2016 IEEE 57th Annual Symposium on Foundations of Computer Science

Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering

Fedor V. Fomin*, Daniel Lokshtanov*, Dániel Marx†, Marcin Pilipczuk‡, Michał Pilipczuk‡, and Saket Saurabh*
*Department of Informatics, University of Bergen, Norway,
Emails: fomin|daniello|Saket.Saurabh@ii.uib.no.
†Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Hungary,
Email: dmarx@cs.bme.hu.
‡Institute of Informatics, University of Warsaw, Poland,
Emails: marcin.pilipczuk|michal.pilipczuk@mimuw.edu.pl.
§Institute of Mathematical Sciences, India,
Email: saket@imsc.res.in.

***Abstract*—We prove the following theorem. Given a planar graph G and an integer k, it is possible in polynomial time to randomly sample a subset A of vertices of G with the following properties:**

- **A induces a subgraph of G of treewidth $\mathcal{O}(\sqrt{k}\log k)$, and**
- **for every connected subgraph H of G on at most k vertices, the probability that A covers the whole vertex set of H is at least $(2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)})^{-1}$, where n is the number of vertices of G.**

Together with standard dynamic programming techniques for graphs of bounded treewidth, this result gives a versatile technique for obtaining (randomized) subexponential parameterized algorithms for problems on planar graphs, usually with running time bound $2^{\mathcal{O}(\sqrt{k}\log^2 k)}n^{\mathcal{O}(1)}$. The technique can be applied to problems expressible as searching for a small, connected pattern with a prescribed property in a large host graph; examples of such problems include DIRECTED k-PATH, WEIGHTED k-PATH, VERTEX COVER LOCAL SEARCH, and SUBGRAPH ISOMORPHISM, among others. Up to this point, it was open whether these problems can be solved in subexponential parameterized time on planar graphs, because they are not amenable to the classic technique of bidimensionality. Furthermore, all our results hold in fact on any class of graphs that exclude a fixed apex graph as a minor, in particular on graphs embeddable in any fixed surface.

***Keywords*-subexponential algorithms, planar graphs, apex-minor-free graphs, subgraph isomorphism**

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 267959 (F. V. Fomin), no. 280152 (D. Marx), and no. 306992 (S. Saurabh). The work of D. Marx is also supported by Hungarian Scientific Research Fund (OTKA) grant NK105645. The research of Ma. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/09/B/ST6/03136. The research of Mi. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. Mi. Pilipczuk is also supported by the Foundation for Polish Science (FNP) via the START stipend programme. Part of the research of F. Fomin, D. Marx, and Ma. Pilipczuk has been done when they were participating in the "Fine-grained complexity and algorithm design" program at the Simons Institute for Theory of Computing in Berkeley.

I. INTRODUCTION

Most of the natural NP-hard problems on graphs remain NP-hard even when the input graph is restricted to be planar. However, it was realized already in the dawn of algorithm design that the planarity of the input can be exploited algorithmically. Using the classic planar separator theorem of Lipton and Tarjan [1], one can design algorithms working in subexponential time, usually of the form $2^{\mathcal{O}(\sqrt{n})}$ or $2^{\mathcal{O}(\sqrt{n}\log n)}$, for a wide variety of problems that behave well with respect to separators; such running time cannot be achieved on general graph unless the Exponential Time Hypothesis (ETH) fails [2]. From the modern perspective, the planar separator theorem implies that a planar graph on n vertices has treewidth $\mathcal{O}(\sqrt{n})$, and the obtained tree decomposition can be used to run a Divide&Conquer algorithm or a dynamic programming subroutine. The idea of exploiting small separators plays a crucial role in modern algorithm design on planar graphs and related graph classes, including polynomial-time, approximation, and parameterized algorithmic paradigms.

Let us take a closer look at the area of parameterized complexity. While for most parameterized NP-hard problems the exponential dependence on the parameter is the best we can hope for, under ETH, there are plenty of problems admitting subexponential parameterized algorithms when restricted to planar graphs. This was first observed in 2000 by Alber et al. [3], who obtained a subexponential parameterized algorithm for deciding whether a given n-vertex planar graph contains a dominating set of size k in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. It turned out that the phenomenon is much more general. A robust framework explaining why various problems like FEEDBACK VERTEX SET, VERTEX COVER, DOMINATING SET or LONGEST PATH admit subexponential parameterized algorithms on planar graphs, usually with running times of the form $2^{\mathcal{O}(\sqrt{k})}\cdot n^{\mathcal{O}(1)}$ or $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$, is provided by the bidimensionality theory of Demaine et al. [4]. Roughly

0272-5428/16 $31.00 © 2016 IEEE
DOI 10.1109/FOCS.2016.62

speaking, the idea is to exploit the relation between the treewidth of a planar graph and the size of a grid minor that can be found in it. More precisely, the following win/win approach is implemented. Either the treewidth of the graph is $\mathcal{O}(\sqrt{k})$ and the problem can be solved using dynamic programming on a tree decomposition, or a $c\sqrt{k} \times c\sqrt{k}$ grid minor can be found, for some large constant c, which immediately implies that we are working with a yes- or with a no-instance of the problem. Furthermore, it turns out that for a large majority of problems, the running time yielded by bidimensionality is essentially optimal under ETH: no $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$-time algorithm can be expected. We refer to the survey [5], as well as the textbook [6, Chapter 7] for an overview of bidimensionality and its applications.

While the requirement that the problem can be solved efficiently on bounded treewidth graphs is usually not restrictive, the assumption that uncovering *any* large grid minor provides a meaningful insight into the instance considerably limits the applicability of the bidimensionality methodology. Therefore, while bidimensionality can be extended to more general classes, like H-minor-free graphs [4], [7], map graphs [8], or unit disk graphs [9], there are many problems that are "ε-close" to being bidimensional, and yet their parameterized complexity remained open for years.

One example where such a situation occurs is the DIRECTED LONGEST PATH problem. While the existence of a $\sqrt{k} \times \sqrt{k}$ grid minor in an undirected graph immediately implies the existence of an undirected path on k vertices, the same principle cannot be applied in the directed setting: even if we uncover a large grid minor in the underlying undirected graph, there is no guarantee that a long directed path can be found, because we do not control the orientation of the arcs. Thus, DIRECTED LONGEST PATH can be solved in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ on general graphs using color coding [10], but no substantially faster algorithms on planar graphs were known. On the other hand, no $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$-time algorithm on planar graphs can be expected under ETH, which leaves a large gap between the known upper and lower bounds. Closing this embarrassing gap was mentioned as an open problem in [11]–[14]. A similar situation happens for WEIGHTED LONGEST PATH, where we are looking for a k-path of minimum weight in an edge-weighted planar graph; the question about the complexity of this problem was raised in [11], [15]. Another example is k-CYCLE: deciding whether a given planar graph contains a cycle of length exactly k. While the property of admitting a cycle of length *at least* k is bidimensional, and therefore admits a $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$-time algorithm on planar graphs, the technique fails for the variant when we ask for length *exactly* k. This question was asked in [11]. We will mention more problems with this behavior later on.

The theme of "subexponential algorithms beyond bidimensionality" has recently been intensively investigated, with various success. For a number of specific problems such algorithms were found; these include planar variants of STEINER TREE parameterized by the size of the tree [16], [17], SUBSET TSP [18], or MAX LEAF OUTBRANCHING [12]. In all these cases, the algorithms were technically very involved and depended heavily on the combinatorics of the problem at hand. A more systematic approach is offered by the work of Dorn et al. [12] and Tazari [13], [14], who obtained "almost" subexponential algorithm for DIRECTED LONGEST PATH on planar, and more generally, apex-minor-free graphs. More precisely, they proved that for any $\varepsilon > 0$, there is δ such that the DIRECTED LONGEST PATH problem is solvable in time $\mathcal{O}((1+\varepsilon)^k \cdot n^\delta)$ on planar directed graphs, and more generally, on directed graphs whose underlying undirected graph excludes a fixed apex graph as a minor. This technique can be extended to other problems that can be characterized as searching for a connected pattern in a large host graph, which suggests that some more robust methodology is hiding just beyond the frontier of our understanding.

Main result. In this paper, we introduce a versatile technique for solving such problems in subexponential parameterized time, by proving the following theorem.

Theorem 1. *Let $\mathcal{C}$ be a class of graphs that exclude a fixed apex graph as a minor. Then there exists a randomized polynomial-time algorithm that, given an n-vertex graph G from $\mathcal{C}$ and an integer k, samples a vertex subset $A \subseteq V(G)$ with the following properties:*

- *(P1) The induced subgraph $G[A]$ has treewidth $\mathcal{O}(\sqrt{k}\log k)$.*
- *(P2) For every vertex subset $X \subseteq V(G)$ with $|X| \leq k$ that induces a connected subgraph of G, the probability that X is covered by A, that is $X \subseteq A$, is at least $(2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)})^{-1}$.*

Here, by an apex graph we mean a graph that can be made planar by removing one vertex. Note that Theorem 1 in particular applies to planar graphs, and to graphs embeddable in a fixed surface.

Applications. Similarly as in the case of bidimensionality, Theorem 1 provides a simple recipe for obtaining subexponential parameterized algorithms: check how fast the considered problem can be solved on graphs of bounded treewidth, and then combine the treewidth-based algorithm with Theorem 1. We now show how Theorem 1 can be used to obtain randomized subexponential parameterized algorithms for a variety of problems on apex-minor-free classes; for these problems, the existence of such algorithms so far was open even for planar graphs. We only list the most interesting examples to showcase possible applications.

Directed and weighted paths and cycles. As mentioned earlier, the question about the existence of subexponential parameterized algorithms for DIRECTED LONGEST PATH

and WEIGHTED LONGEST PATH on planar graphs was asked in [11]–[14]. Let us observe that on a graph of treewidth t, both DIRECTED LONGEST PATH and WEIGHTED LONGEST PATH, as well as their different combinations, like finding a maximum or minimum weight directed path or cycle on k vertices, are solvable in time $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$ by the standard dynamic programming [6, Chapter 7], and single-exponential running time $2^{\mathcal{O}(t)} n^{\mathcal{O}(1)}$ is achievable [19]–[21].

In order to obtain a subexponential parameterized algorithm for, say, DIRECTED LONGEST PATH on planar directed graphs, we do the following. Let G be the given planar directed graph, and let $U(G)$ be its underlying undirected graph. Apply the algorithm of Theorem 1 to $U(G)$, which in polynomial time samples a subset $A \subseteq V(U(G))$ such that $G[A]$ has treewidth at most $\mathcal{O}(\sqrt{k} \log k)$, and the probability that A covers some directed k-path in G, provided it exists, is at least $(2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)})^{-1}$. Then, we verify whether $G[A]$ admits a directed k-path using standard dynamic programming in time $2^{\mathcal{O}(\sqrt{k} \log k)} \cdot n^{\mathcal{O}(1)}$. Provided some directed k-path exists in the graph, this algorithm will find one with probability at least $(2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)})^{-1}$. Thus, by making $2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)}$ independent runs of the algorithm, we can reduce the error probability to at most $1/2$. All in all, the obtained algorithm runs in time $2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)}$ and can only report false negatives with probability at most $1/2$

Note that in order to apply the dynamic programming algorithm, we need to construct a suitable tree decomposition of $G[A]$. However, a variety of standard algorithms, e.g., the classic 4-approximation of Robertson and Seymour [22], can be used to construct such an approximate tree decomposition within the same asymptotic running time. Actually, a closer look into the proof of Theorem 1 reveals that the algorithm can construct, within the same running time, a tree decomposition of $G[A]$ certifying the upper bound on its treewidth.

Observe that the same approach works also for any apex-minor-free class $\mathcal{C}$, and can be applied also to WEIGHTED LONGEST PATH and k-CYCLE. We obtain the following.

Corollary 2. *Let $\mathcal{C}$ be a class of graphs that exclude some fixed apex graph as a minor. Then problems:* WEIGHTED LONGEST PATH*,* k-CYCLE*, and* DIRECTED LONGEST PATH*, are all solvable in randomized time $2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)}$ on graphs from $\mathcal{C}$. In case of* DIRECTED LONGEST PATH*, we mean that the underlying undirected graph of the input graph belongs to $\mathcal{C}$.*

Note here that the approach presented above works in the same way for various combinations and extensions of problems in Corollary 2, like weighted, colored, or directed variants, possibly with some constraints on in- and out-degrees, etc. In essence, the only properties that we need is that the sought pattern persists in the subgraph induced by the covering set A, and that it can be efficiently found using dynamic programming on a tree decomposition. To give one more concrete example, Sau and Thilikos in [15] studied the problem of finding a connected k-edge subgraph with all vertices of degree at most some integer Δ; for $\Delta = 2$ this corresponds to finding a k-path or a k-cycle. For fixed Δ they gave a subexponential algorithm on (unweighted) graphs excluding some fixed graph as a minor, and asked if the weighted version of this problem can be solved in subexponential parameterized time. Theorem 1 immediately implies that for fixed Δ, the weighted variant of the problem is solvable in randomized time $2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)}$ on apex-minor-free graphs.

Subgraph Isomorphism. SUBGRAPH ISOMORPHISM is a fundamental problem, where we are given two graphs: an n-vertex *host* graph G and a k-vertex *pattern* graph P. The task is to decide whether P is isomorphic to a subgraph of G. Eppstein [23] gave an algorithm solving SUBGRAPH ISOMORPHISM on planar graphs in time $k^{\mathcal{O}(k)} n$, which was subsequently improved by Dorn [24] to $2^{\mathcal{O}(k)} n$. The first implication of our main result for SUBGRAPH ISOMORPHISM concerns the case when the maximum degree of P is bounded by a constant. Matoušek and Thomas [25] proved that if a tree decomposition of the host graph G of width t is given, and the pattern graph P is connected and of maximum degree at most some constant Δ, then deciding whether P is isomorphic to a subgraph of G can be done in time $\mathcal{O}(k^{t+1} n)$. By combining this with Theorem 1 as before, we obtain the following.

Corollary 3. *Let $\mathcal{C}$ be a class of graphs that exclude some fixed apex graph as a minor, and let Δ be a fixed constant. Then, given a connected graph P with at most k vertices and maximum degree not exceeding Δ, and a graph $G \in \mathcal{C}$ on n vertices, it is possible to decide whether P is isomorphic to a subgraph of G in randomized time $2^{\mathcal{O}(\sqrt{k} \log^2 k)} \cdot n^{\mathcal{O}(1)}$.*

In a very recent work, Bodlaender et al. [26] proved that SUBGRAPH ISOMORPHISM on planar graphs cannot be solved in time $2^{o(n/\log n)}$ unless ETH fails. The lower bound of Bodlaender et al. holds for two very special cases. The first case is when the pattern graph P is a tree and has only one vertex of super-constant degree. Then the second case is when P is not connected, but its maximum degree is a constant. Thus, the results of Bodlaender et al. show that both the connectivity and the bounded degree constraints on pattern P in Corollary 3 are necessary to keep the square root dependence on k in the exponent. However, a possibility of solving SUBGRAPH ISOMORPHISM in time $2^{\mathcal{O}(k/\log k)} \cdot n^{\mathcal{O}(1)}$, which is still parameterized subexponential, is not ruled out by the work of Bodlaender et al. Interestingly enough, Bodlaender et al. [26], also give a matching dynamic programming algorithm that can be combined with our theorem.

Theorem 4 ([26])**.** *Let H be a fixed graph, and let us fix*

any $\varepsilon > 0$. Given a pattern graph P on at most k vertices, and an H-minor-free host graph G of treewidth at most $\mathcal{O}(k^{1-\varepsilon})$, it is possible to decide whether P is isomorphic to a subgraph of G in time $2^{\mathcal{O}(k/\log k)} \cdot n^{\mathcal{O}(1)}$.

We remark that Bodlaender et al. [26] claim only a subexponential exact algorithm with running time $2^{\mathcal{O}(n/\log n)}$, but the dynamic programming subroutine underlying this result actually gives the stronger algorithmic result as stated above.

By combining Theorem 4 with Theorem 1 in the same way as before, we obtain the following.

Corollary 5. *Let $\mathcal{C}$ be a class of graphs that exclude some fixed apex graph as a minor. Then, given a connected graph P with at most k vertices, and a graph $G \in \mathcal{C}$ on n vertices, it is possible to decide whether P is isomorphic to a subgraph of G in randomized time $2^{\mathcal{O}(k/\log k)} \cdot n^{\mathcal{O}(1)}$.*

Let us stress here that the lower bounds of Bodlaender et al. [26] show that the running time given by Corollary 5 is tight: no $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm can be expected under ETH.

Local search. Fellows et al. [27] studied the following parameterized local search problem on apex-minor-free graphs. In the LS VERTEX COVER problem we are given an n-vertex graph G, a vertex cover S in G, and an integer k. The task is to decide whether G contains a vertex cover S', such that $|S'| < |S|$ and the Hamming distance $|S \triangle S'|$ between sets S and S' is at most k. In other words, for a given vertex cover, we ask if there is a smaller vertex cover which is k-close to the given one, in terms of Hamming (edit) distance. Fellows et al. [27] gave an algorithm solving LS VERTEX COVER in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ on planar graphs. The question whether this can be improved to subexponential parameterized time was raised in [27], [28].

The crux of the approach of Fellows et al. [27] is the following observation. If there is solution to LS VERTEX COVER, then there is a solution S', such that $S \triangle S'$ induces a connected subgraph in G. Since $S \triangle S'$ contains at most k vertices and is connected, our Theorem 1 can be used to sample a vertex subset A that induces a subgraph of treewidth $\mathcal{O}(\sqrt{k}\log k)$ and covers $S \triangle S'$ with high probability. Thus, by applying the same principle of independent repetition of the algorithm, we basically have search for suitable sets $S \setminus S'$ and $S' \setminus S$ in the subgraph of G induced by A. We should be, however, careful here: there can be edges between A and its complement, and these edges also need to be covered by S', so we cannot just restrict our attention to $G[A]$. To handle this, we apply the following preprocessing. For every vertex $v \in A$, if v is adjacent to some vertex outside of A that is not included S, then v must be in S and needs also to remain in S'. Hence, we delete all such vertices from $G[A]$, and it is easy to see that now the problem boils down to looking for feasible $S \setminus S'$ and $S' \setminus S$ within the obtained induced subgraph. This can be easily done in time $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$, where $t \leq \mathcal{O}(\sqrt{k}\log k)$ is the treewidth of this subgraph; hence we obtain the following:

Corollary 6. *Let $\mathcal{C}$ be a class of graphs that exclude some fixed apex graph as a minor. Then* LS VERTEX COVER *on graphs from $\mathcal{C}$ can be solved in randomized time $2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)}$.*

Steiner tree. STEINER TREE is a fundamental network design problem: for a graph G with a prescribed set of terminal vertices S, and an integer k, we ask whether there is a tree on at most k edges that spans all terminal vertices. Pilipczuk et al. [16] gave an algorithm for this problem with running time $2^{\mathcal{O}((k\log k)^{2/3})} \cdot n$ on planar graphs and on graphs of bounded genus. With much more additional work, the running time was improved to $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n$ in [17].

Again, by combining the standard dynamic programming solving STEINER TREE on graphs of treewidth t in time $2^{\mathcal{O}(t\log t)} n^{\mathcal{O}(1)}$, see e.g. [29], with Theorem 1, we immediately obtain the following.

Corollary 7. *Let $\mathcal{C}$ be a class of graphs that exclude some fixed apex graph as a minor. Then* STEINER TREE *on graphs from $\mathcal{C}$ can be solved in randomized time $2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)}$.*

Contrary to the much more involved algorithm of Pilipczuk et al. [17], the algorithm above can equally easily handle various variants of the problem. For instance, we can look for a Steiner tree on k edges that minimizes the total weight of the edges, or we can ask for a Steiner outbranching in a directed graph, or we can put additional constraints on vertex degrees in the tree, and so on.

Outline. This extended abstract contains only an overview of the proof of Theorem 1. A full version of the paper is available on arXiv [30].

II. OVERVIEW OF THE PROOF OF THEOREM 1

We now give an informal overview of the proof of Theorem 1 in the case of planar graphs. In fact, the only two properties of planar graphs which are essential to the proof are (a) planar graphs are minor-closed, and (b) they have locally bounded treewidth by a linear function, that is, there exists a constant c_{tw} such that every planar graph of radius k has treewidth at most $c_{\mathrm{tw}} \cdot k$. In fact, for planar graphs one can take $c_{\mathrm{tw}} = 3$ [31], and as shown in [32], the graph classes satisfying both (a) and (b) are exactly graph classes excluding a fixed apex graph as a minor. However, in a planar graph we can rely on some topological intuition, making the presentation more intuitive. We assume familiarity with tree decompositions.

Locally bounded treewidth of planar graphs. As a warm-up, let us revisit a proof that planar graphs have locally bounded treewidth. The considered proof yields a worse constant than $c_{\mathrm{tw}} = 3$, but one of the main ideas — to

find a separator in a planar graph by finding many disjoint paths, present already in [33] — it is insightful for our argumentation. Let G_0 be a graph of radius k, that is, there exists a root vertex r_0 such that every vertex of G_0 is within distance at most k from r_0.

As with most proofs showing that a graph in question has bounded treewidth, we will recursively construct a tree decomposition of bounded width. To this end, we need to carefully define the state of the recursion. We do it as follows: the recursive step aims at decomposing a subgraph G of the input graph G_0, with some chosen set of terminals $T \subseteq V(G)$ on the outer face of G. The terminals T represent connections between G and the rest of G_0. In order to be able to glue back the obtained decompositions from the recursive step, our goal is to provide a tree decomposition of G with T contained in the root bag of the decomposition, so that later we can connect this bag to decompositions of other pieces of the graph that also contain the vertices of T. During the process, we keep the invariant that $|T| \leq 8(k+2)$, allowing us to bound the width of the decomposition. Furthermore, the assumption that G_0 is of bounded radius projects onto the recursive subinstances by the following invariant: every vertex of G is within distance at most k from some terminal.

In the recursive step, if $T = V(G)$, $|T| < 8(k+2)$, or G is not connected, then we can perform some simple steps that we do not discuss here. The interesting case is when $|T| = 8(k+2)$.

We partition T along the outer face into four parts of size $2(k+2)$ each, called north, east, south, and west terminals. We compute minimum vertex cuts between the north and the south terminals, and between the east and the west terminals. If, in any of these directions, a cut W of size strictly smaller than $2(k+2)$ is found, then we can make a divide-and-conquer step: for every connected component D of $G-(T \cup W)$ we recurse on the graph $G[N[D]]$ with terminals $N(D)$, obtaining a tree decomposition $\mathcal{T}_D$. Finally, we attach all the obtained tree decompositions below a fresh root node with bag $T \cup W$, which is of size less than $10(k+2)$.

The crux is that such a separator W is always present in the graph. Indeed, otherwise there would exist $2(k+2)$ disjoint paths between the north and the south terminals and $2(k+2)$ disjoint paths between the east and the west terminals. Consider the region bounded by the two middle north-south and the two middle east-west paths: the vertices contained in this region are within distance larger than k from the outer face, on which all terminals lie (see Fig. 1a). This contradicts our invariant.

Our recursion. In our case, we use a similar, but much more involved recursion scheme. In the recursive step, we are given a *minor* G of the input graph G_0, with some *light* terminals $T^{\mathsf{li}} \subseteq V(G)$ on the outer face, and some *heavy* terminals $T^{\mathsf{he}} \subseteq V(G) \setminus T^{\mathsf{li}}$ lying anywhere in the graph. As before, the terminals represent connections to the other parts of the graph. We require that the terminals $T := T^{\mathsf{li}} \cup T^{\mathsf{he}}$ need to be contained in the root bag of the tree decomposition that is going to be constructed in this recursion step. Moreover, we maintain an invariant that $|T| = \widetilde{\mathcal{O}}(\sqrt{k})$, in order to bound the width of the decomposition. The graph G is a minor of the input graph G_0, since we often prefer to contract some edges instead of deleting them; thus we maintain some distance properties of G.

In our recursion, the light terminals originate from cutting the graph in a similar fashion as in the proof that planar graphs have locally bounded treewidth, presented above. Hence, we keep the invariant that light terminals lie on the outer face. We sometimes need to cut deeply inside G. The produced terminals are heavy, but every such step corresponds to a significant progress in detecting the pattern, and hence such steps will be rare. In every such step, we artificially provide connectivity of the subinstances through the heavy terminals; this is technical and omitted in this overview.

Recall that our goal is to preserve a connected k-vertex pattern from the input graph. Here, the pattern can become disconnected by recursing on subsequent separations, but such cuttings will always be along light terminals. Therefore, we define a *pattern* in a subinstance $(G, T^{\mathsf{li}}, T^{\mathsf{he}})$ solved in the recursion as a set $X \subseteq V(G)$ of size at most k such that every connected component of $G[X]$ contains a light terminal. Hence, compared to the presented proof of planarity implying locally bounded treewidth, we aim at more restricted width of the decomposition, namely $\widetilde{\mathcal{O}}(\sqrt{k})$, but we can contract or delete parts of G, as long as the probability of spoiling a fixed, but unknown k-vertex pattern X remains inverse subexponential in k.

Clustering. Upon deleting a vertex or an edge, some distance properties that we rely on can be broken. We need the following sampling procedure that partitions the graph into connected components of bounded radii, such that the probability of spoiling a particular pattern is small. The proof of the following theorem is similar to the metric decomposition tool of [34] and to the recursive decomposition used in the construction of Bartal's HSTs [35].

Theorem 8. *There exists a randomized polynomial-time algorithm that, given a graph G on $n > 1$ vertices and a positive integer k, returns a vertex subset $B \subseteq V(G)$ with the following properties:*

(a) The radius of each connected component of $G[B]$ is less than $9k^2 \log n$.

(b) For each vertex subset $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq B$ is at least $1 - \frac{1}{k}$.

Standard divide&conquer step. We would like to apply a similar divide&conquer step as in the presented proof that

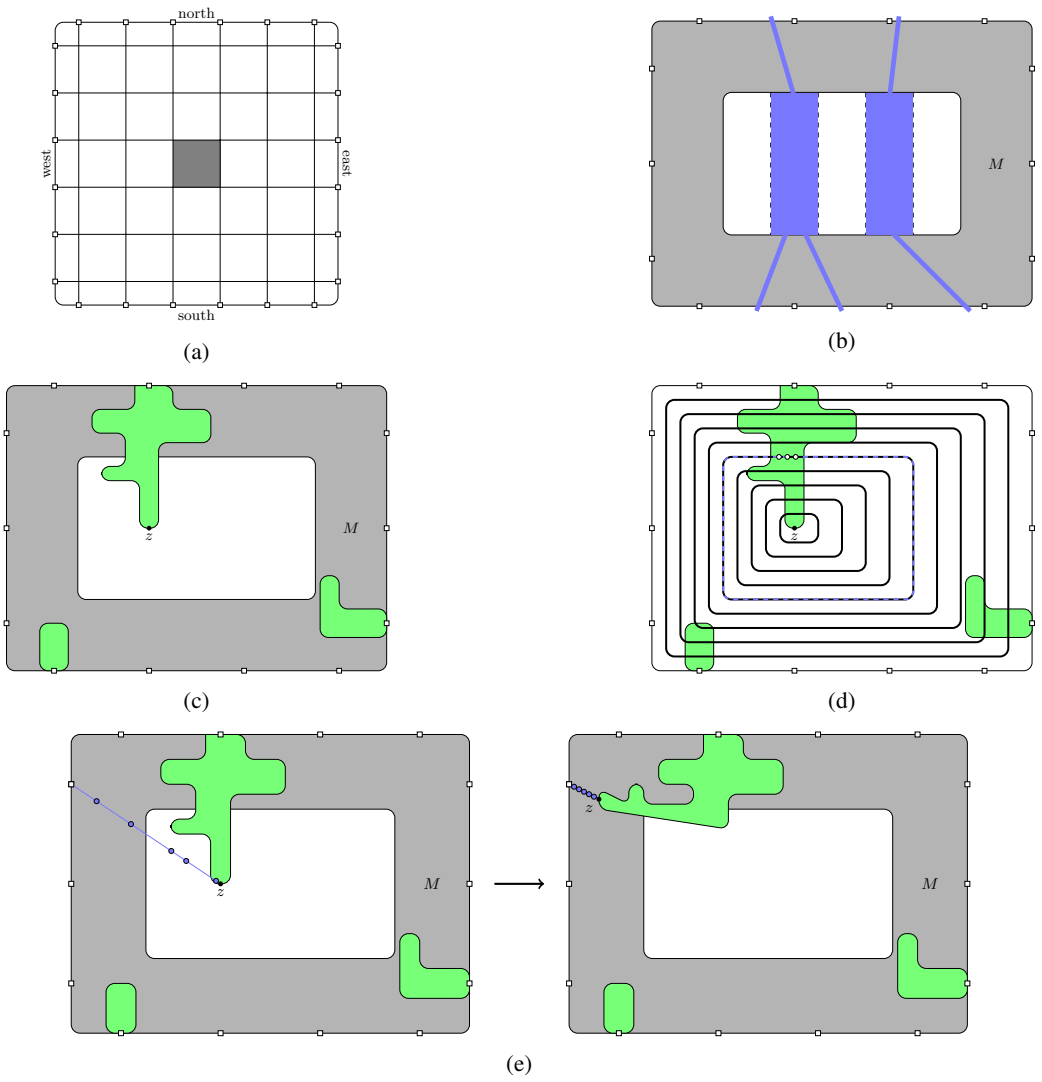

Figure 1: Illustrations for Section II. (a) The proof that planarity implies locally bounded treewidth. The vertices in the gray area are too far from the terminals. (b) Standard partitioning step. The margin is gray and the islands are separated by dashed lines. The blue separator consists of $\widetilde{\mathcal{O}}(\sqrt{k})$ islands and vertices of the margin. If the blue islands are disjoint from the solution, we delete them and obtain a balanced separator of size $\widetilde{\mathcal{O}}(\sqrt{k})$. (c) The situation if the standard partitioning step cannot be applied: we have a component of the pattern (green) stretched between a light terminal and a vertex z inside the margin. (d) Chain of z-T^{li} separators: a sparse separator that partitions the pattern in a balanced fashion is highlighted. (e) Contraction of a path P_i (blue) onto its public vertices (blue circles). A significant number of the vertices of the pattern become much closer to the light terminals.

planar graphs have locally bounded treewidth. The problem is that we can only afford a separator W of size $\widetilde{\mathcal{O}}(\sqrt{k})$, however the radius of the graph can be much larger.

Let us define the *margin* M to be the set of vertices within distance at most $2000\sqrt{k}\lg k = \widetilde{\mathcal{O}}(\sqrt{k})$ from any light terminal. Intuitively, our case should be easy if every vertex of the pattern X is in the margin: we could then just throw away all vertices of $G - M$ and use locally bounded treewidth, as the light terminals lie on the outer face. However, we cannot just branch (guess) whether this is the case: the information that $G - M$ contains a vertex of the pattern is not directly useful.

Instead, we make a localized analog of this guess: we identify a relatively compact set of vertices of $G - M$ that prohibit us from making a single step of the recursion sketched above. First, we apply the clustering procedure (Theorem 8) to the graph $G - M$, so that we can assume that every connected component of $G - M$, henceforth called *an island*, is of radius bounded polynomially in k and $\lg n$. Second, we construct an auxiliary graph H by

contracting every island C into a single vertex u_C. Note that now in H every vertex is within distance at most $2000\sqrt{k}\lg k + 1 = \widetilde{\mathcal{O}}(\sqrt{k})$ from a light terminal. Thus H has treewidth $\widetilde{\mathcal{O}}(\sqrt{k})$. By standard arguments, we can find a balanced separator W_H in H, that is, a separator of size $\widetilde{\mathcal{O}}(\sqrt{k})$ such that every connected component of $H - W_H$, after lifting it back to G by reversing contractions, contains (a) at most $|T|/2$ terminals from G, and (b) at most $|V(G) \setminus T|/2$ non-terminal vertices of G.

The separator W_H can be similarly lifted to a separator W in G that corresponds to $\widetilde{\mathcal{O}}(\sqrt{k})$ vertices of M and $\widetilde{\mathcal{O}}(\sqrt{k})$ islands. Now it is useful to make a guess if some vertex of an island in W (i.e., a vertex of $W \setminus M$) belongs to the solution: if this is not the case, we can delete the whole $W \setminus M$ from the graph, and apply the procedure recursively to connected components of $G - W$; if this is the case, the area to search for such a vertex of the pattern is limited to $\widetilde{\mathcal{O}}(\sqrt{k})$ components of radius polynomial in k and $\lg n$. Therefore, with some probability q we decide to assume that $W \setminus M$ contains a vertex of the pattern, and with the remaining probability $1-q$ we decide that this is not the case. In the latter case, we remove $W \setminus M$ from the graph, and recurse using $W \cap M$, which has size $\widetilde{\mathcal{O}}(\sqrt{k})$, as a separator; see Fig. 1b. The fact that every connected component of $G - W$ contains at most $|T|/2$ terminals allows us to keep the invariant that $|T| = \widetilde{\mathcal{O}}(\sqrt{k})$.

Let us now analyze what probability q we can afford. Observe that in every subinstance solved recursively, the number of nonterminal vertices is halved. Thus, every vertex x of the pattern X is contained in G only in $\mathcal{O}(\lg n)$ subinstances in the whole recursion tree; here we exclude the subinstances where x is a light terminal, because then its treatment is determined by the output specification of the recursive procedure. Consequently, we care about correct choices only in $\mathcal{O}(k \lg n)$ steps of the recursion. In these steps, we want not to make a mistake during the clustering procedure ($1/k$ failure probability) and to correctly guess that $W \setminus M$ is disjoint with the pattern, provided this is actually the case (q failure probability). Thus, if we put $q = 1/k$, then the probability that we succeed in all $\mathcal{O}(k \lg n)$ steps we care about is inverse-polynomial in n; this is sufficient for our needs.

Island with a vertex of the pattern. We are left with the second case, where some island $C \subseteq W$ intersects the pattern. We have $q = 1/k$ probability of guessing correctly that this is the case, and independently we have $(1 - 1/k)$ probability of not making a mistake in the clustering step.

The bound on the radii of the islands, as well as the fact that only $\widetilde{\mathcal{O}}(\sqrt{k})$ islands are contained in W, allow us to localize this vertex of the pattern even closer. Recall that the radius of each island is bounded by $9k^2 \lg n$. For the rest of this overview we assume that $\lg n$ is bounded polynomially in k, and hence the radius of each island is polynomial in k. Intuitively, this is because if, say, we had $\lg n > 100 \cdot k^{100}$, then $n > 2^{100k^{100}}$ and with $2^{100k^{100}}$ allowed factor in the running time bound we can apply a variety of other algorithmic techniques. More formally, we observe that $(\lg n)^{\widetilde{\mathcal{O}}(\sqrt{k})}$ is bounded by $2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot n^{o(1)}$, which is sufficient to make sure that all the experiments whose success probability depend on $\lg n$ succeed simultaneously with probability at least $(2^{\widetilde{\mathcal{O}}(\sqrt{k})} \cdot n^{o(1)})^{-1}$.

We first guess (by sampling at random) an island $C \subseteq W$ that contains a vertex of the pattern. Then, we pick an arbitrary vertex $z \in C$, and guess (by sampling at random) the distance d in C between z and the closest vertex of the pattern in C. By contracting all vertices within distance less than d from z, with success probability inverse-polynomial in k, we arrive at the following situation: (see Fig. 1c)

> we have a vertex $z \notin M$ such that either z or a neighbor of z belongs to the pattern X.

Chain of separators. Hence, one of the components of $G[X]$ is stretched across the margin M, between a light terminal on the outer face and the vertex z inside the margin. Our idea now is to use this information to cut X in a balanced fashion. Note that we have already introduced an inverse-polynomial in k multiplicative factor in the success probability. Hence, to maintain the overall inverse-subexponential dependency on k in the success probability, we should aim at a progress that will allow us to bound the number of such steps by $\widetilde{\mathcal{O}}(\sqrt{k})$.

Unfortunately, it is not obvious how to find such a separation. It is naive to hope for a z-T^{li} separator of size $\widetilde{\mathcal{O}}(\sqrt{k})$, and a larger separator seems useless, if there is only one. However, we can aim at a Baker-style argument: if we find a chain of p pairwise disjoint z-T^{li} separators $C_1, C_2, \ldots, C_p$ (see Fig. 1d), we could guess a "sparse one", and separate along it. Since the separators are pairwise disjoint, there exists a "sparse" separator C_i with at most k/p vertices of the pattern. On the other hand, since the pattern contains a component stretched from z to a light terminal, every C_i intersects the pattern. If we omit the first and the last $p/4$ separators, and look for a sparse separator in between with at most $2k/p$ by means of guessing only $2k/p$ vertices of X. If all separators are of size bounded polynomially in k, the optimal choice is $p \sim k^{2/3}$, which leads to success probability inverse in $2^{\widetilde{\mathcal{O}}(k^{2/3})}$.

However, we can apply a bit smarter counting argument. Take $p = 120\sqrt{k} \lg k$. Look at $C_{p/2}$ and assume that at most half of the vertices of X lie on the side of $C_{p/2}$ with separators C_i for $i < p/2$; the other case is symmetric. The crucial observation is the following: there exists an index $i \leq p/2$ such that if $|C_i \cap X| = \alpha$ then C_i partitions X into two parts of size at least $\alpha\sqrt{k}/10$ each. Indeed, otherwise we have that for every $i \leq p/2$ it holds that

$$|X \cap C_i| \geq \frac{10}{\sqrt{k}} \cdot \sum_{j<i} |X \cap C_j|.$$

This implies $|X \cap \bigcup_{j \leq i} C_j| \geq (1 + 10/\sqrt{k})^i$, and $|X \cap \bigcup_{j \leq p/2} C_j| > k$ for $p = 120\sqrt{k}\lg k$.

Hence, we guess (by sampling at random) such an index i, the value of $\alpha = |X \cap C_i|$, and the set $X \cap C_i$. If the size of C_i is bounded polynomially in k, with success probability $k^{-\mathcal{O}(\alpha)}$ we partition the pattern into two parts of size at least $\alpha\sqrt{k}/10$ each. A simple amortization argument shows that all these guessings incur only the promised $2^{-\widetilde{\mathcal{O}}(\sqrt{k})}$ multiplicative factor in the overall success probability. Furthermore, as such a step creates α heavy terminals, it can be easily seen that the total number of heavy terminals will never grow beyond $\widetilde{\mathcal{O}}(\sqrt{k})$.

However, the above argumentation assumes we are given such a chain of separators C_i: they are not only pairwise disjoint, but also of size polynomial in k. Let us now inspect how to find them.

Duality. In the warm-up proof of planar graphs having locally bounded treewidth, the separator W is obtained from the classic Menger's maximum flow/minimum cut duality. Here, we aim at a chain of separators, but we require that their sizes are polynomial in k. It turns out that we can find such a chain by formulating a maximum flow of minimum cost problem, and extracting the separator chain in question from the optimum solution to its (LP) dual.

Theorem 9. *There is a polynomial-time algorithm that given a connected graph G, a pair $s, t \in V(G)$ of different vertices, and positive integers p, q, outputs one of the following structures in G:*

(a) A chain $(C_1, \ldots, C_p)$ of (s,t)-separators with $|C_j| \leq 2q$ for each $j \in [p]$.

(b) A sequence $(P_1, \ldots, P_q)$ of (s,t)-paths with $|(V(P_i) \cap \bigcup_{i' \neq i} V(P_{i'})) \setminus \{s,t\}| \leq 4p$ for each $i \in [q]$.

Since all light terminals lie on the outer face, we can attach an auxiliary root vertex r_0 adjacent to all light terminals, and apply Theorem 9 to $(s,t) = (r_0, z)$, $p = 120\sqrt{k}\lg k$ and $q = \text{poly}(k)$. If the algorithm of Theorem 9 returns a chain of separators, we proceed as described before. Thus, we are left with the second output: $q = \text{poly}(k)$ nearly-disjoint paths from T^{li} to z.

Nearly-disjoint paths. The vertex set of every path P_i can be partitioned into *public* vertices $\mathsf{Pub}(P_i)$, the ones used also by other paths, and the remaining *private* vertices $\mathsf{Prv}(P_i)$. We have $|\mathsf{Pub}(P_i)| \leq 4p = 480\sqrt{k}\lg k$, and the sets $\mathsf{Prv}(P_i)$ are pairwise disjoint. We can assume $q > k$, so there exists a path P_i such that $\mathsf{Prv}(P_i)$ is disjoint with the pattern X. By incurring an additional $1/k$ multiplicative factor in the success probability, we can guess, by sampling at random, such index i.

How can we use such a path P_i? Clearly, we can delete the private vertices of P_i, because they can be assumed not to be used by the patter X. However, we choose a different way: we contract them onto neighboring public vertices along P_i, reducing P_i to a path with vertex set $\mathsf{Pub}(P_i)$. Observe that by this operation the vertex z changes its location in G: from a vertex deeply inside G, namely not within the margin M, it is moved to a place within distance $|\mathsf{Pub}(P_i)| \leq 480\sqrt{k}\lg k$ from the light terminals, which is less than a quarter of the width of the margin; see Fig. 1e.

Furthermore, by the connectivity assumptions on the pattern X, the vertex z drags along a number of vertices of X that are close to it. More precisely, if Q is a path in $G[X]$ connecting z or a neighbor of z with a light terminal, then the first $500\sqrt{k}\lg k$ vertices on Q are moved from being within distance at least $1500\sqrt{k}\lg k$ from all light terminals, to being within distance at most $1000\sqrt{k}\lg k$ from some light terminal. Hence, if we define that a vertex $x \in X$ is *far* if it is within distance larger than $1000\sqrt{k}\lg k$ (i.e., half of the width of the margin) from all light terminals, and *close* otherwise, then by contracting the private vertices of P_i as described above, at least $500\sqrt{k}\lg k$ vertices of k change their status from far to close.

By a careful implementation of all separation steps, we can ensure that no close vertex of X becomes far again. Consequently, we ensure that the above step can happen only $\widetilde{\mathcal{O}}(\sqrt{k})$ times. Since the probability of succeeding in all guessings within this step is inverse-polynomial in k, this incurs only a $2^{-\widetilde{\mathcal{O}}(\sqrt{k})}$ multiplicative factor in the overall success probability.

This finishes the overview of the proof of Theorem 1.

III. Conclusions

In this work we have laid foundations for a new tool for obtaining subexponential parameterized algorithms for problems on planar graphs, and more generally on graphs that exclude a fixed apex graph as a minor. The technique is applicable to problems that can be expressed as searching for a small, connected pattern in a large host graph. Using the new approach, we designed, in a generic manner, a number of subexponential parameterized algorithms for problems for which the existence of such algorithms was open. We believe, however, that this work provides only the basics of a new methodology for the design of parameterized algorithms on planar and apex-minor-free graphs. This methodology goes beyond the paradigm of bidimensionality and is yet to be developed.

An immediate question raised by our work is whether the technique can be derandomized. Note that Theorem 1 immediately yields the following combinatorial statement.

Theorem 10. *Let $\mathcal{C}$ be a class of graphs that exclude a fixed apex graph as a minor. Suppose G is an n-vertex graph from $\mathcal{C}$ and k is a positive integer. Then there exists a family $\mathcal{F}$ of subsets of vertices G, with the following properties satisfied:*

(P1) For each $A \in \mathcal{F}$, the treewidth of $G[A]$ is at most $\mathcal{O}(\sqrt{k}\log k)$.

(P2) For each vertex subset $X \subseteq V(G)$ such that $G[X]$ is connected and $|X| \leq k$, there exists some $A \in \mathcal{F}$ for which $X \subseteq A$.

(P3) It holds that $|\mathcal{F}| \leq 2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)}$.

The proof of Theorem 10 gives only a randomized algorithm constructing a family $\mathcal{F}$ that indeed covers all small patterns with high probability. We conjecture that the algorithm can be derandomized; that is, that the family whose existence is asserted by Theorem 10 can be computed in time $2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)}$. So far we are able to derandomize most of the components of the algorithm, primarily using standard constructions based on splitters and perfect hash families [36]. One part of the reasoning with which we still struggle is the clustering step (Theorem 8). Our optimism with derandomizing this last part stems from its resemblance to the construction of HSTs of [35], which have been subsequently derandomized [37].

Q1. Is it possible to construct a family with properties described in Theorem 10 in deterministic time $2^{\mathcal{O}(\sqrt{k}\log^c k)} \cdot n^{\mathcal{O}(1)}$, for some constant c?

In full version [30] we attempt to generalize our technique to the case when the pattern is disconnected, and when the class only excludes some fixed (but arbitrary) graph H as a minor. In the case of disconnected patterns, we were able to prove a suitable generalization of Theorem 1, however the success probability of the algorithm depends inverse-exponentially on number of connected components of the pattern. In the case of general H-minor-free classes, we needed to assume that the pattern admits a spanning tree of constant maximum degree. So far we do not see any reason for any of these constraints to be necessary.

Q2. Is it possible to prove Theorem 1 without the assumption that the subgraph induced by X has to be connected?

Q3. Is it possible to prove Theorem 1 only under the assumption that all graphs from $\mathcal{C}$ exclude some fixed (but arbitrary) graph H as a minor?

Our next question concerns local search problems in the spirit of the LS VERTEX COVER problem considered in Section I. Apart from this problem, Fellows et al. [27] designed FPT algorithms also for the local search for a number of other problems on apex-minor-free classes, including LS DOMINATING SET and its distance-d generalization. Here, we are given a dominating set S in a graph G from some apex-minor-free class $\mathcal{C}$, and we ask whether there exists a strictly smaller dominating set S' that is at Hamming distance at most k from S. Again, the approach of Fellows et al. [27] is based on the observation that if there is some solution, then there is also a solution S' such that $S \triangle S'$ can be connected using at most k additional vertices. Thus, we need to search for a connected pattern of size $2k$, instead of k, in which suitable sets $S \setminus S'$ and $S' \setminus S$ are to be found. Unfortunately, now the preprocessing step fails: vertices outside A may require to be dominated from within A, which poses additional constraints that are not visible in the graph $G[A]$ only. Hence, we cannot just focus on the graph $G[A]$. Observe, however, that the whole reasoning would go through if A covered not just $S \triangle S'$, but also its neighborhood. More generally, if the considered problem concerns domination at distance d, then we should cover the distance-d neighborhood of $S \triangle S'$. This motivates the following question.

Q4. For a fixed $d > 0$, is it possible to prove a stronger version of Theorem 1, where the sampled set A is required to cover the whole distance-d neighborhood of the set X with the same asymptotic lower bound on the success probability?

Finally, so far we do not know whether the connectivity condition in Theorem 5 is necessary.

Q5. Is it possible to solve SUBGRAPH ISOMORPHISM on planar graphs in time $2^{\mathcal{O}(k/\log k)} \cdot n^{\mathcal{O}(1)}$ without the assumption that the pattern graph is connected?

Note that a positive answer to Q2 implies a positive answer here as well, as the algorithm of Theorem 4 does not require the pattern graph to be connected.

ACKNOWLEDGEMENTS

The authors thank Felix Reidl, Fernando Sánchez Villaamil, Somnath Sikdar, and Yngve Villanger for some preliminary discussions on the topic of this paper, as well as an anynomous reviewer for pointing out the similarities between our clustering procedure and tools from the theory of metric embeddings.

REFERENCES

[1] R. J. Lipton and R. E. Tarjan, "Applications of a planar separator theorem," *SIAM J. Computing*, vol. 9, pp. 615–627, 1980.

[2] R. Impagliazzo and R. Paturi, "On the complexity of k-SAT," *J. Comput. Syst. Sci.*, vol. 62, no. 2, pp. 367–375, 2001.

[3] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier, "Fixed parameter algorithms for dominating set and related problems on planar graphs," *Algorithmica*, vol. 33, no. 4, pp. 461–493, 2002.

[4] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos, "Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs," *J. ACM*, vol. 52, no. 6, pp. 866–893, 2005.

[5] E. D. Demaine and M. Hajiaghayi, "The bidimensionality theory and its algorithmic applications," *Comput. J.*, vol. 51, no. 3, pp. 292–302, 2008.

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

[7] E. D. Demaine and M. Hajiaghayi, "Linearity of grid minors in treewidth with applications through bidimensionality," *Combinatorica*, vol. 28, no. 1, pp. 19–36, 2008.

[8] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos, "Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs," *ACM Trans. Alg.*, vol. 1, no. 1, pp. 33–47, 2005.

[9] F. V. Fomin, D. Lokshtanov, and S. Saurabh, "Bidimensionality and geometric graphs," in *SODA*. SIAM, 2012, pp. 1563–1575.

[10] N. Alon, R. Yuster, and U. Zwick, "Color-coding," *J. ACM*, vol. 42, no. 4, pp. 844–856, 1995.

[11] G. Borradaile, P. Klein, D. Marx, and C. Mathieu, "Algorithms for Optimization Problems in Planar Graphs (Dagstuhl Seminar 13421)," *Dagstuhl Reports*, vol. 3, no. 10, pp. 36–57, 2014.

[12] F. Dorn, F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh, "Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs," *Inf. Comput.*, vol. 233, pp. 60–70, 2013.

[13] S. Tazari, "Algorithmic graph minor theory: approximation, parameterized complexity, and practical aspects," Ph.D. dissertation, Humboldt University of Berlin, 2010.

[14] ——, "Faster approximation schemes and parameterized algorithms on (odd-)H-minor-free graphs," *Theor. Comput. Sci.*, vol. 417, pp. 95–107, 2012.

[15] I. Sau and D. M. Thilikos, "Subexponential parameterized algorithms for degree-constrained subgraph problems on planar graphs," *J. Discrete Algorithms*, vol. 8, no. 3, pp. 330–338, 2010.

[16] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, "Subexponential-time parameterized algorithm for steiner tree on planar graphs," in *STACS*, ser. LIPIcs, vol. 20. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013, pp. 353–364.

[17] ——, "Network sparsification for Steiner problems on planar and bounded-genus graphs," in *FOCS*. IEEE Computer Society, 2014, pp. 276–285.

[18] P. N. Klein and D. Marx, "A subexponential parameterized algorithm for subset TSP on planar graphs," in *SODA*. SIAM, 2014, pp. 1812–1830.

[19] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof, "Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth," *Inf. Comput.*, vol. 243, pp. 86–111, 2015.

[20] F. Dorn, F. V. Fomin, and D. M. Thilikos, "Catalan structures and dynamic programming in H-minor-free graphs," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1606–1622, 2012.

[21] F. V. Fomin, D. Lokshtanov, and S. Saurabh, "Efficient computation of representative sets with applications in parameterized and exact algorithms," in *SODA*. SIAM, 2014, pp. 142–151.

[22] N. Robertson and P. D. Seymour, "Graph minors. XIII. The disjoint paths problem," *J. Combinatorial Theory Ser. B*, vol. 63, no. 1, pp. 65–110, 1995.

[23] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," *J. Graph Algorithms Appl.*, vol. 3, pp. 1–27, 1999.

[24] F. Dorn, "Planar Subgraph Isomorphism revisited," in *STACS*, ser. LIPIcs, vol. 5. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 263–274.

[25] J. Matoušek and R. Thomas, "On the complexity of finding iso- and other morphisms for partial k-trees," *Discrete Mathematics*, vol. 108, no. 1-3, pp. 343–364, 1992.

[26] H. L. Bodlaender, J. Nederlof, and T. van der Zanden, "Subexponential time algorithms for embedding H-minor free graphs," in *ICALP*, 2016, pp. 9:1–9:14.

[27] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, and Y. Villanger, "Local search: Is brute-force avoidable?" *J. Comput. Syst. Sci.*, vol. 78, no. 3, pp. 707–719, 2012.

[28] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos, "Bidimensional Structures: Algorithms, Combinatorics and Logic (Dagstuhl Seminar 13121)," *Dagstuhl Reports*, vol. 3, no. 3, pp. 51–74, 2013.

[29] M. Chimani, P. Mutzel, and B. Zey, "Improved Steiner tree algorithms for bounded treewidth," *J. Discrete Algorithms*, vol. 16, pp. 67–78, 2012.

[30] F. V. Fomin, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, "Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering," *CoRR*, vol. abs/1604.05999, 2016.

[31] N. Robertson and P. D. Seymour, "Graph minors. III. Planar tree-width," *J. Combinatorial Theory Ser. B*, vol. 36, pp. 49–64, 1984.

[32] E. D. Demaine and M. T. Hajiaghayi, "Equivalence of local treewidth and linear local treewidth and its algorithmic applications," in *SODA*. SIAM, 2004, pp. 840–849.

[33] N. Alon, P. D. Seymour, and R. Thomas, "Planar separators," *SIAM J. Discrete Math.*, vol. 7, no. 2, pp. 184–193, 1994.

[34] N. Linial and M. E. Saks, "Low diameter graph decompositions," *Combinatorica*, vol. 13, no. 4, pp. 441–454, 1993.

[35] Y. Bartal, "On approximating arbitrary metrices by tree metrics," in *STOC*, J. S. Vitter, Ed. ACM, 1998, pp. 161–168.

[36] M. Naor, L. J. Schulman, and A. Srinivasan, "Splitters and near-optimal derandomization," in *FOCS*. IEEE, 1995, pp. 182–191.

[37] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. A. Plotkin, "Approximating a finite metric by a small number of tree metrics," in *FOCS*. IEEE Computer Society, 1998, pp. 379–388.