

Kernel(s) for Problems with No Kernel: On Out-Trees with Many Leaves

Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, Yngve Villanger

▶ **To cite this version:**

Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, et al.. Kernel(s) for Problems with No Kernel: On Out-Trees with Many Leaves. 26th International Symposium on Theoretical Aspects of Computer Science - STACS 2009, Feb 2009, Freiburg, Germany. pp.421-432. inria-00358112

HAL Id: inria-00358112
https://inria.hal.science/inria-00358112

Submitted on 6 Feb 2009

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.

Author manuscript, published in "26th International Symposium on Theoretical Aspects of Computer Science - STACS 2009 (2009) 421-432"

Symposium on Theoretical Aspects of Computer Science 2009 (Freiburg), pp. 421–432
www.stacs-conf.org

KERNEL(S) FOR PROBLEMS WITH NO KERNEL: ON OUT-TREES WITH MANY LEAVES (EXTENDED ABSTRACT)

HENNING FERNAU [1] AND FEDOR V. FOMIN [2] AND DANIEL LOKSHTANOV [2] AND DANIEL RAIBLE [1] AND SAKET SAURABH [2] AND YNGVE VILLANGER [2]

[1] Univ. Trier, FB 4, Abteilung Informatik, 54286 Trier, Germany.
E-mail address: {fernau|raible}@uni-trier.de

[2] Department of Informatics, University of Bergen, Bergen Norway.
E-mail address: {fedor.fomin|daniello|saket.saurabh|yngve.villanger}@ii.uib.no

Abstract. The k-Leaf Out-Branching problem is to find an out-branching, that is a rooted oriented spanning tree, with at least k leaves in a given digraph. The problem has recently received much attention from the viewpoint of parameterized algorithms. Here, we take a kernelization based approach to the k-Leaf-Out-Branching problem. We give the first polynomial kernel for Rooted k-Leaf-Out-Branching, a variant of k-Leaf-Out-Branching where the root of the tree searched for is also a part of the input. Our kernel has cubic size and is obtained using extremal combinatorics.

For the k-Leaf-Out-Branching problem, we show that no polynomial kernel is possible unless the polynomial hierarchy collapses to third level by applying a recent breakthrough result by Bodlaender et al. (ICALP 2008) in a non-trivial fashion. However, our positive results for Rooted k-Leaf-Out-Branching immediately imply that the seemingly intractable k-Leaf-Out-Branching problem admits a data reduction to n independent $O(k^3)$ kernels. These two results, tractability and intractability side by side, are the first ones separating *many-to-one kernelization* from *Turing kernelization*. This answers affirmatively an open problem regarding "cheat kernelization" raised by Mike Fellows and Jiong Guo independently.

1. Introduction

Parameterized decision problems are defined by specifying the input (I), the parameter (k), and the question to be answered. A parameterized problem that can be solved in time $f(k)|I|^{O(1)}$ where f is a function of k alone is said to be fixed parameter tractable (FPT). Kernelization is a powerful and natural technique in the design of parameterized algorithms. The main idea of kernelization is to replace a given parameterized instance (I, k) of a problem Π by a simpler instance (I', k') of Π in polynomial time, such that (I, k)

Key words and phrases: Parameterized Algorithms, Kernelization, Out-Branching, Max-Leaf, Lower Bounds.

The authors gratefully acknowledge the support given by a German-Norwegian research grant.

SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

© H. Fernau, F. V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger
Creative Commons Attribution-NoDerivs License

inria-00358112, version 1 - 6 Feb 2009

inria-00358112, version 1 - 6 Feb 2009

is a yes-instance if and only if (I', k') is a yes-instance and the size of I' is bounded by a function of k alone. The reduced instance I' is called the *kernel* for the problem. Typically kernelization algorithms work by applying reduction rules, which iteratively reduce the instance to an equivalent "smaller" instance. From this point of view, kernelization can be seen as pre-processing with an explicit performance guarantee, "a humble strategy for coping with hard problems, almost universally employed" [14].

A parameterized problem is said to have a polynomial kernel if we have a polynomial time kernelization algorithm which reduces the size of the input instance down to a polynomial in the parameter. There are many parameterized problems for which polynomial, and even linear kernels are known [9, 8, 13, 17, 25]. Notable examples include a $2k$-sized kernel for k-Vertex Cover [9], a $O(k^2)$ kernel for k-Feedback Vertex Set [25] and a $67k$ kernel for k-Planar-Dominating Set [8], among many others. While positive kernelization results have been around for quite a while, the first results ruling out polynomial kernels for parameterized problems have appeared only recently. In a seminal paper Bodlaender et al. [4] have shown that a variety of important FPT problems cannot have polynomial kernels unless the polynomial hierarchy collapses to the third level ($PH = \Sigma_p^3$), a well known complexity theory hypothesis. Examples of such problems are k-Path, k-Minor Order Test, k-Planar Graph Subgraph Test, and many others. However, while this negative result rules out the existence of a polynomial kernel for these problems, it does not rule out the possibility of a kernelization algorithm reducing the instance to $|I|^{O(1)}$ independent polynomial kernels. This raises the question of the relationship between *many-to-one kernelization* and *Turing kernelization*, see [3, 13, 17]: Is there a natural parameterized problem having no polynomial kernel, but where we can "cheat" this lower bound by providing $|I|^{O(1)}$ polynomial kernels? Besides being of theoretical interest, this type of results would be very desirable from a practical point of view, as well. We show k-Leaf Out-Branching as the first example of such a problem.

The Maximum Leaf Spanning Tree problem on connected undirected graphs is: find a spanning tree with the maximum number of leaves in a given input graph G. The problem is well studied both from an algorithmic [16, 22, 23] and combinatorial [11, 19, 21] point of view, as well as from the parameterized complexity perspective [5, 13, 15]. An extension of Maximum Leaf Spanning Tree to directed graphs is defined as follows. 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*. If T is a spanning out-tree, i.e., $V(T) = V(D)$, then T is called an *out-branching* of D. The Directed Maximum Leaf Out-Branching problem is to find an out-branching in a given digraph with the maximum number of leaves. The parameterized version of the Directed Maximum Leaf Out-Branching problem is k-Leaf Out-Branching, where for a given digraph D and integer k, it is asked to decide whether D has an out-branching with at least k leaves. If we replace "out-branching" with "out-tree" in the definition of k-Leaf Out-Branching, we get a problem called k-Leaf Out-Tree.

Unlike its undirected counterpart, the study of k-Leaf Out-Branching has only begun recently. Alon et al. [1, 2] proved that the problem is fixed parameter tractable (FPT) by providing an algorithm deciding in time $O(f(k)n)$ whether a strongly connected digraph has an out-branching with at least k leaves. Bonsma and Dorn [6] extended this result to connected digraphs, and improved the running time of the algorithm. Recently, Kneis et al. [20] provided a parameterized algorithm solving the problem in time $4^k n^{O(1)}$. This result was further improved by Daligaut et al. [10]. In a related work, Drescher

	k-Out-Tree	k-Out-Branching
Rooted	$O(k^3)$ kernel	$O(k^3)$ kernel
Unrooted	No $poly(k)$ kernel, n kernels of size $O(k^3)$	No $poly(k)$ kernel, n kernels of size $O(k^3)$

Table 1: Our Results

and Vetta [12] described an $\sqrt{OPT}$-approximation algorithm for the Directed Maximum Leaf Out-Branching problem. Let us remark that, despite similarities between directed and undirected variants of Maximum Leaf Spanning Tree, the directed case requires a totally different approach (except from [20]). However, the existence of a polynomial kernel for k-Leaf Out-Branching has not been addressed until now.

Our contribution. We prove that Rooted k-Leaf Out-Branching, where for a given vertex r one asks for a k-leaf out-branching rooted at r, admits a $O(k^3)$ kernel. A similar result also holds for Rooted k-Leaf Out-Tree, where we are looking for a rooted (not necessary spanning) tree with k leaves. While many polynomial kernels are known for undirected graphs, this is the first known non-trivial parameterized problem on digraphs admitting a polynomial kernel. To obtain the kernel we establish a number of results on the structure of digraphs not having a k-leaf out-branching. These results may be of independent interest.

In the light of our positive results it is natural to suggest that k-Leaf Out-Branching admits a polynomial kernel, as well. We find it a bit striking that this is not the case – k-Leaf Out-Branching and k-Leaf Out-Tree do not admit polynomial kernels unless $PH = \Sigma_p^3$. While the main idea of our proof is based on the framework of Bodlaender et al. [4], our adaptation is non-trivial. In particular, we use the cubic kernel obtained for Rooted k-Leaf Out-Branching to prove the lower bound. Our contributions are summarized in Table 1.

Finally, notice that the polynomial kernels for the rooted versions of our problems yield a "cheat" solution for the poly-kernel-intractable k-Leaf Out-Branching and k-Leaf Out-Tree. Let D be a digraph on n vertices. By running the kernelization for the rooted version of the problem for every vertex of D as a root, we obtain n graphs where each of them has $O(k^3)$ vertices, at least one of them having a k-leaf out-branching iff D does.

Most proofs had to be omitted due to space limitations. More information can be found at `http://arxiv.org/abs/0810.4796`.

2. Preliminaries

Let D be a directed graph or digraph for short. By $V(D)$ and $A(D)$, we represent the vertex set and arc set, respectively, of D. Given a subset $V' \subseteq V(D)$ of a digraph D, by $D[V']$ we mean the digraph induced on V'. A vertex y of D is an *in-neighbor* (*out-neighbor*) of a vertex x if $yx \in A$ ($xy \in A$). The *in-degree* (*out-degree*) of a vertex x is the number of its in-neighbors (out-neighbors) in D. Let $P = p_1p_2 \dots p_l$ be a given path. Then by $P[p_ip_j]$ we denote a subpath of P starting at vertex p_i and ending at vertex p_j. For a given vertex $q \in V(D)$, by q-out-branching (or q-out-tree) we denote an out-branching (out-tree) of D rooted at vertex q. We say that the removal of an arc uv (or a vertex set S) *disconnects* a vertex w from the root r if every path from r to w in D contains arc uv (or one of the

inria-00358112, version 1 - 6 Feb 2009

vertices in S). 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 say that a reduction rule is *safe* for a value k if whenever the rule is applied to an instance (D, k) to obtain an instance (D', k'), D has an r-out-branching with $\geq k$ leaves if and only if D' has an r-out-branching with $\geq k'$ leaves.

Proposition 2.1. [20] *Let D be a digraph and r be a vertex from which every vertex in $V(D)$ is reachable. Then if we have an out-tree rooted at r with k leaves then we also have an out-branching rooted at r with k leaves.*

Let T be an out-tree of a digraph D. We say that u is a *parent* of v and v is a *child* of u if $uv \in A(T)$. We say that u is an *ancestor* of v if there is a directed path from u to v in T. An arc uv in $A(D) \setminus A(T)$ is called a *forward* arc if u is an ancestor of v, a *backward* arc if v is an ancestor of u and a *cross* arc, otherwise.

3. Reduction Rules for Rooted k-Leaf Out-Branching

In this section we give all the data reduction rules we apply on the given instance of Rooted k-Leaf Out-Branching to shrink its size.

Reduction Rule 1. [Reachability Rule] *If there exists a vertex u which is disconnected from the root r, then return* No.

For the Rooted k-Leaf Out-Tree problem, Rule 1 translates into the following: If a vertex u is disconnected from the root r, then remove u and all in-arcs and out-arcs of u.

Reduction Rule 2. [Useless Arc Rule] *If vertex u disconnects a vertex v from the root r, then remove the arc vu.*

Lemma 3.1. *Reduction Rules 1 and 2 are safe.*

Reduction Rule 3. [Bridge Rule] *If an arc uv disconnects at least two vertices from the root r, contract arc uv.*

Lemma 3.2. *Reduction Rule 3 is safe.*

Reduction Rule 4. [Avoidable Arc Rule] *If a vertex set S, $|S| \leq 2$, disconnects a vertex v from the root r, $vw \in A(D)$ and $xw \in A(D)$ for all $x \in S$, then delete the arc vw.*

Lemma 3.3. *Reduction Rule 4 is safe.*

Reduction Rule 5. [Two Directional Path Rule] *If there is a path $P = p_1 p_2 \dots p_{l-1} p_l$ with $l = 7$ or $l = 8$ such that*

- *p_1 and $p_{in} \in \{p_{l-1}, p_l\}$ are the only vertices with in-arcs from the outside of P.*
- *p_l and $p_{out} \in \{p_1, p_2\}$ are the only vertices with out-arcs to the outside of P.*
- *The path P is the unique out-branching of $D[V(P)]$ rooted at p_1.*
- *There is a path Q that is the unique out-branching of $D[V(P)]$ rooted at p_{in}.*
- *The vertex after p_{out} on P is not the same as the vertex after p_l on Q.*

Then delete $R = P \setminus \{p_1, p_{in}, p_{out}, p_l\}$ and all arcs incident to these vertices from D. Add two vertices u and v and the arc set $\{p_{out}u, uv, vp_{in}, p_l v, vu, up_1\}$ to D.

Notice that every vertex on P has in-degree at most 2 and out-degree at most 2. Figure 1 gives an example of an application of Reduction Rule 5.

inria-00358112, version 1 - 6 Feb 2009

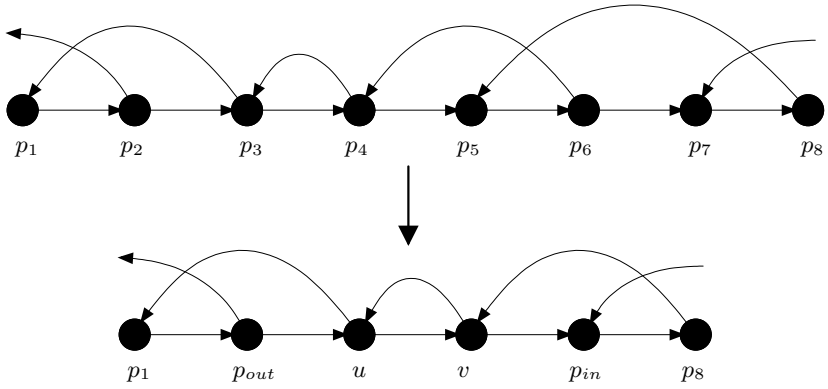


Figure 1: An Illustration of Reduction Rule 5.

Lemma 3.4. *Reduction Rule 5 is safe.*

Proof. Let D' be the graph obtained by performing Reduction Rule 5 to a path P in D. Let P_u be the path $p_1p_{out}uvp_{in}p_l$ and Q_v be the path $p_{in}p_lvup_1p_{out}$. Notice that P_u is the unique out-branching of $D'[V(P_u)]$ rooted at p_1 and that Q_v is the unique out-branching of $D'[V(P_u)]$ rooted at p_{in}.

Let T be an r-out-branching of D with at least k leaves. Notice that since P is the unique out-branching of $D[V(P)]$ rooted at p_1, Q is the unique out-branching of $D[V(P)]$ rooted at p_{in} and p_1 and p_{in} are the only vertices with in-arcs from the outside of P, $T[V(P)]$ is either a path or the union of two vertex disjoint paths. Thus, T has at most two leaves in $V(P)$ and at least one of the following three cases must apply.

(1) $T[V(P)]$ is the path P from p_1 to p_l.
(2) $T[V(P)]$ is the path Q from p_{in} to p_{out}.
(3) $T[V(P)]$ is the vertex disjoint union of a path $\tilde{P}$ that is a subpath of P rooted at p_1, and a path $\tilde{Q}$ that is a subpath of Q rooted at p_{in}.

In the first case we can replace the path P in T by the path P_u to get an r-out-branching of D' with at least k leaves. Similarly, in the second case, we can replace the path Q in T by the path Q_v to get an r-out-branching of D' with at least k leaves. For the third case, observe that $\tilde{P}$ must contain p_{out} since $p_{out} = p_1$ or p_1 appears before p_{out} on Q and thus, p_{out} can only be reached from p_1. Similarly, $\tilde{Q}$ must contain p_l. Thus, $T \setminus R$ is an r-out-branching of $D \setminus R$. We build an r-out-branching T' of D' by taking $T \setminus R$ and letting u be the child of p_{out} and v be the child of p_l. In this case T and T' have same number of leaves outside of $V(P)$ and T has at most two leaves in $V(P)$ while both u and v are leaves in T'. Hence T' has at least k leaves.

The proof for the reverse direction is similar. ∎

A digraph D is a *reduced instance* of Rooted k-Leaf Out-Branching if none of the reduction rules (Rules 1–5) can be applied to D. The following statement is easy to see:

Lemma 3.5. *For a digraph D on n vertices, we can obtain a reduced instance D' in polynomial time.*

inria-00358112, version 1 - 6 Feb 2009

inria-00358112, version 1 - 6 Feb 2009

4. Polynomial Kernel: Bounding a Reduced No-instance

Here, we show that any reduced no-instance of Rooted k-Leaf Out-Branching must have at most $O(k^3)$ vertices. In order to do so we start with T, a breadth-first search-tree (or BFS-tree for short) rooted at r, of a reduced instance D and look at a path P of T such that every vertex on P has out-degree one in T. We bound the number of endpoints of arcs with one endpoint in P and one endpoint outside of P (Section 4.1). We then use these results to bound the size of any maximal path with every vertex having out-degree one in T (Section 4.2). Finally, we combine these results to bound the size of any reduced no-instance of Rooted k-Leaf Out-Branching by $O(k^3)$.

4.1. Bounding the Number of Entry and Exit Points of a Path

Let D be a reduced no-instance, and T be a BFS-tree rooted at r. The BFS-tree T has at most $k-1$ leaves and hence at most $k-2$ vertices with out-degree at least 2 in T. Now, let $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T (P does not need to be a maximal path of T). Let T_1 be the subtree of T induced by the vertices reachable from r in T without using vertices in P and let T_2 be the subtree of T rooted at the child r_2 of p_l in T. Since T is a BFS-tree, it does not have any forward arcs, and thus p_lr_2 is the only arc from P to T_2. Thus all arcs originating in P and ending outside of P must have their endpoint in T_1.

Lemma 4.1. *Let D be a reduced instance, T be a BFS-tree rooted at r, and $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T. Let $up_i \in A(D)$, for some i between 1 and l, be an arc with $u \notin P$. There is a path P_{up_i} from r to p_i using the arc up_i, such that $V(P_{up_i}) \cap V(P) \subseteq \{p_i, p_l\}$.*

Proof. Let T_1 be the subtree of T induced by the vertices reachable from r in T without using vertices in P and let T_2 be the subtree of T rooted at the child r_2 of p_l in T. If $u \in V(T_1)$ there is a path from r to u avoiding P. Appending the arc up_i to this path yields the desired path P_{up_i}, so assume $u \in V(T_2)$. If all paths from r to u use the arc $p_{l-1}p_l$ then $p_{l-1}p_l$ is an arc disconnecting p_l and r_2 from r, contradicting the fact that Reduction Rule 3 can not be applied. Let P' be a path from r to u not using the arc $p_{l-1}p_l$. Let x be the last vertex from T_1 visited by P'. Since P' avoids $p_{l-1}p_l$ we know that P' does not visit any vertices of $P \setminus \{p_l\}$ after x. We obtain the desired path P_{up_i} by taking the path from r to x in T_1 followed by the subpath of P' from x to u appended by the arc up_i. ∎

Corollary 4.2. *Let D be a reduced no-instance, T be a BFS-tree rooted at r and $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T. There are at most k vertices in P that are endpoints of arcs originating outside of P.*

Lemma 4.3. *Let D be a reduced no-instance, T be a BFS-tree rooted at r and $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T. There are at most $7(k-1)$ vertices outside of P that are endpoints of arcs originating in P.*

Proof. Let X be the set of vertices outside P which are out-neighbors of the vertices on P. Let P' be the path from r to p_1 in T and r_2 be the unique child of p_l in T. First, observe that since there are no forward arcs, r_2 is the only out-neighbor of vertices in $V(P)$ in the subtree of T rooted at r_2. In order to bound the size of X, we differentiate between two kinds of out-neighbors of vertices on P: (a) Out-neighbors of P that are not in $V(P')$; and

(b) Out-neighbors of P in $V(P')$. First, observe that $|X \setminus V(P')| \leq k-1$. Otherwise we could have made an r-out-tree with at least k leaves by taking the path $P'P$ and adding $X \setminus V(P')$ as leaves with parents in $V(P)$.

In the rest of the proof we bound $|X \cap V(P')|$. Let Y be the set of vertices on P' with out-degree at least 2 in T and let $P_1, P_2, \ldots, P_t$ be the remaining subpaths of P' when vertices in Y are removed. For every $i \leq t$, $P_i = v_{i1}v_{i2}\ldots v_{iq}$. We define the vertex set Z to contain the two last vertices of each path P_i. The number of vertices with out-degree at least 2 in T is upper bounded by $k-2$ as T has at most $k-1$ leaves. Hence, $|Y| \leq k-2$, $t \leq k-1$ and $|Z| \leq 2(k-1)$.

Claim 1. *For every path $P_i = v_{i1}v_{i2}\ldots v_{iq}$, $1 \leq i \leq t$, there is either an arc u_iv_{iq-1} or u_iv_{iq} where $u_i \notin V(P_i)$.*

To see the claim observe that the removal of arc $v_{iq-2}v_{iq-1}$ does not disconnect the root r from both v_{iq-1} and v_{iq} else Rule 3 would have been applicable to our reduced instance. For brevity assume that v_{iq-1} is reachable from r after the removal of arc $v_{iq-2}v_{iq-1}$. Hence there exists a path from r to v_{iq}. Let u_iv_{iq} be the last arc of this path. The fact that the BFS-tree T does not have any forward arcs implies that $u_i \notin V(P_i)$.

To every path $P_i = v_{i1}v_{i2}\ldots v_{iq}$, $1 \leq i \leq t$, we associate an interval $I_i = v_{i1}v_{i2}\ldots v_{iq-2}$ and an arc $u_iv_{iq'}$, $q' \in \{q-1, q\}$. This arc exists by Claim 1. Claim 1 and Lemma 4.1 together imply that for every path P_i there is a path P_{ri} from the root r to $v_{iq'}$ that does not use any vertex in $V(P_i)\setminus\{v_{iq-1}, v_{iq}\}$ as an intermediate vertex. That is, $V(P_{ri} \cap (V(P_i)\setminus \{v_{iq-1}, v_{iq}\}) = \emptyset$.

Let P'_{ri} be a subpath of P_{ri} starting at a vertex x_i before v_{i1} on P' and ending in a vertex y_i after v_{iq-2} on P'. We say that a path P'_{ri} *covers* a vertex x if x is on the subpath of P' between x_i and y_i and we say that it *covers* an interval I_j if x_i appears before v_{j1} on the path P' and y_i appears after v_{jq-2} on P'. Hence, the path P'_{ri} covers the interval I_i.

Let $\mathcal{P} = \{P'_1, P'_2, \ldots, P'_l\} \subseteq \{P'_{r1}, \ldots, P'_{rt}\}$ be a minimum collection of paths, such that every interval I_i, $1 \leq i \leq t$, is covered by at least one of the paths in $\mathcal{P}$. Furthermore, let the paths of $\mathcal{P}$ be numbered by the appearance of their first vertex on P'. The minimality of $\mathcal{P}$ implies that for every $P'_i \in \mathcal{P}$ there is an interval $I'_i \in \{I_1, \ldots, I_t\}$ such that P'_i is the only path in $\mathcal{P}$ that covers I'_i.

Claim 2. *For every $1 \leq i \leq l$, no vertex of P' is covered by both P'_i and P'_{i+3}.*

The path P'_{i+1} is the only path in $\mathcal{P}$ that covers the interval I'_{i+1} and hence P'_i does not cover the last vertex of I'_{i+1}. Similarly P'_{i+2} is the only path in $\mathcal{P}$ that covers the interval I'_{i+2} and hence P'_{i+3} does not cover the first vertex of I'_{i+2}. Thus the set of vertices covered by both P'_i and P'_{i+3} is empty.

Since paths P'_i and P'_{i+3} do not cover a common vertex, we have that the end vertex of P'_i appears before the start vertex of P'_{i+3} on P' or is the same as the start vertex of P'_{i+3}. Partition the paths of $\mathcal{P}$ into three sets $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2$, where path $P'_i \in \mathcal{P}_{i \bmod 3}$. Also let $\mathcal{I}_i$ be the set of intervals covered by $\mathcal{P}_i$. Observe that every interval I_j, $1 \leq j \leq t$, is part of some $\mathcal{I}_i$ for $i \in \{0, 1, 2\}$.

Let $i \leq 3$ and consider an interval $I_j \in \mathcal{I}_i$. There is a path $P_{j'} \in \mathcal{P}_i$ that covers I_j such that both endpoints of $P_{j'}$ and none of the inner vertices of $P_{j'}$ lie on P'. Furthermore for any pair of paths P_a, $P_b \in \mathcal{P}_i$ such that $a < b$, there is a subpath in P' from the endpoint of P_a to the starting point of P_b. Thus for every $i \leq 3$ there is a path P^*_i from the root r to p_1 which does not use any vertex of the intervals covered by the paths in $\mathcal{P}_i$.

inria-00358112, version 1 - 6 Feb 2009

We now claim that the total number of vertices on intervals I_j, $1 \leq j \leq t$, which are out-neighbors of vertices on $V(P)$ is bounded by $3(k-1)$. If not, then for some i, the number of out-neighbors in $\mathcal{I}_i$ is at least k. Now we can make an r-out-tree with k leaves by taking any r-out-tree in $D[V(P_i^*) \cup V(P)]$ and adding the out-neighbors of the vertices on $V(P)$ in $\mathcal{I}_i$ as leaves with parents in $V(P)$.

Summing up the obtained upper bounds yields $|X| \leq (k-1)+|\{r_2\}|+|Y|+|Z|+3(k-1) \leq (k-1)+1+(k-2)+2(k-1)+3(k-1) = 7(k-1)$, concluding the proof. ∎

Remark: Observe that the path P used in Lemmas 4.1 and 4.3 and Corollary 4.2 need not be a maximal path in T with its vertices having out-degree one in T.

4.2. Bounding the Length of a Path: On Paths Through Nice Forests

For a reduced instance D, a BFS-tree T of D rooted at r, let $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T, and let S be the set of vertices in $V(P) \setminus \{p_l\}$ with an in-arc from the outside of P.

Definition 4.4. A subforest $F = (V(P), A(F))$ of $D[V(P)]$ is said to be a *nice forest of* P if the following three properties are satisfied: (a) F is a forest of directed trees rooted at vertices in S; (b) If $p_ip_j \in A(F)$ and $i < j$ then p_i has out-degree at least 2 in F or p_j has in-degree 1 in D; and (c) If $p_ip_j \in A(F)$ and $i > j$ then for all $q > i$, $p_qp_j \notin A(D)$.

In order to bound the size of a reduced no-instance D we are going to consider a nice forest with the maximum number of leaves. However, in order to do this, we first need to show the existence of a nice forest of P.

In the following discussion let D be a reduced no-instance, T be a BFS-tree T of D rooted at r, $P = p_1p_2 \dots p_l$ be a path in T such that all vertices in $V(P)$ have out-degree 1 in T and S be the set of vertices in $V(P) \setminus \{p_l\}$ with an in-arc from the outside of P.

Lemma 4.5. *There is a nice forest in* P.

For a nice forest F of P, we define the set of *key* vertices of F to be the set of vertices in S, the leaves of F, the vertices of F with out-degree at least 2 and the set of vertices whose parent in F has out-degree at least 2.

Lemma 4.6. *Let* F *be a nice forest of* P. *There are at most* $5(k-1)$ *key vertices of* F.

We can now turn our attention to a nice forest F of P with the maximum number of leaves. Our goal is to show that if the key vertices of F are too spaced out on P then some of our reduction rules must apply. We need some more observations concerning P and F.

Observation 4.7. [Unique Path] For any two vertices p_i, p_j in $V(P)$ such that $i < j$, $p_ip_{i+1} \dots p_j$ is the only path from p_i to p_j in $D[V(P)]$.

Corollary 4.8. *No arc* p_ip_{i+1} *is a forward arc of* F.

Observation 4.9. Let p_tp_j be an arc in $A(F)$ such that neither p_t nor p_j are key vertices, and $t \in \{j-1, j+1, \dots, l\}$. Then for all $q > t$, $p_qp_j \notin A(D)$.

Observation 4.9 follows directly from the definitions of a nice forest and key vertices.

Observation 4.10. If neither p_i nor p_{i+1} are key vertices, then either $p_ip_{i+1} \notin A(F)$ or $p_{i+1}p_{i+2} \notin A(F)$.

inria-00358112, version 1 - 6 Feb 2009

In the following discussion let F be a nice forest of P with the maximum number of leaves and let $P' = p_x p_{x+1} \dots p_y$ be a subpath of P containing no key vertices, and additionally having the property that $p_{x-1}p_x \notin A(F)$ and $p_y p_{y+1} \notin A(F)$.

Lemma 4.11. *$V(P')$ induces a directed path in F.*

In the following discussion let Q' be the directed path $F[V(P')]$.

Observation 4.12. For any pair of vertices $p_i, p_j \in V(P')$ if $i \leq j-2$ then p_j appears before p_i in Q'.

Lemma 4.13. *All arcs of $D[V(P')]$ are contained in $A(P') \cup A(F)$.*

Combining our previous observations with Lemma 4.11, we can show:

Lemma 4.14. *If $|P'| \geq 3$, there are exactly 2 vertices in P' that are endpoints of arcs starting outside of P'.*

Observation 4.15. Let $Q' = F[V(P')]$. For any pair of vertices u, v such that there is a path $Q'[uv]$ from u to v in Q', $Q'[uv]$ is the unique path from u to v in $D[V(P')]$.

Lemma 4.16. *For any vertex $x \notin V(P')$, there are at most 2 vertices in P' with arcs to x.*

This assertion follows by combining the previously derived lemmas and observations in a proof by contradition.

Corollary 4.17. *There are $\leq 14(k-1)$ vertices in P' with out-neighbors outside of P'.*

Proof. By Lemma 4.3, there are $\leq 7(k-1)$ vertices that are endpoints of arcs originating in P'. By Lemma 4.16, each such vertex is the endpoint of ≤ 2 arcs from vertices in P'. ∎

Lemma 4.18. $|P'| \leq 154(k-1)+10$.

Proof. Assume for contradiction that $|P'| > 154(k-1)+10$ and let X be the set of vertices in P' with arcs to vertices outside of P'. By Corollary 4.17, $|X| \leq 14(k-1)$. Hence there is a subpath of P' on at least $(154(k-1)+10)/(14(k-1)+1) = 9$ vertices containing no vertices of X. By Observation 4.10 there is a subpath $P'' = p_a p_{a+1} \dots p_b$ of P' on 7 or 8 vertices such that neither $p_{a-1}p_a$ nor $p_b p_{b+1}$ are arcs of F. By Lemma 4.11 $F[V(P'')]$ is a directed path Q''. Let p_q and p_t be the first and last vertices of Q'', respectively. By Lemma 4.14 p_a and p_q are the only vertices with in-arcs from outside of P''. By Observation 4.12 $p_q \in \{p_{b-1}, p_b\}$ and $p_t \in \{p_a, p_{a+1}\}$. By the choice of P'' no vertex of P'' has an arc to a vertex outside of P'. Furthermore, since P'' is a subpath of P' and Q'' is a subpath of Q' Lemma 4.13 implies that p_b and p_t are the only vertices of P' with out-arcs to the outside of P''. By Lemma 4.7, the path P'' is the unique out-branching of $D[V(P'')]$ rooted at p_a. By Lemma 4.15, the path Q'' is the unique out-branching of $D[V(P'')]$ rooted at p_q. By Observation 4.12 p_{b-2} appears before p_{a+2} in Q'' and hence the vertex after p_b in Q'' and p_{t+1} is not the same vertex. Thus Rule 5 can be applied on P'', contradicting the fact that D is a reduced instance. ∎

Lemma 4.19. *Let D be a reduced no-instance to* Rooted k-Leaf Out-Branching. *Then $|V(D)| = O(k^3)$. More specifically, $|V(D)| \leq 1540k^3$.*

Lemma 4.19 results in a cubic kernel for Rooted k-Leaf Out-Branching as follows.

Theorem 4.20. Rooted k-Leaf Out-Branching *and* Rooted k-Leaf Out-Tree *admits a kernel of size $O(k^3)$.*

inria-00358112, version 1 - 6 Feb 2009

Proof. Let D be the reduced instance of Rooted k-Leaf Out-Branching obtained in polynomial time using Lemma 3.5. If $|V(D)| > 1540k^3$, then return Yes. Else, we have an instance of size bounded by $O(k^3)$. The correctness of this step follows from Lemma 4.19 which shows that any reduced no-instance to Rooted k-Leaf Out-Branching has size bounded by $O(k^3)$. The result for Rooted k-Leaf Out-Tree follows similarly. ∎

5. Kernelization Lower Bounds

In the last section we gave a cubic kernel for Rooted k-Leaf Out-Branching. It is natural to ask whether the closely related k-Leaf Out-Branching has a polynomial kernel. The answer to this question, somewhat surprisingly, is no, unless an unlikely collapse of complexity classes occurs. To show this we utilize a recent result of Bodlaender et al. [4] that states that any *compositional* parameterized problem does not have a polynomial kernel unless the polynomial hierarchy collapses to the third level.

Definition 5.1 ([4])**.** A *composition algorithm* for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with: (1) $(y, k') \in L \iff (x_i, k) \in L$ for some $1 \leq i \leq t$; (2) k' is polynomial in k. A parameterized problem is *compositional* if there is a composition algorithm for it.

Now we state the main result of [4] which we need for our purpose.

Theorem 5.2 ([4])**.** *Let L be a compositional parameterized language whose unparameterized version $\widetilde{L}$ is* NP*-complete. Unless* PH$=\Sigma_p^3$*, there is no polynomial kernel for L.*

By considering disjoint graph unions, a relatively simple composition shows:

Theorem 5.3. *k-Leaf Out-Tree has no polynomial kernel unless* PH$=\Sigma_p^3$.

A *willow* graph [12] $D = (V, A_1 \cup A_2)$ is a directed graph such that $D' = (V, A_1)$ is a directed path $P = p_1 p_2 \ldots p_n$ on all vertices of D and $D'' = (V, A_2)$ is a directed acyclic graph with one vertex r of in-degree 0, such that every arc of A_2 is a backwards arc of P. p_1 is called the *bottom* vertex of the willow, p_n is called the *top* of the willow and P is called the *stem*. A *nice willow* graph $D = (V, A_1 \cup A_2)$ is a willow graph where $p_n p_{n-1}$ and $p_n p_{n-2}$ are arcs of D, neither p_{n-1} nor p_{n-2} are incident to any other arcs of A_2 and $D'' = (V, A_2)$ has a p_n-out-branching.

Observation 5.4. Let $D = (V, A_1 \cup A_2)$ be a nice willow graph. Every out-branching of D with the maximum number of leaves is rooted at the top vertex p_n.

Lemma 5.5. *k-Leaf Out-Tree in nice willow graphs is NP-hard under Karp reductions.*

Theorem 5.6. *k-Leaf Out-Branching has no polynomial kernel unless* PH$=\Sigma_p^3$.

Proof. We prove that if k-Leaf Out-Branching has a polynomial kernel then so does k-Leaf Out-Tree. Let (D, k) be an instance to k-Leaf Out-Tree. For every vertex $v \in V$ we make an instance (D, v, k) to Rooted k-Leaf Out-Tree. Clearly, (D, k) is a yes-instance for k-Leaf Out-Tree if and only if (D, v, k) is a yes-instance to Rooted k-Leaf Out-Tree for some $v \in V$. By Theorem 4.20 Rooted k-Leaf Out-Tree has a $O(k^3)$ kernel, so we can apply the kernelization algorithm for Rooted k-Leaf Out-Tree separately to each of the n instances of Rooted k-Leaf Out-Tree to get n instances

inria-00358112, version 1 - 6 Feb 2009

(D_1, v_1, k), (D_2, v_2, k), ..., (D_n, v_n, k) with $|V(D_i)| = O(k^3)$ for each $i \leq n$. By Lemma 5.5, k-Leaf Out-Branching in nice willow graphs is NP-complete under Karp reductions, so we can reduce each instance (D_i, v_i, k) of Rooted k-Leaf Out-Tree to an instance (W_i, b_i) of k-Leaf Out-Branching in nice willow graphs in polynomial time in $|D_i|$, and hence in polynomial time in k. Thus, in each such instance, $b_i \leq (k+1)^c$ for some fixed constant c independent of both n and k. Let $b_{max} = \max_{i \leq n} b_i$. Without loss of generality, $b_i = b_{max}$ for every i. This assumption is safe because if it does not hold we can modify the instance (W_i, b_i) by replacing b_i with b_{max}, subdividing the last arc of the stem $b_{max} - b_i$ times and adding an edge from r_i to each subdivision vertex.

¿From the instances (W_1, b_{max}), ..., (W_n, b_{max}) we build an instance $(D', b_{max}+1)$ of k-Leaf Out-Branching. Let r_i and s_i be the top and bottom vertices of W_i, respectively. We build D' simply by taking the disjoint union of the willows graphs $W_1, W_2, \ldots, W_n$ and adding in an arc $r_i s_{i+1}$ for $i < n$ and the arc $r_n s_1$. Let C be the directed cycle in D obtained by taking the stem of D' and adding the arc $r_n s_1$.

If for any $i \leq n$, W_i has an out-branching with at least b_{max} leaves, then W_i has an out-branching rooted at r_i with at least b_{max} leaves. We can extend this to an out-branching of D' with at least $b_{max}+1$ leaves by following C from r_i. In the other direction suppose D' has an out-branching T with at least $b_{max}+1$ leaves. Let i be the integer such that the root r of T is in $V(W_i)$. For any vertex v in $V(D')$ outside of $V(W_i)$, the only path from r to v in D' is the directed path from r to v in C. Hence, T has at most 1 leaf outside of $V(W_i)$. Thus, $T[V(W_1)]$ contains an out-tree with at least b_{max} leaves.

By assumption, k-Leaf Out-Branching has a polynomial kernel. Hence, we can apply a kernelization algorithm to get an instance (D'', k'') of k-Leaf Out-Branching with $|V(D'')| \leq (b_{max}+1)^{c_2}$ for a constant c_2 independent of n and b_{max} such that (D'', k'') is a yes-instance iff (D', b_{max}) is. Since k-Leaf Out-Tree is NP-complete, we can reduce (D'', k'') to an instance (D^*, k^*) of k-Leaf Out-Tree in polynomial time. Hence, $k^* \leq |V(D^*)| \leq (|V(D'')|+1)^{c_3} \leq (k+1)^{c_4}$ for some constants c_3 and c_4. Hence, if k-Leaf Out-Branching has a polynomial kernel then so does k-Leaf Out-Tree. Theorem 5.3 implies that k-Leaf Out-Branching has no polynomial kernel unless PH=Σ_p^3. ∎

6. Conclusion and Discussions

We demonstrated that Turing kernelization is a more poweful technique than many-to-one kernelization. We showed that while k-Leaf Out-Branching and k-Leaf Out-Tree do not have a polynomial kernel unless an unlikely collapse of complexity classes occurs, they do have n independent cubic kernels. Our paper raises far more questions than it answers. We believe that there are many more problems waiting to be addressed from the viewpoint of Turing kernelization. A few concrete open problems in this direction are as follows:

(1) Is there a framework to rule out the possibility of $|I|^{O(1)}$ polynomial kernels similar to the framework developed in [4]?
(2) Which other problems admit a Turing kernelization like the cubic kernels for k-Leaf Out-Branching and k-Leaf Out-Tree obtained here?
(3) Does there exist a problem for which we do not have a linear many-to-one kernel, but does have linear kernels from the viewpoint of Turing kernelization?

inria-00358112, version 1 - 6 Feb 2009

inria-00358112, version 1 - 6 Feb 2009

References

[1] N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, S. Saurabh. *Parameterized algorithms for Directed Maximum Leaf problems.* ICALP, LNCS 4596, 352–362 (2007).

[2] N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, S. Saurabh. *Better algorithms and bounds for Directed Maximum Leaf problems.* FSTTCS, LNCS 4855, 316–327 (2007).

[3] H. L. Bodlaender, E. D. Demaine, M. R. Fellows, J. Guo, D. Hermelin, D. Lokshtanov, M. Müller, V. Raman, J. van Rooij, F. A. Rosamond. *Open problems in parameterized and exact computation – IWPEC 2008.* Technical Report UU-CS-2008-017, Dept. of Inform. and Comput. Sci., Utrecht Univ.

[4] H. L. Bodlaender, R. G. Downey, M. R. Fellows, D. Hermelin. *On problems without polynomial kernels.* ICALP, LNCS 5125, 563–574 (2008).

[5] P. S. Bonsma, T. Brueggermann, G. J. Woeginger. *A faster FPT algorithm for finding spanning trees with many leaves.* MFCS, LNCS 2747, 259–268 (2003).

[6] P. S. Bonsma and F. Dorn. *Tight bounds and faster algorithms for Directed Max-Leaf Spanning Tree.* ESA, LNCS 5193, 222–233 (2008).

[7] Y. Caro, D. B. West, R. Yuster. *Connected domination and spanning trees with many leaves.* SIAM J. Discrete Math. 13, 202–211 (2000).

[8] J. Chen, H. Fernau, I. A. Kanj, G. Xia. *Parametric duality and kernelization: lower bounds and upper bounds on kernel size.* SIAM J. Comput. 37, 1077–1106 (2007).

[9] J. Chen, I. A. Kanj, W. Jia. *Vertex Cover: further observations and further improvements.* J. Algorithms 41, 280–301 (2001).

[10] J. Daligaut, G. Gutin, E. Jung Kim, A. Yeo. FPT algorithms and kernels for the Directed k-Leaf Problem. Technical Report 0810.4946v2, ArXiv, 2008.

[11] G. Ding, Th. Johnson, P. Seymour. *Spanning trees with many leaves.* J. Graph Theory 37, 189–197 (2001).

[12] M. Drescher and A. Vetta. *An approximation algorithm for the maximum leaf spanning arborescence problem.* To appear in ACM Trans. Algorithms.

[13] V. Estivill-Castro, M. R. Fellows, M. A. Langston, F. A. Rosamond, *FPT is P-Time extremal structure I.* Proc. ACiD, 1–41, (2005).

[14] M. R. Fellows. *The lost continent of polynomial time: preprocessing and kernelization.* IWPEC, LNCS 4169, 276–277 (2006).

[15] M. R. Fellows, C. McCartin, F. A. Rosamond, U. Stege. *Coordinated kernels and catalytic reductions: an improved FPT algorithm for Max Leaf Spanning Tree and other problems.* FSSTCS, LNCS 1974, 240–251 (2000).

[16] G. Galbiati, A. Morzenti, F. Maffioli. *On the approximability of some maximum spanning tree problems.* Theoret. Comp. Sci. 181, 107–118 (1997).

[17] J. Guo and R. Niedermeier. *Invitation to data reduction and problem kernelization.* ACM SIGACT News. 38, 31–45 (2007).

[18] R. Karp. *Reducibility among combinatorial problems.* Complexity of Computer Computations (Symposium Proceedings), Plenum Press (1972).

[19] D. J. Kleitman and D. B. West. *Spanning trees with many leaves.* SIAM J. Discrete Mathematics 4, 99–106 (1991).

[20] J. Kneis, A. Langer and P. Rossmanith. *A new algorithm for finding trees with many leaves.* To appear in Proc. of ISAAC 2008.

[21] N. Linial and D. Sturtevant. Storer's lower bound [24] holds even for graphs with minimum degree three. (quoted in [19])

[22] H. I. Lu and R. Ravi. *Approximating maximum leaf spanning trees in almost linear time.* J. Algorithms 29, 132–141 (1998).

[23] R. Solis-Oba. *2-approximation algorithm for finding a spanning tree with the maximum number of leaves.* ESA, LNCS 1461, 441–452 (1998).

[24] J. A. Storer. *Constructing full spanning trees for cubic graphs.* Inf. Process. Lett. 13, 8–11, 1981.

[25] S. Thomassé. *A quadratic kernel for feedback vertex set.* To appear in Proc. of SODA 2009.

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/.