan Linial and Michael E. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993. `doi:10.1007/BF01303516`.

26 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: `http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96`.

27 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 Siu-Wing Cheng and Olivier Devillers, editors, *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08-11, 2014*, page 67. ACM, 2014. `doi:10.1145/2582112.2582124`.

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

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

30   Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for Steiner problems on planar and bounded-genus graphs. In *FOCS 2014*, pages 276–285. IEEE Computer Society, 2014.

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