HAL
open science

Beyond Bidimensionality: Parameterized Subexponential Algorithms on Directed Graphs

Frederic Dorn, Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh

To cite this version:

Frederic Dorn, Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh. Beyond Bidimensionality: Parameterized Subexponential Algorithms on Directed Graphs. 27th International Symposium on Theoretical Aspects of Computer Science - STACS 2010, Inria Nancy Grand Est & Loria, Mar 2010, Nancy, France. pp.251-262. inria-00455210

HAL Id: inria-00455210

https://inria.hal.science/inria-00455210

Submitted on 9 Feb 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 251-262
www.stacs-conf.org

BEYOND BIDIMENSIONALITY: PARAMETERIZED SUBEXPONENTIAL ALGORITHMS ON DIRECTED GRAPHS

FREDERIC DORN [1] AND FEDOR V. FOMIN [1] AND DANIEL LOKSHTANOV [1] AND VENKATESH RAMAN [2] AND SAKET SAURABH [2]

[1] Department of Informatics, University of Bergen, Bergen, Norway.
E-mail address: {dorn|fedor.fomin|daniello}@ii.uib.no

[2] The Institute of Mathematical Sciences, Chennai, India.
E-mail address: {vraman|saket}@imsc.res.in

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 for any $\varepsilon > 0$, the k-Directed Path problem is solvable in time $\mathcal{O}((1+\varepsilon)^k n^{f(\varepsilon)})$, 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

1998 ACM Subject Classification: F.2.2; G.2.2.
Key words and phrases: Parameterized Subexponential Algorithms, Directed Graphs, Out-Branching, Internal Out-Branching.

© F. Dorn, F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh
Creative Commons Attribution-NoDerivs License

$f(k) \cdot n^{\mathcal{O}(1)}$ such that f is as slow 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 [7], k-Leaf Spanning Tree [26], k-Odd Cycle Transversal [29], k-Path [4], and k-Vertex Cover [8] in undirected graphs. A natural question was whether we can get *subexponential time* algorithms for these problems, that is, 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) [21, 25] 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, 12, 23], and more generally, on H-minor-free graphs [13, 14, 15]. 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. [13] 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 Minors Theory of Robertson and Seymour [30] and provides a simple criteria 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 $k \times k$ 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 an yes instance to Π or the treewidth of G is $\mathcal{O}(\sqrt{k})$. In the second case, using known constant factor approximation algorithm for the treewidth, 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 [12, 13, 19] and in some cases like k-Dominating Set and k-Path on H-minor free graphs [13, 17]. We refer to surveys by Demaine and Hajiaghayi [14] and

Dorn et al. [18] 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 $2^{\mathcal{O}(k^2)}$ 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. 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.

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.

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 $o(k)$. So given an instance (D, k) to a problem Π, in polynomial time we either decide that it is an 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 a 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 $\mathcal{O}(f(k)n)$ whether a strongly connected digraph has an out-branching with at least k leaves. Bonsma and Dorn [6] extended this result to all digraphs, and improved the running time of the algorithm. Recently, Kneis et al. [26] 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 Daligaut et al. [10]. Fernau et

al. [20] 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 a $\mathcal{O}(k^3)$ kernel. Furthermore they also show that k-LOB does not admit polynomial kernel unless polynomial hierarchy collapses to third level. Finally, Daligault and Thomassé [11] obtained a $\mathcal{O}(k^2)$ 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 kernelization result of [20], we get 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.

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 [5], 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 an yes instance if and only if at least one of the newly generated instance 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 [28] 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. [22] obtained a vertex kernel of size $3k$ and gave an algorithm for the undirected version of k-IOB running in time $8^k n^{\mathcal{O}(1)}$. Gutin et al. [24] obtained an algorithm of running time $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$ for k-IOB and gave a kernel of size of $\mathcal{O}(k^2)$ using the well known method of crown-decomposition. Cohen et al. [9] 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 which 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 $d^+(r) \geq 2$. 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 (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 $N[D] = \cup_{v \in D} N[v]$. 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 a (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 [27] 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 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 3.1. *Let D be a digraph such that $UG(D)$ excludes a fixed graph H as a minor. For any constant $c \geq 1$, if there exists a subset $S \subseteq V(D)$ with $|S| = s$ such that $\mathbf{tw}(UG(D[V(D) \setminus S])) \leq c$, then $\mathbf{tw}(UG(D)) = \mathcal{O}(\sqrt{s})$.*

Proof. By [14], 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 a $\frac{r}{\delta} \times \frac{r}{\delta}$ 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 can not 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 [21, 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 3.1, 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 3.2. *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 the 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 following holds.*

- *$UG(D')$ excludes the fixed graph H as a minor;*
- *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 $\mathbf{tw}(U(D'[V(D') \setminus S]) \leq c$, c a constant.*

Combining Lemmata 3.1 and 3.2 we obtain the following result.

Lemma 3.3. *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 [24]. 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 3.2.

Theorem 3.4. *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 3.3,

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. [16] for computing the treewidth of a 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 exists one, 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. Fernau et al. [20] 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 [11], 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 [20]. 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. ■

Given a tree decomposition of width w of $UG(D)$ for a planar digraph D, we can solve k-LOB using dynamic programming methods in time $2^{\mathcal{O}(w)}n$. This brings us to the following theorem.

Theorem 3.5. [⋆][1] *The* k-LOB *problem can be solved in time* $2^{\mathcal{O}(\sqrt{k})}n + n^{\mathcal{O}(1)}$ *on digraphs with* n *vertices when the underlying undirected graph is planar.*

3.1. Proof of Lemma 3.2

To prove the combinatorial lemma we need a few recent results from the literature on out-branching problems. We start with some definitions given in [11]. 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 3.6 ([11])**.** *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 [11].

Lemma 3.7 ([11])**.** *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 3.2. 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.*

[1]The proofs marked with [⋆] will appear in the final version of the paper.

We prove this case in the following claim.

Claim 1. [$\star$] *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.*

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 mentioned in 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 [20]. We provide a proof for completion.

Claim 2. [$\star$] *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.*

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 $\mathcal{S}_{\geq 2}$ 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) $\mathcal{S}_{\geq 2} \neq \emptyset$.

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

Claim 4. [$\star$] *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 such an out-branching can be found in polynomial time.*

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 $|\mathcal{S}_{\geq 2}| \geq k$ 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 $|\mathcal{S}_{\geq 2}| = \ell \leq k-1$.

We now do a transformation to the given digraph D'. For every vertex $x \in \mathcal{S}_{\geq 2}$, 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 $x^i v$ if there is an arc $xv \in A(D')$. Basically we duplicate the vertices in $\mathcal{S}_{\geq 2}$. Let the transformed graph be called D^{dup}. We have the following two properties about D^{dup}. First, no vertex in $\mathcal{S}_{\geq 2} \cup \{x^i | x \in \mathcal{S}_{\geq 2}\}$ is a cut vertex in D^{dup}. We sum up the second property in the following claim.

Claim 5. [$\star$] *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.*

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

1. *The digraph D_c^{dup} is 2-connected;*
2. *If D_c^{dup} 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_c^{dup} is 2-connected by the construction as we have iteratively removed all cut-vertices. If D_c^{dup} has an r-out-branching T with at least $k+\ell$ leaves then we can obtain a 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_c^{dup} 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 $\mathbf{tw}(UG(D_c^{dup}[V(D_c^{dup}) \setminus S]))$ 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 $\mathbf{tw}(UG(D_c^{dup}[V(D_c^{dup}) \setminus S']))$ 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 $|S^*| \leq 2|S'| \leq 60(k+\ell) \leq 120k = \mathcal{O}(k)$. 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 $tw(UG(D[V(D) \setminus S])) \leq 3$. 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 an "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 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 exists one, 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 4.1 ([9])**.** *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 $|V(T)| \leq 2k-1$. 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 4.2 ([24])**.** k-Internal Out-Branching *admits a polynomial kernel of size* $8k^2+6k$.

In fact, the kernelization algorithm presented in [24] 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 $|V(T)| \leq 2k-1$ 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 4.3. [$\star$] *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*

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

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 $k' \leq k$ such that (D_i, k', r) has an r-out-branching with at least k' internal vertices.

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 [24]. This brings us to the main theorem of this section.

Theorem 4.4. [$\star$] *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.*

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 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 5.1. [$\star$] *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.*

Let use remark that similar $\mathcal{O}((1+\varepsilon)^k n^{f(\varepsilon)})$ results can also be obtained for many other problems including Planar Steiner Tree.

References

[1] 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*, 33(4):461–493, 2002.

[2] N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, and S. Saurabh. Spanning directed trees with many leaves. *SIAM Journal on Discrete Mathematics*, 23(1):466–476, 2009.

[3] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *ICALP 09*, volume 5555 of *LNCS*, pages 49–58. Springer, 2009.

[4] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[5] B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[6] P. Bonsma and F. Dorn. Tight bounds and a fast FPT algorithm for directed max-leaf spanning tree. *to appear in Transaction on Algorithms*.

[7] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

[8] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. In *MFCS 06*, volume 4162 of *LNCS*, pages 238–249. Springer, 2006.

[9] N. Cohen, F. V. Fomin, G. Gutin, E. J. Kim, S. Saurabh, and A. Yeo. Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. In *COCOON*, volume 5609 of *LNCS*, pages 37–46, 2009.

[10] J. Daligault, G. Gutin, E. J. Kim, and A. Yeo. FPT algorithms and kernels for the directed k-leaf problem. *CoRR*, abs/0810.4946, 2008.

[11] J. Daligault and S. Thomassé. On finding directed trees with many leaves. In *IWPEC 09*, LNCS, page to appear, Berlin, 2009. Springer-Verlag.

[12] 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. Algorithms*, 1(1):33–47, 2005.

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

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

[15] E. D. Demaine and M. T. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Computer Journal*, 51(3):292–302, 2008.

[16] E. D. Demaine, M. T. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *FOCS 05*, pages 637–646, 2005.

[17] F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan structures and dynamic programming in H-minor-free graphs. In *SODA 08*, pages 631–640. SIAM, 2008.

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

[19] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *to appear in Algorithmica*.

[20] H. Fernau, F. V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. In *STACS*, pages 421–432, 2009.

[21] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, Berlin, 2006.

[22] F. V. Fomin, S. Gaspers, S. Saurabh, and S. Thomassé. A linear vertex kernel for maximum internal spanning tree. In *ISAAC 09*, page to appear. Springer, 2009.

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

[24] G. Gutin, E. J. Kim, and I. Razgon. Minimum leaf out-branching problems. *CoRR*, abs/0801.1979, 2008.

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

[26] J. Kneis, A. Langer, and P. Rossmanith. A new algorithm for finding trees with many leaves. In *ISAAC 08*, volume 5369 of *LNCS*, pages 270–281. Springer-Verlag, 2008.

[27] R. Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, Oxford, 2006.

[28] E. Prieto and C. Sloper. Reducing to independent set structure – the case of k-internal spanning tree. *Nord. J. Comput.*, 12(3):308–318, 2005.

[29] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[30] N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Th. Ser. B*, 62:323–348, 1994.

This work is licensed under the Creative Commons Attribution-NoDerivs License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/3.0/.