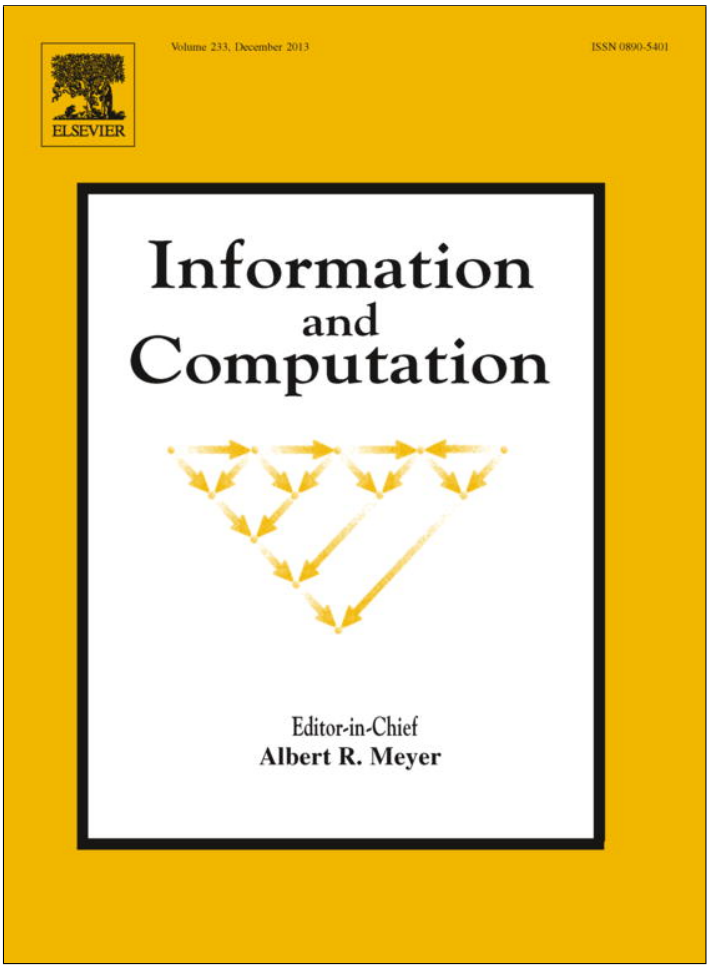

Information and Computation 233 (2013) 60–70

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs ☆

Frederic Dorn [a], Fedor V. Fomin [b,*], Daniel Lokshtanov [b], Venkatesh Raman [c], Saket Saurabh [c]

[a] SINTEF Energy Research, Trondheim, Norway
[b] Department of Informatics, University of Bergen, P.O. Box 7803, N-5020, Bergen, Norway
[c] The Institute of Mathematical Sciences, Chennai, India

ARTICLE INFO

Article history:
Received 6 December 2011
Received in revised form 27 August 2013
Available online 28 November 2013

Keywords:
Parameterized algorithms
Subexponential running time
Directed graph
Graph algorithms

ABSTRACT

In this paper we make the first step beyond bidimensionality by obtaining subexponential time algorithms for problems on directed graphs. We develop two different methods to achieve subexponential time parameterized algorithms for problems on sparse directed graphs. We exemplify our approaches with two well studied problems. For the first problem, k-Leaf Out-Branching, which is to find an oriented spanning tree with at least k leaves, we obtain an algorithm solving the problem in time $2^{\mathcal{O}(\sqrt{k}\log k)}n+n^{\mathcal{O}(1)}$ on directed graphs whose underlying undirected graph excludes some fixed graph H as a minor. For the special case when the input directed graph is planar, the running time can be improved to $2^{\mathcal{O}(\sqrt{k})}n+n^{\mathcal{O}(1)}$. The second example is a generalization of the Directed Hamiltonian Path problem, namely k-Internal Out-Branching, which is to find an oriented spanning tree with at least k internal vertices. We obtain an algorithm solving the problem in time $2^{\mathcal{O}(\sqrt{k}\log k)}+n^{\mathcal{O}(1)}$ on directed graphs whose underlying undirected graph excludes some fixed apex graph H as a minor. Finally, we observe that on these classes of graphs, the k-Directed Path problem is solvable in time $\mathcal{O}((1+\varepsilon)^k n^{f(\varepsilon)})$, for any $\varepsilon>0$, where f is some function of ε.

Our methods are based on non-trivial combinations of obstruction theorems for undirected graphs, kernelization, problem-specific combinatorial structures, and a layering technique similar to the one employed by Baker to obtain PTAS for planar graphs.

1. Introduction

Parameterized complexity theory is a framework for a refined analysis of hard (NP-hard) problems. Here, every input instance I of a problem Π is accompanied with an integer parameter k and Π is said to be fixed parameter tractable (FPT) if there is an algorithm running in time $f(k)\cdot n^{\mathcal{O}(1)}$, where $n=|I|$ and f is a computable function. A central problem in parameterized algorithms is to obtain algorithms with running time $f(k)\cdot n^{\mathcal{O}(1)}$ such that f is as slowly growing function as possible. This has led to the development of various graph algorithms with running time $2^{\mathcal{O}(k)}n^{O(1)}$ — notable ones include k-Feedback Vertex Set [9,12], k-Leaf Spanning Tree [39], k-Odd Cycle Transversal [43], k-Path [4], and k-Vertex Cover [13] in undirected graphs. A natural question was whether we can get *subexponential time* algorithms for these problems, that is,

☆ A preliminary version of this paper appeared in the proceedings of STACS 2010.

* Corresponding author. Fax: +47 55 58 41 99.
E-mail addresses: frederic.dorn@sintef.no (F. Dorn), fomin@ii.uib.no (F.V. Fomin), daniello@ii.uib.no (D. Lokshtanov), vraman@imsc.res.in (V. Raman), saket@imsc.res.in (S. Saurabh).

0890-5401/$ – see front matter
http://dx.doi.org/10.1016/j.ic.2013.11.006

can we have algorithms with running time $2^{o(k)}n^{\mathcal{O}(1)}$. It is now possible to show that these problems do not admit algorithms with running time $2^{o(k)}n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) [28,36] fails. Finding algorithms with subexponential running time on general undirected graphs is a trait uncommon to parameterized algorithms.

However, the situation changes completely when we consider problems on topological graph classes like planar graphs or graphs of bounded genus. In 2000, Alber et al. [1] obtained the first parameterized subexponential algorithm on undirected planar graphs by showing that k-Dominating Set is solvable in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$. This result triggered an extensive study of parameterized problems on planar and more general classes of sparse graphs like graphs of bounded genus, apex-minor-free graphs and H-minor-free graphs. All this work led to subexponential time algorithms for several fundamental problems like k-Feedback Vertex Set, k-Edge Dominating Set, k-Leaf Spanning Tree, k-Path, k–r-Dominating Set, k-Vertex Cover to name a few on planar graphs [1,18,33], and more generally, on H-minor-free graphs [19,21,22]. These algorithms are obtained by showing a combinatorial relation between the parameter and the structure of the input graph, and proofs require strong graph-theoretic arguments. This graph-theoretic and combinatorial component in the design of subexponential time parameterized algorithms makes it of an independent interest.

Demaine et al. [19] abstracted out the "common theme" among the parameterized subexponential time algorithms on sparse graphs and created the meta-algorithmic theory of bidimensionality. The bidimensionality theory unifies and improves almost all known previous subexponential algorithms on spare graphs. The theory is based on algorithmic and combinatorial extensions to various parts of Graph Minor Theory of Robertson et al. [44] and provides a simple criterion for checking whether a parameterized problem is solvable in subexponential time on sparse graphs. The theory applies to graph problems that are *bidimensional* in the sense that the value of the solution for the problem in question on a $k \times k$ grid or "grid like graph" is at least $\Omega(k^2)$ and the value of solution decreases while contracting or sometime deleting the edges. Problems that are bidimensional include k-Feedback Vertex Set, k-Edge Dominating Set, k-Leaf Spanning Tree, k-Path, k–r-Dominating Set, k-Vertex Cover and many others. In most cases we obtain subexponential time algorithms for a problem using bidimensionality theory in following steps. Given an instance (G, k) to a bidimensional problem Π, in polynomial time we either decide that it is a YES-instance to Π or the treewidth of G is $\mathcal{O}(\sqrt{k})$. In the second case, using known parameterized constant factor approximation algorithm for the treewidth [5,10], we find a tree decomposition of width $\mathcal{O}(\sqrt{k})$ for G and then solve the problem by doing dynamic programming over the obtained tree decomposition. This approach combined with Catalan structure based dynamic programming over graphs of bounded treewidth has led to $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ time algorithm for k-Feedback Vertex Set, k-Edge Dominating Set, k-Leaf Spanning Tree, k-Path, k–r-Dominating Set, k-Vertex Cover and many others on planar graphs [18,19,26] and in some cases like k-Dominating Set and k-Path on H-minor-free graphs [19,25]. We refer to surveys by Demaine and Hajiaghayi [22] and Dorn et al. [24] for further details on bidimensionality and subexponential parameterized algorithms.

While bidimensionality theory is a powerful algorithmic framework on undirected graphs, it remains unclear how to apply it to problems on directed graphs (or digraphs). The main reason is that Graph Minor Theory for digraphs is still in a nascent stage and there are no suitable obstruction theorems so far. For an example, even the first step of the framework does not work easily on digraphs, as there is no unique notion of directed $k \times k$ grid. Given a $k \times k$ undirected grid we can make $2^{\Omega(k^2)}$ distinct directed grids by choosing orientations for the edges. Hence, unless we can guarantee a lower bound of $\Omega(k^2)$ on the size of solution of a problem for *any* directed $k \times k$ grid, the bidimensionality theory does not look applicable for problems on digraphs. Even the analogue of treewidth for digraphs is not unique and several alternative definitions have been proposed [7,35,37]. Only recently the first non-trivial subexponential parameterized algorithms on digraphs was obtained. Alon et al. [3] introduced the method of chromatic coding, a variant of color coding [4], and combined it with divide and conquer to obtain $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ for k-Feedback Arc Set in tournaments. See also [27,32,38] for more recent refinements of this result.

1.1. Our contribution

In this paper we make the first step beyond bidimensionality by obtaining subexponential time algorithms for problems on sparse digraphs. We develop two different methods to achieve subexponential time parameterized algorithms for digraph problems when the input graph can be embedded on some surface or the underlying undirected graph excludes some fixed graph H as a minor.

1.2. Quasi-bidimensionality

Our first technique can be thought of as "bidimensionality in disguise". We observe that given a digraph D, whose underlying undirected graph $UG(D)$ excludes some fixed graph H as a minor, if we can remove $o(k^2)$ vertices from the given digraph to obtain a digraph whose underlying undirected graph has a constant treewidth, then the treewidth of $UG(D)$ is $o(k)$ (Lemma 1). So given an instance (D, k) to a problem Π, in polynomial time we either decide that it is a YES-instance to Π or the treewidth of $UG(D)$ is $o(k)$. In the second case, as in the framework based on bidimensionality, we solve the problem by doing dynamic programming over the tree decomposition of $UG(D)$. The dynamic programming part of the framework is problem-specific and runs in time $2^{o(k)} + n^{\mathcal{O}(1)}$. We exemplify this technique on the well studied problem of k-Leaf Out-Branching.

We say that a subdigraph T on vertex set $V(T)$ of a digraph D on vertex set $V(D)$ is an *out-tree* if T is an oriented tree with only one vertex r of in-degree zero (called the *root*). The vertices of T of out-degree zero are called *leaves* and every other vertex is called an *internal vertex*. If T is a spanning out-tree, that is, $V(T) = V(D)$, then T is called an *out-branching* of D. Now we are in position to define the problem formally.

k-Leaf Out-Branching (k-LOB): Given a digraph D with the vertex set $V(D)$ and the arc set $A(D)$ and a positive integer k, check whether there exists an out-branching with at least k leaves.

The study of k-Leaf Out-Branching has been at forefront of research in parameterized algorithms in the last few years. Alon et al. [2] showed that the problem is fixed parameter tractable by giving an algorithm that decides in time $f(k) \cdot n^{\mathcal{O}(1)}$ whether a strongly connected digraph has an out-branching with at least k leaves. Bonsma and Dorn [11] extended this result to all digraphs, and improved the running time of the algorithm. Kneis et al. [39] provided a parameterized algorithm solving the problem in time $4^k n^{\mathcal{O}(1)}$. This result was further improved to $3.72^k n^{\mathcal{O}(1)}$ by Daligault et al. [16]. Binkele-Raible et al. [8] showed that for the rooted version of the problem, where apart from the input instance we are also given a root r and one asks for a k-leaf out-branching rooted at r, admits an $\mathcal{O}(k^3)$ kernel. Furthermore they also show that k-LOB does not admit polynomial kernel unless the polynomial hierarchy collapses to its third level. Finally, Daligault and Thomassé [17] obtained an $\mathcal{O}(k^2)$ kernel for the rooted version of the k-LOB problem and gave a constant factor approximation algorithm for k-LOB.

Using our new technique in combination with the kernelization result of [8], we obtain an algorithm for k-LOB that runs in time $2^{\mathcal{O}(\sqrt{k}\log k)}n + n^{\mathcal{O}(1)}$ for digraphs whose underlying undirected graph is H-minor-free. For planar digraphs our algorithm runs in $2^{\mathcal{O}(\sqrt{k})}n + n^{\mathcal{O}(1)}$ time.

1.3. Kernelization and divide & conquer

Our second technique is a combination of divide and conquer, kernelization and dynamic programming over graphs of bounded treewidth. Here, using a combination of kernelization and a Baker style layering technique for obtaining polynomial time approximation schemes [6], we reduce the instance of a given problem to $2^{o(k)}n^{\mathcal{O}(1)}$ many new instances of the same problem. These new instances have the following properties: (a) the treewidth of the underlying undirected graph of these instances is bounded by $o(k)$; and (b) the original input is a YES-instance if and only if at least one of the newly generated instances is. We exhibit this technique on the k-Internal Out-Branching problem, a parameterized version of a generalization of Directed Hamiltonian Path.

k-Internal Out-Branching (k-IOB): Given a digraph D with the vertex set $V(D)$ and the arc set $A(D)$ and a positive integer k, check whether there exists an out-branching with at least k internal vertices.

Prieto and Sloper [42] studied the *undirected* version of this problem and gave an algorithm with running time $2^{4k\log k}n^{\mathcal{O}(1)}$ and obtained a kernel of size $\mathcal{O}(k^2)$. Recently, Fomin et al. [29] obtained a vertex kernel of size $3k$ and gave an algorithm for the undirected version of k-IOB running in time $8^k n^{\mathcal{O}(1)}$. For the (directed) k-IOB, Gutin et al. [34] obtained an algorithm of running time $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$ and gave a kernel of size of $\mathcal{O}(k^2)$. Cohen et al. [14] improved the algorithm for k-IOB and gave an algorithm with running time $49.4^k n^{\mathcal{O}(1)}$. Here, we obtain a subexponential time algorithm for k-IOB with running time $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$ on directed planar graphs and digraphs whose underlying undirected graphs are apex-minor-free.

Finally, we also observe that for any $\varepsilon > 0$, there is an algorithm finding in time $\mathcal{O}((1+\varepsilon)^k n^{f(\varepsilon)})$ a directed path of length at least k (the k-Directed Path problem) in a digraph whose underlying undirected graph excludes a fixed apex graph as a minor. The existence of subexponential parameterized algorithm for this problem remains open.

2. Preliminaries

Let D be a digraph. By $V(D)$ and $A(D)$ we represent the vertex set and arc set of D, respectively. Given a subset $V' \subseteq V(D)$ of a digraph D, let $D[V']$ denote the digraph induced by V'. The *underlying graph* $UG(D)$ of D is obtained from D by omitting all orientations of arcs and by deleting one edge from each resulting pair of parallel edges. A vertex u of D is an *in-neighbor* (*out-neighbor*) of a vertex v if $uv \in A(D)$ ($vu \in A(D)$, respectively). The *in-degree* $d^-(v)$ (*out-degree* $d^+(v)$) of a vertex v is the number of its in-neighbors (out-neighbors). We say that a subdigraph T of a digraph D is an *out-tree* if T is an oriented tree with only one vertex r of in-degree zero (called the *root*). The vertices of T of out-degree zero are called *leaves* and every other vertex is called an *internal vertex*. If T is a spanning out-tree, that is, $V(T) = V(D)$, then T is called an *out-branching* of D. An out-branching (respectively, out-tree) rooted at r is called *r-out-branching* (respectively, r-out-tree). We define the operation of a *contraction of a directed arc* as follows. An arc uv is contracted as follows: add a new vertex u', and for each arc wv or wu add the arc wu' and for an arc vw or uw add the arc $u'w$, remove all arcs incident to u and v and the vertices u and v. We call a loopless digraph D *rooted*, if there exists a pre-specified vertex r of in-degree 0 as a root r and $d^+(r) \geqslant 2$. The rooted digraph D is called *connected* if every vertex in $V(D)$ is reachable from r by a directed path.

Let G be an undirected graph with the vertex set $V(G)$ and the edge set $E(G)$. For a subset $V' \subseteq V(G)$, by $G[V']$ we mean the subgraph of G induced by V'. By $N(u)$ we denote the (open) neighborhood of u that is the set of all vertices adjacent to u and by $N[u] = N(u) \cup \{u\}$. Similarly, for a subset $D \subseteq V$, we define $N[D] = \bigcup_{v \in D} N[v]$. The *diameter* of a graph G, denoted by $diam(G)$, is defined to be the maximum length of a shortest path between any pair of vertices of $V(G)$.

Given an edge $e = uv$ of a graph G, the graph G/e is obtained by contracting the edge uv; that is, we get G/e by identifying the vertices u and v and removing all the loops and duplicate edges. A *minor* of a graph G is a graph H that can be obtained from a subgraph of G by contracting edges. A graph class $\mathcal{C}$ is *minor closed* if any minor of any graph in $\mathcal{C}$ is also an element of $\mathcal{C}$. A minor closed graph class $\mathcal{C}$ is *H-minor-free* or simply *H-free* if $H \notin \mathcal{C}$. A graph H is called an apex graph if the removal of one vertex makes it a planar graph.

A *tree decomposition* of an (undirected) graph G is a pair (X, T) where T is a tree whose vertices we will call *nodes* and $X = (\{X_i \mid i \in V(T)\})$ is a collection of subsets of $V(G)$ such that (a) $\bigcup_{i \in V(T)} X_i = V(G)$, (b) for each edge $vw \in E(G)$, there is an $i \in V(T)$ such that $v, w \in X_i$, and (c) for each $v \in V(G)$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T. The *width* of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ equals $\max_{i \in V(T)}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. We use notation $\mathbf{tw}(G)$ to denote the treewidth of a graph G.

A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (where the degree of the polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. See [40] for an introduction to kernelization.

3. Method I — Quasi-bidimensionality

In this section we present our first approach. In general, a subexponential time algorithm using bidimensionality is obtained by showing that the solution for a problem in question is at least $\Omega(k^2)$ on $k \times k$ (contraction) grid minor. Using this we reduce the problem to a question on a graph with treewidth $o(k)$. We start with a lemma which enables us to use the framework of bidimensionality for digraph problems, though not as directly as for undirected graph problems.

Lemma 1. *Let D be a digraph such that $UG(D)$ excludes a fixed graph H as a minor. For any constant $c \geqslant 1$, if there exists a subset $S \subseteq V(D)$ with $|S| = s$ such that $\mathbf{tw}(UG(D[V(D) \setminus S])) \leqslant c$, then $\mathbf{tw}(UG(D)) = \mathcal{O}(\sqrt{s})$.*

Proof. By [22], for any H-minor-free graph G with treewidth more than r, there is a constant $\delta > 1$ only dependent on H such that G has an $\frac{r}{\delta} \times \frac{r}{\delta}$ grid minor. Suppose $\mathbf{tw}(UG(D)) > \delta(c+1)\sqrt{s}$. Then $UG(D)$ contains a $(c+1)\sqrt{s} \times (c+1)\sqrt{s}$ grid as a minor. Notice that this grid minor cannot be destroyed by any vertex set S of size at most s. That is, if we delete any vertex set S with $|S| = s$ from this grid, it will still contain a $(c+1) \times (c+1)$ subgrid. Thus, $UG(D[V(D) \setminus S])$ contains a $(c+1) \times (c+1)$ grid minor and hence by [28, Exercise 11.6] we have that $\mathbf{tw}(UG(D[V(D) \setminus S])) > c$. This shows that we need to delete more than s vertices from $UG(D)$ to obtain a graph with treewidth at most c, a contradiction. □

Using Lemma 1, we show that k-Leaf Out-Branching problem has a subexponential time algorithm on digraphs D such that $UG(D)$ exclude a fixed graph H as a minor. For our purpose a rooted version of k-LOB will also be useful which we define now. In the Rooted k-Leaf Out-Branching (R-k-LOB) problem apart from D and k, the root r of the tree searched for is also a part of the input and the objective is to check whether there exists an r-out-branching with at least k leaves. We now state our main combinatorial lemma and postpone its proof for a while.

Lemma 2. *Let D be a digraph, k be a positive integer and $r \in V(D)$ be the root. Then in polynomial time we can either construct an r-out-branching with at least k leaves in D or find a digraph D' such that the following hold.*

- *D has an r-out-branching with at least k leaves if and only if D' has an r-out-branching with at least k leaves;*
- *There exists a subset $S \subseteq V(D')$ such that $|S| = \mathcal{O}(k)$ and $\mathbf{tw}(UG(D'[V(D') \setminus S])) \leqslant c$, c is a constant.*

Furthermore, if $UG(D)$ excludes a fixed graph H as a minor, then so does $UG(D')$.

Combining Lemmata 1 and 2 we obtain the following result.

Lemma 3. *Let D be a digraph such that $UG(D)$ excludes a fixed graph H as a minor, k be a positive integer and $r \in V(D)$ be a root. Then in polynomial time either we can construct an r-out-branching with at least k leaves in D or find a digraph D' such that D has an r-out-branching with at least k leaves if and only if D' has an r-out-branching with at least k leaves. Furthermore $\mathbf{tw}(UG(D')) = \mathcal{O}(\sqrt{k})$.*

When a tree decomposition of $UG(D)$ is given, dynamic programming methods can be used to decide whether D has an out-branching with at least k leaves, see [34]. The time complexity of such a procedure is $2^{\mathcal{O}(w \log w)}n$, where $n = |V(D)|$ and w is the width of the tree decomposition. Now we are ready to prove the main theorem of this section assuming the combinatorial Lemma 2.

Theorem 1. *The k-LOB problem can be solved in time $2^{\mathcal{O}(\sqrt{k}\log k)}n + n^{\mathcal{O}(1)}$ on digraphs with n vertices such that the underlying undirected graph excludes a fixed graph H as a minor.*

Proof. Let D be a digraph where $UG(D)$ excludes a fixed graph H as a minor. We guess a vertex $r \in V(D)$ as a root. This only adds a factor of n to our algorithm. By Lemma 3, we can either compute, in polynomial time, an r-out-branching with at least k leaves in D or find a digraph D' with $UG(D')$ excluding a fixed graph H as a minor and $\mathbf{tw}(UG(D')) = \mathcal{O}(\sqrt{k})$. In the later case, using the constant factor approximation algorithm of Demaine et al. [23] for computing the treewidth of an H-minor-free graph, we find a tree decomposition of width $\mathcal{O}(\sqrt{k})$ for $UG(D')$ in time $n^{\mathcal{O}(1)}$. With the previous observation that we can find an r-out-branching with at least k leaves, if one exists, in time $2^{\mathcal{O}(\sqrt{k}\log k)}n$ using dynamic programming over graphs of bounded treewidth, we have that we can solve R-k-LOB in time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$. Hence, we need $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$ to solve the k-LOB problem.

To obtain the claimed running time bound we use the known kernelization algorithm after we have guessed the root r. Binkele-Raible et al. [8] gave an $\mathcal{O}(k^3)$ kernel for R-k-LOB which preserves the graph class. That is, given an instance (D, k) of R-k-LOB, in polynomial time they output an equivalent instance (D'', k) of R-k-LOB such that (a) if $UG(D)$ is H-minor-free then so is $UG(D'')$; and (b) $|V(D'')| = \mathcal{O}(k^3)$. We will use this kernel for our algorithm rather than the $\mathcal{O}(k^2)$ kernel for R-k-LOB obtained by Daligault and Thomassé [17], as they do not preserve the graph class. So after we have guessed the root r, we obtain an equivalent instance (D'', k) for R-k-LOB using the kernelization procedure described in [8]. Then using the algorithm described in the previous paragraph we can solve R-k-LOB in time $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$. Hence, we need $2^{\mathcal{O}(\sqrt{k}\log k)}n + n^{\mathcal{O}(1)}$ to solve k-LOB. □

Since this paper was submitted, new algorithmic techniques for solving connectivity problems on graphs of bounded treewidth have appeared. Cygan et al. [15] designed randomized algorithm solving k-LOB on a tree decomposition of width w of $UG(D)$ and running in time $6^w n^{O(1)}$. By making use of recent developments in this area, like rank based approach [9] or the approach based on representative sets [31], a deterministic algorithm solving k-LOB in time $2^{\mathcal{O}(w)}n^{O(1)}$ is possible. By making use of these observations, the running time in Theorem 1 can be improved to $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$.

3.1. Proof of Lemma 2

To prove the combinatorial lemma, we need a few results from the literature on out-branching problems. Recall that we have a directed graph D with a designated root vertex r. We start with some definitions given in [17]. A *cut* of D is a subset S such that there exists a vertex $z \in V(D) \setminus S$ such that z is not reachable from r in $D[V(D) \setminus S]$. We say that D is 2-*connected* if there exists no cut of size one in D or equivalently there are at least two vertex disjoint paths from r to every vertex in D.

Lemma 4. *(See [17, Theorem 1].) Let D be a rooted 2-connected digraph with r being its root. Let α be the number of vertices in D with in-degree at least 3. Then D has an out-branching rooted at r with at least $\alpha/6$ leaves and such an out-branching can be found in polynomial time.*

A vertex $v \in V(D)$ is called a *nice vertex* if v has an in-neighbor which is not its out-neighbor. The following lemma is proved in [17].

Lemma 5. *(See [17, Theorem 2].) Let D be a rooted 2-connected digraph rooted at a vertex r. Let β be the number of nice vertices in D. Then D has an out-branching rooted at r with at least $\beta/24$ leaves and such an out-branching can be found in polynomial time.*

Proof of Lemma 2. To prove the combinatorial lemma, we consider two cases based on whether or not D is 2-connected.

Case 1. D is a rooted 2-connected digraph.

We prove this case in the following claim.

Claim 1. *Let D be a rooted 2-connected digraph with root r and a positive integer k. Then in polynomial time, we can find an out-branching rooted at r with at least k leaves or find a set S of at most $30k$ vertices whose removal results in a digraph whose underlying undirected graph has treewidth one.*

Proof. If $\alpha \geqslant 6k$, then we are done by Lemma 4. Similarly if $\beta \geqslant 24k$, then we are done by Lemma 5. Hence we assume that $\alpha < 6k$ and $\beta < 24k$. Let S be the set of nice vertices and vertices of in-degree at least 3 in G. Then $|S| \leqslant \alpha + \beta \leqslant 30k$. Observe that $D[V(D) \setminus S]$ is simply a collection of directed paths where every edge of the path is a directed 2-cycle. This is because $D[V(D) \setminus S]$ contains only those vertices which are not nice (that is, those vertices whose in-neighbors are also

out-neighbors) and are of in-degree at most two. Hence, if there is an arc xy in $D[V(D) \setminus S]$, then the arc yx also exists in $D[V(D) \setminus S]$. Next we note that $D[V(D) \setminus S]$ does not contain a directed cycle of length more than two. We prove the last assertion as follows. Let $\mathcal{C}$ be a directed cycle in $D[V(D) \setminus S]$ of length at least 3. Since D is a rooted 2-connected digraph, we have a vertex v on the cycle $\mathcal{C}$ such that there is a path from r to v without using any other vertex from the cycle $\mathcal{C}$. This implies that the in-degree of v is at least 3 in D and hence $v \in S$, contrary to our assumption that $v \notin S$. This proves that $D[V(D) \setminus S]$ does not contain a directed cycle of length more than two. Hence the underlying undirected graph $UG(D[V(D) \setminus S])$ is just a collection of paths and hence $\mathbf{tw}(UG(D[V(D) \setminus S]))$ is one. □

Case 2. D is not 2-connected.

Since D is not 2-connected, it has cut-vertices, those vertices that separate r from some other vertices. We deal with the cut-vertices in three cases. Let x be a cut-vertex of D. The three cases we consider are following.

Case 2a. There exists an arc xy that disconnects at least two vertices from r.

In this case, we *contract the arc* xy. After repeatedly applying Case 2a, we obtain a digraph D' such that any arc out of a cut-vertex x of D' disconnects at most 1 vertex. The resulting digraph D' is the one required in the statement of the lemma. Since we have only contracted some arcs iteratively to obtain D', it is clear that $UG(D')$ also excludes H as a minor. The proof that such contraction does not decrease the number of leaves follows from a reduction rule given in [8]. We provide a proof for completion.

Claim 2. *Let D be a rooted connected digraph with root r, let xy be an arc that disconnects at least two vertices from r and D' be the digraph obtained after contracting the arc xy. Then D has an r-out-branching with at least k leaves if and only if D' has an r-out-branching with at least k leaves.*

Proof. Let the arc xy disconnect at least two vertices y and w from r and let D' be the digraph obtained from D by contracting the arc xy. Let T be an r-out-branching of D with at least k leaves. Since every path from r to w contains the arc xy, T contains xy as well and neither x nor y is a leaf of T. Let T' be the tree obtained from T by contracting xy. T' is an r-out-branching of D' with at least k leaves.

For the converse, let T' be an r-out-branching of D' with at least k leaves. Let x' be the vertex in D' obtained by contracting the arc xy, and let u be the parent of x' in T'. Notice that the arc ux' in T' was initially the arc ux before the contraction of xy, since there is no path from r to y avoiding x in D. We obtain an r-out-branching T of D from T', by replacing the vertex x' by the vertices x and y and adding the arcs ux, xy and arc sets $\{yz\colon\ x'z \in A(T') \wedge yz \in A(D)\}$ and $\{xz\colon\ x'z \in A(T') \wedge yz \notin A(D)\}$. All these arcs belong to $A(D)$ because all the out-neighbors of x' in D' are out-neighbors either of x or of y in D. Finally, x' must be an internal vertex of T' since x' disconnects w from r. Hence T has at least as many leaves as T'. □

Now we handle the remaining cut-vertices of D' as follows. Let $\mathcal{S}$ be the set of cut-vertices in D'. For every vertex $x \in \mathcal{S}$, we associate *a cut-neighborhood* $C(x)$, which is the set of out-neighbors of x such that there is no path from r to any vertex in $C(x)$ in $D'[V(D') \setminus \{x\}]$. By $C[x]$ we denote $C(x) \cup \{x\}$. The following observation is used to handle other cases.

Claim 3. *Let $\mathcal{S}$ be the set of cut-vertices in D'. Then for every pair of vertices $x, y \in \mathcal{S}$ and $x \neq y$, we have that $C[x] \cap C[y] = \emptyset$.*

Proof. To the contrary let us assume that $C[x] \cap C[y] \neq \emptyset$. We note that $C[x] \cap C[y]$ can only have a vertex $v \in \{x, y\}$. To prove this, assume to the contrary that we have a vertex $v \in C[x] \cap C[y]$ and $v \notin \{x, y\}$. But then it contradicts the fact that $v \in C[x]$, as x doesn't separate v from r due to the path between r and v through y. Thus, either $x \in C(y)$ or $y \in C(x)$. Without loss of generality let $y \in C(x)$. This implies that we have an arc xy and there exists a vertex $z \in C(y)$ such that $z \notin C(x)$. But then the arc xy disconnects at least two vertices y and z from r and hence Case 2a would have applied. This proves the claim. □

Now we distinguish cases based on cut-vertices having cut-neighborhood of size at least 2 or 1. Let $\mathcal{S}_{\geqslant 2}$ and $\mathcal{S}_{=1}$ be the subset of cut-vertices of D' having at least two cut-neighbors and exactly one neighbor respectively.

Case 2b. $\mathcal{S}_{\geqslant 2} \neq \emptyset$.

We first bound $|\mathcal{S}_{\geqslant 2}|$. Let $A_c = \{xy \mid x \in \mathcal{S}_{\geqslant 2},\ y \in C(x)\}$ be the set of out-arcs emanating from the cut-vertices in $\mathcal{S}_{\geqslant 2}$ to its cut-neighbors. We now prove the following structural claim which is useful for bounding the size of $\mathcal{S}_{\geqslant 2}$.

Claim 4. *If D' has an r-out-branching T' with at least k leaves then D' has an r-out-branching T with at least k leaves and containing all the arcs of A_c, that is, $A_c \subseteq A(T)$. Furthermore, when T' is given, then T can be found in polynomial time.*

Proof. Let T^* be an r-out-branching of D' with at least k leaves and containing the maximum number of arcs from the set A_c. If $A_c \subseteq A(T^*)$, then we are through. So let us assume that there is an arc $e = xy \in A_c$ such that $e \notin A(T^*)$. Notice that since the vertices of $\mathcal{S}_{\geqslant 2}$ are cut-vertices, they are always internal vertices in any out-branching rooted at r in D. In particular, the vertices of $\mathcal{S}_{\geqslant 2}$ are internal vertices in T^*. Furthermore by Claim 3 we know that y is an end-point of exactly one arc in A_c. Let z be the parent of y in T^*. Now obtain $T^*_e = T^* \setminus \{zy\} \cup \{xy\}$. Observe that T^*_e contains at least k leaves and has more arcs from A_c than T^*. This is contrary to our assumption that T^* is an r-out-branching of D' with at least k leaves and containing the maximum number of arcs from the set A_c. This proves that T^* contains all the arcs of A_c.

Observe that starting from any r-out-branching T' of D' we can obtain the desired T in polynomial time by simple arc exchange operations described in the previous paragraph. □

We know that in any out-tree, the number of internal vertices of out-degree at least 2 is bounded by the number of leaves. Hence if $|\mathcal{S}_{\geqslant 2}| \geqslant k$ then we obtain an r-out-branching T of D' with at least k leaves using Claim 4 and we are done. So from now onwards we assume that $|\mathcal{S}_{\geqslant 2}| = \ell \leqslant k-1$.

We now do a transformation to the given digraph D'. For every vertex $x \in \mathcal{S}_{\geqslant 2}$, we introduce an *imaginary* vertex x^i and add an arc ux^i if there is an arc $ux \in A(D')$ and add an arc x^iv if there is an arc $xv \in A(D')$. Basically we duplicate the vertices in $\mathcal{S}_{\geqslant 2}$. We also add edges xx^i for all $x \in \mathcal{S}_{\geqslant 2}$. Let the transformed graph be called D^{dup}. We have the following two properties about D^{dup}. First, no vertex in $\mathcal{S}_{\geqslant 2} \cup \{x^i | x \in \mathcal{S}_{\geqslant 2}\}$ is a cut-vertex in D^{dup}. We sum up the second property in the following claim.

Claim 5. *The digraph D' has an r-out-branching T with at least k leaves if and only if D^{dup} has an r-out-branching T' with at least $k + \ell$ leaves.*

Proof. Given an r-out-branching T of D' with at least k leaves, we obtain an out-branching T' of D^{dup} with at least $k + \ell$ leaves by adding an arc xx^i to T for every $x \in \mathcal{S}_{\geqslant 2}$. Since every vertex of $\mathcal{S}_{\geqslant 2}$ is an internal vertex in T, this process only adds $\{x^i \mid x \in \mathcal{S}_{\geqslant 2}\}$ as leaves and hence we have at least $k + \ell$ leaves in T'.

For the converse, assume that D^{dup} has an r-out-branching T' with at least $k + \ell$ leaves. First, we modify the out-branching so that not both of x and x^i are internal vertices and we do not lose any leaf. This can be done easily by making all out-arcs in the out-branching from x and making x^i a leaf. That is, if $N^+_{T'}(x^i)$ is the set of out-neighbors of x^i in T' then we delete the arcs x^iz, $z \in N^+_{T'}(x^i)$ and add xz for all $z \in N^+_{T'}(x^i)$. This process cannot decrease the number of leaves. Furthermore we can always assume that if exactly one of x and x^i is an internal vertex, then x is the internal vertex in T'. Now delete all the vertices of $\{x^i \mid x \in \mathcal{S}_{\geqslant 2}\}$ from T' and obtain T. Since the vertices in the set $\{x^i \mid x \in \mathcal{S}_{\geqslant 2}\}$ are leaves of T', we have that T is an r-out-branching in D'. Since in the whole process we have only deleted ℓ vertices we have that T has at least k leaves. □

Now we move on to the last case.

Case 2c. $\mathcal{S}_{=1} \neq \emptyset$.

Consider the arc set $A_p = \{xy \mid x \in \mathcal{S}_{=1},\ y \in C(x)\}$. Observe that $A_p \subseteq A(D') \subseteq A(D^{dup})$ and A_p forms a *matching* in D^{dup} because of Claim 3. Let D^{dup}_c be the digraph obtained from D^{dup} by contracting the arcs of A_p. That is, for every arc $uv \in A_p$, the contracted graph is obtained by identifying the vertices u and v as uv and removing all the loops and duplicate arcs.

Claim 6. *Let D^{dup}_c be the digraph obtained by contracting the arcs of A_p in D^{dup}. Then the following hold.*

1. *The digraph D^{dup}_c is 2-connected*;
2. *If D^{dup}_c has an r-out-branching T with at least $k + \ell$ leaves then D^{dup} has an r-out-branching with at least $k + \ell$ leaves.*

Proof. The digraph D^{dup}_c is 2-connected by the construction as we have iteratively removed all cut-vertices. If D^{dup}_c has an r-out-branching T with at least $k + \ell$ leaves then we can obtain an r-out-branching with at least $k + \ell$ leaves for D^{dup} by expanding each of the contracted vertices to arcs in A_p. □

We are now ready to combine the above claims to complete the proof of the lemma. We first apply Claim 1 on D^{dup}_c with $k + \ell$. Either we get an r-out-branching T' with at least $k + \ell$ leaves or a set S' of size at most $30(k + \ell)$ such that $\mathbf{tw}(UG(D^{dup}_c[V(D^{dup}_c) \setminus S]))$ is one. In the first case, by Claims 5 and 6 we get an r-out-branching T with at least k leaves in D'. In the second case we know that there is a vertex set S' of size at most $30(k+\ell)$ such that $\mathbf{tw}(UG(D^{dup}_c[V(D^{dup}_c) \setminus S']))$ is one. Let $S^* = \{u \mid uv \in S',\ vu \in S',\ u \in S'\}$ be the set of vertices obtained from S' by expanding the contracted vertices in S'. Clearly the size of $|S^*| \leqslant 2|S'| \leqslant 60(k + \ell) \leqslant 120k = \mathcal{O}(k)$. We now show that the treewidth of the underlying

undirected graph of $D^{dup}[V(D^{dup}) \setminus S^*]$ is at most 3. This follows from the observation that $\mathbf{tw}(UG(D_c^{dup}[V(D_c^{dup}) \setminus S']))$ is one. Hence given a tree decomposition of width one for $UG(D_c^{dup}[V(D_c^{dup}) \setminus S'])$ we can obtain a tree decomposition for $UG(D^{dup}[V(D^{dup}) \setminus S^*])$ by expanding the contracted vertices. This can only double the bag size and hence the treewidth of $UG(D^{dup}[V(D^{dup}) \setminus S^*])$ is at most 3, as the bag size can at most be 4. Now we take $S = S^* \cap V(D')$ and since $V(D') \subseteq V(D^{dup})$, we have that $\mathbf{tw}(UG(D[V(D) \setminus S])) \leqslant 3$. This concludes the proof of the lemma. □

4. Method II — Kernelization and divide & conquer

In this section we exhibit our second method of designing subexponential time algorithms for digraph problems through the k-Internal Out-Branching problem. In this method we utilize the known polynomial kernel for the problem and obtain a collection of $2^{o(k)}$ instances such that the input instance is a YES-instance if and only if one of the instances in our collection is. The property of the instances in the collection which we make use of is that the treewidth of the underlying undirected graph of each of these instances is $o(k)$. The last property brings dynamic programming on graphs of bounded treewidth into picture as the final step of the algorithm.

Here, we will solve a rooted version of the k-IOB problem, called Rooted k-Internal Out-Branching (R-k-IOB), where apart from D and k we are also given a root $r \in V(D)$, and the objective is to find an r-out-branching, if one exists, with at least k internal vertices. The k-IOB problem can be reduced to R-k-IOB by guessing the root r at the additional cost of $|V(D)|$ in the running time of the R-k-IOB problem. Henceforth, we will only consider R-k-IOB. We call an r-out-tree T with k internal vertices *minimal* if deleting any leaf results in an r-out-tree with at most $k-1$ internal vertices. A well-known result relating minimal r-out-tree T with k internal vertices with a solution to R-k-IOB is as follows.

Lemma 6. *(See [14, Lemmata 3.1 and 3.4].) Let D be a rooted connected digraph with root r. Then D has an r-out-branching T' with at least k internal vertices if and only if D has a minimal r-out-tree T with k internal vertices with $|V(T)| \leqslant 2k-1$. Furthermore, given a minimal r-out-tree T, we can find an r-out-branching T' with at least k internal vertices in polynomial time.*

We also need another known result about kernelization for k-IOB.

Lemma 7. *(See [34, Lemma 4.6].)* k-Internal Out-Branching *admits a polynomial kernel of size $8k^2 + 6k$.*

In fact, the kernelization algorithm presented in [34] works for all digraphs and has a unique reduction rule which only *deletes vertices*. This implies that if we start with a graph $G \in \mathscr{G}$ where $\mathscr{G}$ excludes a fixed graph H as a minor, then the graph G' obtained after applying kernelization algorithm still belongs to $\mathscr{G}$.

Our algorithm tries to find a minimal r-out-tree T with k internal vertices with $|V(T)| \leqslant 2k-1$ recursively. As the first step of the algorithm we obtain a set of $2^{o(k)}$ digraphs such that the underlying undirected graphs have treewidth $\mathcal{O}(\sqrt{k})$, and the original problem is a YES-instance if and only at least one of the $2^{o(k)}$ instances is a YES-instance. More formally, we prove the following lemma.

Lemma 8. *Let H be a fixed apex graph and $\mathscr{G}$ be a minor closed graph class excluding H as a minor. Let (D,k) be an instance to* k-Internal Out-Branching *such that $UG(D) \in \mathscr{G}$. Then there exists a collection*

$$\mathcal{C} = \left\{ (D_i, k', r) \;\middle|\; D_i \text{ is a subgraph of } D,\ k' \leqslant k,\ r \in V(D),\ 1 \leqslant i \leqslant \binom{8k^2+6k}{\sqrt{k}} \right\},$$

of instances such that $\mathbf{tw}(UG(D_i)) = \mathcal{O}(\sqrt{k})$ for all i and (D,k) has an out-branching with at least k internal vertices if and only if there exists an i, r and $k' \leqslant k$ such that (D_i, k', r) has an r-out-branching with at least k' internal vertices.

Proof. The idea of the proof is to a Baker style layering technique [6] combined with kernelization. In the first step we apply the kernelization algorithm given by Lemma 7 on (D,k) and obtain an equivalent instance (D',k') where $|D'| \leqslant 8k^2+6k$ and $k' \leqslant k$ for k-IOB. From now onwards we will confine ourselves to (D',k'). Observe that since the kernelization algorithm only deletes vertices to obtain the reduced instance from the input digraph, we have that $UG(D') \in \mathscr{G}$.

Now we reduce the k-IOB problem to the R-k-IOB problem by guessing a vertex $r \in V(D')$ as a root. Furthermore we try to find a minimal r-out-tree T with k' internal vertices with $|V(T)| \leqslant 2k'-1$. This suffices for our purpose if we know that every vertex in $V(D')$ is reachable from the root r, as in this case Lemma 6 is applicable.

We start with a BFS starting at the vertex r in $UG(D')$. Let the layers created by doing BFS on r be $L_0^r, L_1^r, \ldots, L_t^r$. If $t \leqslant \lceil\sqrt{k}\rceil$, then the collection $\mathcal{C}_r$ consists of (D',k',r). For $t \leqslant \sqrt{k}$, the fact that $\mathbf{tw}(D') = \mathcal{O}(\sqrt{k})$ follows from the comments later in the proof. Hence from now onwards we assume that $t > \lceil\sqrt{k}\rceil$. Now we partition the vertex set into $\lceil\sqrt{k}\rceil$ parts where the q-th part contains all vertices which are at a distance of $q \bmod (\lceil\sqrt{k}\rceil)$ from r. That is, let $V(D') = \bigcup_q P_q$, $q \in \{0,\ldots,\lceil\sqrt{k}\rceil - 1\}$. We define $P_q = \bigcup L^r_{q+i(\lceil\sqrt{k}\rceil+1)}$, $i \in \{0,\ldots,\lfloor\frac{t-\sqrt{k}}{\lceil\sqrt{k}\rceil+1}\rfloor\}$. It is clear from the definition of P_q that it partitions the vertex set $V(D')$. If the input is a YES-instance then there exists a partition P_a such that it contains at most

$\lceil \frac{2k'-1}{\lceil\sqrt{k}\rceil} \rceil \leqslant 2\sqrt{k}$ vertices of the minimal r-out-tree T we are seeking for. We guess the partition P_a and a subset Z of size at most $2\sqrt{k}$ of P_a and obtain the collection

$$\mathcal{C}_r(P_a) = \left\{ \left(D'\left[V\left(D'\right) \setminus P_a \cup Z\right], k', r\right) \mid Z \subseteq P_a,\ |Z| \leqslant 2\sqrt{k} \right\}.$$

We now claim that for every $Z \subseteq P_a$, $|Z| \leqslant 2\sqrt{k}$, $\mathbf{tw}(UG(D'[V(D') \setminus P_a \cup Z])) = \mathcal{O}(\sqrt{k})$. Let $V' = V(D') \setminus P_a$ be the set of vertices after removal of P_a from the vertex set of D'. Let the resulting underlying undirected graph be $G' = UG(D'[V'])$ with connected components $C_1, \ldots, C_\ell$. We show that each connected component C_i of G' has $\mathcal{O}(\sqrt{k})$ treewidth. More precisely, every connected component C_i of G' is a subset of at most $\lceil\sqrt{k}\rceil - 1$ consecutive layers of the BFS starting at r. If we start with $UG(D')$, and delete all BFS layers after these layers and contract all BFS layers before these layers into a single vertex v, we obtain a minor M of $UG(D')$. This minor M has diameter at most $\lceil\sqrt{k}\rceil$ and contains C_i as an induced subgraph. Since $UG(D') \in \mathscr{G}'$, we have that $M \in \mathscr{G}$. Furthermore, Demaine and Hajiaghayi [20] have shown that for any fixed apex graph H, every H-minor-free graph of diameter d has treewidth $\mathcal{O}(d)$. This is the reason that we exclude apex graphs only in this section. This implies that $\mathbf{tw}(C_i) \leqslant \mathbf{tw}(M) \leqslant \mathcal{O}(\sqrt{k})$. Notice that since every connected component of G' has treewidth $\mathcal{O}(\sqrt{k})$, G' itself has $\mathcal{O}(\sqrt{k})$ treewidth. Given a tree decomposition of width $\mathcal{O}(\sqrt{k})$ for G', we can obtain a tree decomposition of width $\mathcal{O}(\sqrt{k})$ for $UG(D'[V(D') \setminus P_a \cup Z])$ by adding Z to every bag. The collection $\mathcal{C}_r$ is given by $\bigcup_{a=0}^{\lceil\sqrt{k}\rceil} \mathcal{C}_r(P_a)$. Finally the collection $\mathcal{C} = \bigcup_{r \in V(D')} \mathcal{C}_r$.

By the pigeon hole principle we know that if (D', k') is a YES-instance then there exists a P_a containing at most $2\sqrt{k}$ vertices of the minimal tree T we are looking for. Since we have run through all $r \in V(D')$ as a potential root as well as all subsets of size at most $2\sqrt{k}$ as the possible intersection of $V(T)$ with P_a, we know that (D', k') has an out-branching with at least k internal vertices if and only if there exists an i, r and $k' \leqslant k$ such that $(D_i, k', r) \in \mathcal{C}$ has an r-out-branching with at least k' internal vertices. This concludes the proof of the lemma. □

Given a tree decomposition of width w for $UG(D)$, one can solve R-k-IOB in time $2^{\mathcal{O}(w \log w)}n$ using a dynamic programming over graphs of bounded treewidth as described in [34]. This brings us to the main theorem of this section.

Theorem 2. *The k-IOB problem can be solved in time $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$ on digraphs with n vertices such that the underlying undirected graph excludes a fixed apex graph H as a minor.*

Proof. As the first step of the algorithm we apply Lemma 8 and obtain collection $\mathcal{C}$ such that for every $(D, k, r) \in \mathcal{C}$, $\mathbf{tw}(UG(D)) \in \mathcal{O}(\sqrt{k})$. Then using the constant factor approximation algorithm of Demaine et al. [23] for computing the treewidth of an H-minor-free graph, we find a tree decomposition of width $\mathcal{O}(\sqrt{k})$ for $UG(D)$ in time $k^{\mathcal{O}(1)}$. Finally, we apply dynamic programming algorithm running in time $(\sqrt{k})^{\mathcal{O}(\sqrt{k})} = 2^{\mathcal{O}(\sqrt{k}\log k)}$ on each instance in $\mathcal{C}$. If for any of them we obtain a yes answer we return "yes", else we return "no". The running time of the algorithm is bounded by

$$|\mathcal{C}| \cdot 2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\sqrt{k}\log k)} \cdot 2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}.$$

We have an additive term of $n^{\mathcal{O}(1)}$ as we apply the algorithm only on the $\mathcal{O}(k^2)$ size kernel. This completes the proof. □

5. Conclusion and discussions

We have given the first subexponential parameterized algorithms on planar digraphs and on the class of digraphs whose underlying undirected graph excludes a fixed graph H or an apex graph as a minor. We have outlined two general techniques, and have illustrated them on two well studied problems concerning oriented spanning trees (out-branching) — one that maximizes the number of leaves and the other that maximizes the number of internal vertices. One of our techniques uses the grid theorem on H-minor graphs, albeit in a different way than how it is used on undirected graphs. The other uses a Baker type layering technique combined with kernelization and solves the problem on a subexponential number of problems whose instances have sublinear treewidth.

We believe that our techniques will be widely applicable and it would be interesting to find other problems where such subexponential algorithms are possible. Two famous open problems in this context are whether the k-Directed Path problem (does a digraph contains a directed path of length at least k) and the k-Directed Feedback Vertex Set problem (does a digraph can be turned into acyclic digraph by removing at most k vertices) have subexponential algorithms (at least) on planar digraphs. However, for the k-Directed Path problem, we can reach "almost" subexponential running time. More precisely, we have the following theorem.

Theorem 3. *For any $\varepsilon > 0$, there is δ such that the k-Directed Path problem is solvable in time $\mathcal{O}((1+\varepsilon)^k \cdot n^\delta)$ on digraphs with n vertices such that the underlying undirected graph excludes a fixed apex graph H as a minor.*

Proof. Let P be a path of length k in a digraph D. The vertex set of P can be covered by at most b balls of radius k/b in the metric of $UG(D)$. Let F be a subgraph of $UG(D)$ induced by the vertices contained in b balls of radius k/b. We claim

that there is a constant c (depending only on the size of the apex graph H), such that $\mathbf{tw}(F) \leqslant c \cdot k/\sqrt{b}$. Indeed, because F is apex-minor-free, it contains a partially triangulated $(d \cdot \mathbf{tw}(F) \times d \cdot \mathbf{tw}(F))$ grid as a contraction for some $d > 0$ [30]. One needs $\Omega((\mathbf{tw}(F)b/k)^2)$ balls of radius k/b to cover such a grid, and hence to cover F [18]. But on the other hand, F is covered by at most b balls of radius k/b, and the claim follows. By an easy adaptation of the algorithm from [25] for undirected H-minor-free graphs, it is possible to find in time $2^{\mathcal{O}(\mathbf{tw}(F))} \cdot n^{\mathcal{O}(1)}$ if the subdigraph of D with the underlying undirected graph F contains a directed path of length k. Thus these computations can be done in time $2^{c_H \cdot k/\sqrt{b}} \cdot n^{\mathcal{O}(1)}$ for some constant $c_H > 0$ depending only on the size of H.

Putting things together, to check if D contains a path of length k (and if yes, to construct such a path), we try all possible sets of b vertices B and for each such set we construct a graph F induced by vertices at distance at most k/b from vertices of B. If D contain a k-path, then this path should be covered by at least one such set of b balls. For each such graph, we check, if the corresponding directed subgraph contains a k-path. The total running time of the algorithm is

$$\mathcal{O}\left(\binom{n}{b} 2^{c \cdot k/\sqrt{b}} \cdot n^c\right) = \mathcal{O}\left(2^{c \cdot k/\sqrt{b}} \cdot n^{b+c}\right)$$

for some constant c. By putting $b = (c/(\log(1+\varepsilon)))^2$ and $\delta = b + c$, we complete the proof of the theorem. □

Let us remark that similar $\mathcal{O}((1+\varepsilon)^k n^{f(\varepsilon)})$ results can also be obtained for several other problems including Steiner Tree in apex-minor-free graphs. Recently, Pilipczuk et al. [41] gave a parameterized subexponential algorithm for this problem parameterized by the size of the Steiner tree.

References

[1] J. Alber, H.L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier, Fixed parameter algorithms for dominating set and related problems on planar graphs, Algorithmica 33 (2002) 461–493.
[2] N. Alon, F.V. Fomin, G. Gutin, M. Krivelevich, S. Saurabh, Spanning directed trees with many leaves, SIAM J. Discrete Math. 23 (2009) 466–476.
[3] N. Alon, D. Lokshtanov, S. Saurabh, Fast FAST, in: ICALP 09, in: Lect. Notes Comput. Sci., vol. 5555, Springer, 2009, pp. 49–58.
[4] N. Alon, R. Yuster, U. Zwick, Color-coding, J. ACM 42 (1995) 844–856.
[5] E. Amir, Approximation algorithms for treewidth, Algorithmica 56 (2010) 448–479.
[6] B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs, J. ACM 41 (1994) 153–180.
[7] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, J. Obdržálek, The DAG-width of directed graphs, J. Comb. Theory, Ser. B 102 (2012) 900–923.
[8] D. Binkele-Raible, H. Fernau, F.V. Fomin, D. Lokshtanov, S. Saurabh, Y. Villanger, Kernel(s) for problems with no kernel: On out-trees with many leaves, ACM Trans. Algorithms 8 (2012) 38.
[9] H.L. Bodlaender, M. Cygan, S. Kratsch, J. Nederlof, Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth, in: Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP), in: Lect. Notes Comput. Sci., vol. 7965, Springer, 2013, pp. 196–207.
[10] H.L. Bodlaender, P.G. Drange, M.S. Dregi, F.V. Fomin, D. Lokshtanov, M. Pilipczuk, A $O(c^k n)$ 5-approximation algorithm for treewidth, CoRR, arXiv:1304.6321, 2013.
[11] P. Bonsma, F. Dorn, Tight bounds and a fast FPT algorithm for directed max-leaf spanning tree, ACM Trans. Algorithms 7 (2011) 44.
[12] J. Chen, F.V. Fomin, Y. Liu, S. Lu, Y. Villanger, Improved algorithms for feedback vertex set problems, J. Comput. Syst. Sci. 74 (2008) 1188–1198.
[13] J. Chen, I.A. Kanj, G. Xia, Improved upper bounds for vertex cover, Theor. Comput. Sci. 411 (2010) 3736–3756.
[14] N. Cohen, F.V. Fomin, G. Gutin, E.J. Kim, S. Saurabh, A. Yeo, Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem, J. Comput. Syst. Sci. 76 (2010) 650–662.
[15] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. van Rooij, J.O. Wojtaszczyk, Solving connectivity problems parameterized by treewidth in single exponential time, in: FOCS 2011, IEEE, 2011, pp. 150–159.
[16] J. Daligault, G. Gutin, E.J. Kim, A. Yeo, FPT algorithms and kernels for the directed k-leaf problem, J. Comput. Syst. Sci. 76 (2010) 144–152.
[17] J. Daligault, S. Thomassé, On finding directed trees with many leaves, in: Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC), in: Lect. Notes Comput. Sci., vol. 5917, Springer, 2009, pp. 86–97.
[18] E.D. Demaine, F.V. Fomin, M. Hajiaghayi, D.M. Thilikos, Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs, ACM Trans. Algorithms 1 (2005) 33–47.
[19] E.D. Demaine, F.V. Fomin, M. Hajiaghayi, D.M. Thilikos, Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs, J. ACM 52 (2005) 866–893.
[20] E.D. Demaine, M. Hajiaghayi, Equivalence of local treewidth and linear local treewidth and its algorithmic applications, in: SODA 04, 2004, pp. 840–849.
[21] E.D. Demaine, M. Hajiaghayi, The bidimensionality theory and its algorithmic applications, Comput. J. 51 (2008) 292–302.
[22] E.D. Demaine, M. Hajiaghayi, Linearity of grid minors in treewidth with applications through bidimensionality, Combinatorica 28 (2008) 19–36.
[23] E.D. Demaine, M. Hajiaghayi, K. Kawarabayashi, Algorithmic graph minor theory: Decomposition, approximation, and coloring, in: FOCS 05, 2005, pp. 637–646.
[24] F. Dorn, F.V. Fomin, D.M. Thilikos, Subexponential parameterized algorithms, Comput. Sci. Rev. 2 (2008) 29–39.
[25] F. Dorn, F.V. Fomin, D.M. Thilikos, Catalan structures and dynamic programming in h-minor-free graphs, J. Comput. Syst. Sci. 78 (2012) 1606–1622.
[26] F. Dorn, E. Penninkx, H.L. Bodlaender, F.V. Fomin, Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions, Algorithmica 58 (2010) 790–810.
[27] U. Feige, Faster FAST (Feedback Arc Set in Tournaments), CoRR, arXiv:0911.5094, 2009.
[28] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, Berlin, 2006.
[29] F.V. Fomin, S. Gaspers, S. Saurabh, S. Thomassé, A linear vertex kernel for maximum internal spanning tree, in: ISAAC, in: Lect. Notes Comput. Sci., vol. 5878, Springer, 2009, pp. 275–282.
[30] F.V. Fomin, P.A. Golovach, D.M. Thilikos, Contraction obstructions for treewidth, J. Comb. Theory, Ser. B 101 (2011) 302–314.
[31] F.V. Fomin, D. Lokshtanov, S. Saurabh, Efficient computation of representative sets with applications in parameterized and exact algorithms, CoRR, arXiv:1304.4626, 2013.
[32] F.V. Fomin, M. Pilipczuk, Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph, in: Proceedings of the 21st Annual European Symposium on Algorithms (ESA), in: Lect. Notes Comput. Sci., vol. 8125, Springer, 2013, pp. 505–516.

[33] F.V. Fomin, D.M. Thilikos, Dominating sets in planar graphs: Branch-width and exponential speed-up, SIAM J. Comput. 36 (2006) 281–309.
[34] G. Gutin, I. Razgon, E.J. Kim, Minimum leaf out-branching and related problems, Theor. Comput. Sci. 410 (2009) 4571–4579.
[35] P. Hunter, S. Kreutzer, Kelly decompositions, games, and orderings, in: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), ACM Press, New York, NY, USA, 2007, pp. 637–644.
[36] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity, J. Comput. Syst. Sci. 63 (2001) 512–530.
[37] T. Johnson, N. Robertson, P.D. Seymour, R. Thomas, Directed tree-width, J. Comb. Theory, Ser. B 82 (2001) 138–154.
[38] M. Karpinski, W. Schudy, Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament, in: Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC), in: Lect. Notes Comput. Sci., vol. 6506, Springer, 2010, pp. 3–14.
[39] J. Kneis, A. Langer, P. Rossmanith, A new algorithm for finding trees with many leaves, Algorithmica 61 (2011) 882–897.
[40] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, Oxford, 2006.
[41] M. Pilipczuk, M. Pilipczuk, P. Sankowski, E.J. van Leeuwen, Subexponential-time parameterized algorithm for Steiner tree on planar graphs, in: N. Portier, T. Wilke (Eds.), 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), in: Leibniz Int. Proc. Inform. (LIPIcs), vol. 20, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2013, pp. 353–364.
[42] E. Prieto, C. Sloper, Reducing to independent set structure – the case of k-internal spanning tree, Nord. J. Comput. 12 (2005) 308–318.
[43] B.A. Reed, K. Smith, A. Vetta, Finding odd cycle transversals, Oper. Res. Lett. 32 (2004) 299–301.
[44] N. Robertson, P. Seymour, R. Thomas, Quickly excluding a planar graph, J. Comb. Theory, Ser. B 62 (1994) 323–348.