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

DANIEL BINKELE-RAIBLE, Universität Trier
HENNING FERNAU, Universität Trier
FEDOR V. FOMIN, University of Bergen
DANIEL LOKSHTANOV, University of Bergen
SAKET SAURABH, The Institute of Mathematical Sciences
YNGVE VILLANGER, University of Bergen

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 with $O(k^3)$ vertices is obtained using extremal combinatorics.

For the $k$-LEAF-OUT-BRANCHING problem, we show that no polynomial-sized kernel is possible unless $coNP$ is in $NP/poly$. 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 polynomial-sized kernels. These two results, tractability and intractability side by side, are the first ones separating *Karp kernelization* from *Turing kernelization*. This answers affirmatively an open problem regarding "cheat kernelization" raised by Mike Fellows and Jiong Guo independently.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Kernelization, Lower Bounds, Max-Leaf Spanning Tree, Out-Branching, Parameterized Algorithms

**ACM Reference Format:**
Binkele-Raible, D., Fernau, H., Fomin, F. V., Lokshtanov, D., Saurabh, S., and Villanger, Y. 2011. Kernel(s) for problems with no kernel: on out-trees with many leaves. ACM Trans. Algor. 9, 4, Article 39 (March 2011), 20 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

The authors gratefully acknowledge the support given by a German-Norwegian research grant. F. Fomin is supported by the European Research Council (ERC) grant Rigorous Theory of Preprocessing, reference 267959. An extended abstract of this paper appeared in [Fernau et al. 2009].
Author's addresses: D. Binkele-Raible and H. Fernau, Fachbereich 4, Abteilung Informatik, Universität Trier, 54286 Trier, Germany, {fernau|raible}@informatik.uni-trier.de; F. V. Fomin, D. Lokshtanov and Y. Villanger, Department of Informatics, University of Bergen, Bergen, Norway, {fedor.fomin|daniello|yngve.villanger}@ii.uib.no; S. Saurabh, The Institute of Mathematical Sciences, C.I.T Campus, Taramani, Chennai 600 113, India, saket@imsc.res.in.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

## 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. In fact, kernelization characterizes fixed parameter tractability, that is, a problem is fixed parameter tractable if and only if there exists a polynomial time Karp reduction that maps a given instance to an instance of size effectively bounded in terms of the parameter.

The main idea of kernelization is to replace a given parameterized instance $(I, k)$ of a problem $\Pi$ by a simpler instance $(I', k')$ of $\Pi$ in polynomial time, such that $(I, k)$ 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" [Fellows 2006].

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 (vertex) kernels are known [Bodlaender 2009; Bodlaender et al. 2009; Chen et al. 2007; Chen et al. 2001; Estivill-Castro et al. 2005; Fomin et al. 2010; Guo and Niedermeier 2007; Thomassé 2010]. Notable examples include a $2k$-vertex kernel for $k$-VERTEX COVER [Chen et al. 2001], an $O(k^2)$ kernel for $k$-FEEDBACK VERTEX SET [Thomassé 2010] and a $67k$ kernel for $k$-PLANAR-DOMINATING SET [Chen et al. 2007], among many others. Notice that all the (bounds on) kernel sizes of graph problems mentioned in this paper are considering the number of vertices as reflecting the size of the instance. 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. [2009] have shown that a variety of important FPT problems cannot have polynomial kernels unless $coNP$ is in $NP/poly$, 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 *Karp kernelization* and *Turing kernelization*, a question raised in [Bodlaender et al. 2008; Estivill-Castro et al. 2005; Guo and Niedermeier 2007]. That is, can we have a natural parameterized problem for which there is no polynomial kernel but we can "cheat" this lower bound by providing $|I|^{O(1)}$ independent polynomial kernels. Besides being of theoretical interest, this type of results would be very desirable from a practical point of view as well. In this paper, we address the issue of Karp kernelization versus Turing kernelization through $k$-LEAF OUT-BRANCHING.

The MAXIMUM LEAF SPANNING TREE problem on connected undirected graphs is to 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 [Binkele-Raible and Fernau 2010; Galbiati et al. 1994; Lu and Ravi 1998; Solis-Oba 1998; Fomin et al. 2008] and combinatorial [Ding et al. 2001; Griggs et al. 1989; Griggs and Wu 1992; Kleitman and West 1991] point of view. The problem has been studied from the parameterized

complexity perspective as well [Bonsma et al. 2003; Estivill-Castro et al. 2005; Fellows et al. 2000; Raible and Fernau 2010]. 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 begun only recently. Alon *et al.* [2007; 2009] 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 [2008] extended this result to connected digraphs, and improved the running time of the algorithm. Recently, Kneis et al. [2008] provided a parameterized algorithm solving the problem in time $O(4^k n^{O(1)})$. This result was further improved by Daligault et al. [2010]. In a related work, Drescher and Vetta [2010] 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 [Kneis et al. 2008]). However, the existence of a polynomial kernel for $k$-LEAF OUT-BRANCHING has not been addressed until now. After the appearance of the conference version of this paper, Daligault et al. [2010] exhibited a vertex-linear kernel for ROOTED $k$-LEAF OUT-BRANCHING, restricted to directed acyclic graphs. Recently, Daligault and Thomassé [2009] improve our bound on the number of vertices in the kernel for ROOTED $k$-LEAF OUT-BRANCHING on general graphs from $O(k^3)$ to $O(k^2)$.

**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 polynomial kernel. In particular, we show how to obtain a kernel of $O(k^3)$ vertices. 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 $coNP \subseteq NP/poly$. While the main idea of our proof is based on the framework of Bodlaender et al. [2009], our adaptation is non-trivial. In particular, we use the polynomial kernel obtained for ROOTED $k$-LEAF OUT-BRANCHING to prove the lower bound. Our contributions are summarized in Table I.

Finally, let us remark that the polynomial kernels for the rooted versions of our problems provide a "cheat" solution for the poly-kernel-intractable problems $k$-LEAF OUT-BRANCHING and $k$-LEAF OUT-TREE. Indeed, let $D$ be a digraph on $n$ vertices. By running the kernelization for the rooted version of the problem for every vertex of $D$

Table I. Our Results

| | $k$-LEAF OUT-TREE | $k$-LEAF OUT-BRANCHING |
|---|---|---|
| Rooted | $O(k^3)$-vertex kernel | $O(k^3)$-vertex kernel |
| Unrooted | No $poly(k)$ kernel<br>$n$ kernels on $O(k^3)$ vertices | No $poly(k)$ kernel<br>$n$ kernels on $O(k^3)$ vertices |

as a root, we obtain $n$ graphs where each of them has $O(k^3)$ vertices, such that at least one of them has a $k$-leaf out-branching if and only if $D$ does.

## 2. PRELIMINARIES

*Graphs and digraphs.* 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 denote 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 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 at least $k$ leaves if and only if $D'$ has an $r$-out-branching with at least $k'$ leaves. We also need the following.

PROPOSITION 2.1. [Kneis et al. 2008] *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.

*Kernelization and Turing Kernelization.* A parameterized problem $\Pi$ is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet $\Gamma$. An instance of a parameterized problem consists of $(x, k)$, where $k$ is called the parameter. We assume that $k$ is *given in unary* and hence $k \leq |x|$. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means, for a given instance $(x, k)$, solvability in time $f(k) \cdot p(|x|)$, where $f$ is an arbitrary function of $k$ and $p$ is a polynomial in the input size. We refer to the monographs [Downey and Fellows 1999; Flum and Grohe 2006; Niedermeier 2006] for more information on parameterized complexity.

The notion of *kernelization* is formally defined as follows.

*Definition* 2.2. A kernelization algorithm, or in short, a *kernelization*, for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that given $(I, k) \in \Gamma^* \times \mathbb{N}$ outputs in time polynomial in $|I| + k$ a pair $(I', k') \in \Gamma^* \times \mathbb{N}$ such that

(a) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$, and
(b) $\max\{|I'|, k'\} \leq g(k)$,

where $g$ is some computable function. The reduced problem $(I', k')$ is referred to as the *kernel* and the function $g$ is referred to as the *size of the kernel*.

If $g(k) = k^{O(1)}$ or $g(k) = O(k)$, then we say that $\Pi$ admits a polynomial kernel and linear kernel respectively. As we are mostly dealing with graph problems, we try to be more specific when stating kernelization results by explicitly mentioning how we measure the size $|I'|$ of a reduced instance. For example, when we speak of $2k$-vertex kernel, we mean that $I'$ refers to a graph with at most $2k$ vertices.

We also define the notion of Turing kernelization. In order to do this we first define the notion of *$t$-oracle*.

*Definition* 2.3. A $t$-oracle for a parameterized problem $\Pi$ is an oracle that takes as input $(I, k)$ with $|I| \leq t$, $k \leq t$, and decides whether $(I, k) \in \Pi$ in constant time.

*Definition* 2.4. A parameterized problem $\Pi$ is said to have *$g(k)$-sized Turing kernel* if there is an algorithm which, given an input $(I, k)$ together with a $g(k)$-oracle for $\Pi$, decides whether $(I, k) \in \Pi$ in time polynomial in $|I|$ and $k$: the mentioned algorithm is also termed *Turing kernelization*.

Observe that the standard notion of kernelization (*Karp kernelization*) can be viewed as a special case of Turing kernelization given in Definition 2.2. More specifically, Karp kernelizations are equivalent to Turing kernelizations where the kernelization algorithm is only allowed to make one oracle call at the very end and must return the same answer as the oracle.

## 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 one: If a vertex $u$ is disconnected from the root $r$, then remove $u$ and all in-arcs to $u$ and out-arcs from $u$.

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.*

PROOF. If there exists a vertex which can not be reached from the root $r$, then a digraph cannot have any $r$-out-branching. For Reduction Rule 2, all paths from $r$ to $v$ contain the vertex $u$ and thus the arc $vu$ is a backward arc in any $r$-out-branching of $D$. ■

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.*

PROOF. Let the arc $uv$ disconnect at least two vertices $v$ and $w$ from $r$ and let $D'$ be the digraph obtained from $D$ by contracting the arc $uv$. Let $T$ be an $r$-out-branching of $D$ with at least $k$ leaves. Since every path from $r$ to $w$ contains the arc $uv$, $T$ contains $uv$ as well and neither $u$ nor $v$ are leaves of $T$. Let $T'$ be the tree obtained from $T$ by contracting $uv$. $T'$ is an $r$-out-branching of $D'$ with at least $k$ leaves.

In the opposite direction, let $T'$ be an $r$-out-branching of $D'$ with at least $k$ leaves. Let $u'$ be the vertex in $D'$ obtained by contracting the arc $uv$, and let $x$ be the parent of $u'$ in $T'$. Notice that the arc $xu'$ in $T'$ was initially the arc $xu$ before the contraction of $uv$, since there is no path from $r$ to $v$ avoiding $u$ in $D$. We make an $r$-out-branching

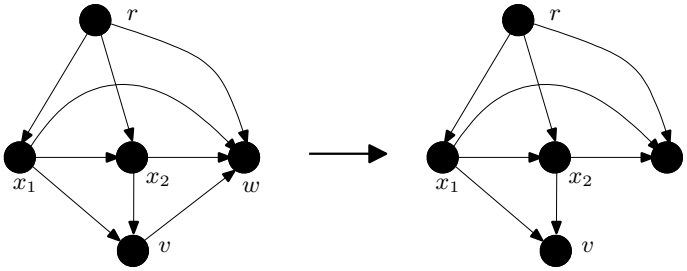


Fig. 1. Illustration of Reduction Rule 4. Vertices $x_1, x_2$ are the vertices contained in $S$. Every path from $r$ to $v$ passes through $x_1$ or $x_2$.

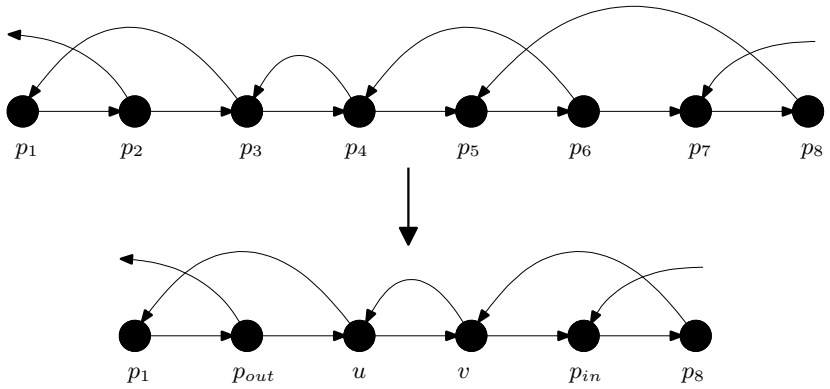


Fig. 2. Illustration of Reduction Rule 5.

$T$ of $D$ from $T'$ by replacing the vertex $u'$ by the vertices $u$ and $v$ and adding the arcs $xu$, $uv$ and arc sets $\{vy : u'y \in A(T') \wedge vy \in A(D)\}$ and $\{uy : u'y \in A(T') \wedge vy \notin A(D)\}$. All these arcs belong to $A(D)$, because all out-neighbors of $u'$ in $D'$ are out-neighbors either of $u$ or of $v$ in $D$. Finally, $u'$ must be an inner vertex of $T'$ since $u'$ disconnects $w$ from $r$. Hence, $T$ has at least as many leaves as $T'$. ∎

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.*

PROOF. Let $D'$ be the graph obtained by removing the arc $vw$ from $D$ and let $T$ be an $r$-out-branching of $D$. If $vw \notin A(T)$, $T$ is an $r$-out-branching of $D'$, so suppose $vw \in A(T)$. Any $r$-out-branching of $D$ contains the vertex $v$, and since all paths from $r$ to $v$ contain some vertex $x \in S$, some vertex $u \in S$ is an ancestor of $v$ in $T$. Let $T' = (T \cup uw) \setminus vw$. $T'$ is an out-branching of $D'$. Furthermore, since $u$ is an ancestor of $v$ in $T$, $T'$ has at least as many leaves as $T$. For the opposite direction, observe that any $r$-out-branching of $D'$ is also an $r$-out-branching of $D$. ∎

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 $p_{in}$ and ending in $p_{out}$.*
— *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 $\{p_{out}u, uv, vp_{in}, p_lv, vu, up_1\}$ to $D$.*

The unique out-branchings of $D[V(P)]$ rooted at $p_1$ and $p_{in}$ are paths, and as a consequence there are no forward arcs on the paths $P$ and $Q$, as this will generate more than one out-branching. Another consequence of this is that every vertex on $P$ has in-degree at most 2 and out-degree at most 2. Figure 2 gives an example of an application 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 the 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.

To show the other direction, let $T'$ be an $r$-out-branching of $D'$ with at least $k$ leaves. Notice that since $P_u$ is the unique out-branching of $D'[V(P_u)]$ rooted at $p_1$, $Q_v$ is the unique out-branching of $D'[V(P_u)]$ rooted at $p_{in}$ and $p_1$ and $p_{in}$ are the only vertices with in-arcs from the outside of $V(P_u)$, $T'[V(P_u)]$ is either a path or the union of two vertex disjoint paths. Thus, $T'$ has at most two leaves in $V(P_u)$ and at least one of the following three cases must apply.

(1) $T'[V(P_u)]$ is the path $P_u$ from $p_1$ to $p_l$.
(2) $T'[V(P_u)]$ is the path $Q_v$ from $p_{in}$ to $p_{out}$.
(3) $T'[V(P_u)]$ is the vertex disjoint union of a path $\tilde{P_u}$ that is a subpath of $P_u$ rooted at $p_1$, and a path $\tilde{Q_v}$ that is a subpath of $Q_v$ rooted at $p_{in}$.

In the first case, path $P_u$ in $T'$ can be replaced by the path $P$, resulting in an $r$-out-branching of $D$ with at least $k$ leaves. In the second case, an $r$-out-branching of $D'$ with at least $k$ leaves is obtained by replacing the path $Q_v$ in $T'$ by path $Q$. In the third

case, we have that $p_{out} = p_1$ or $p_1$ appears before $p_{out}$ on $Q_v$ and thus, $p_{out}$ can only be reached from $p_1$, hence, $\tilde{P_u}$ must contain $p_{out}$. By the same arguments, $\tilde{Q_v}$ must contain $p_l$. Thus, $T' \setminus \{u, v\}$ is an $r$-out-branching of $D' \setminus \{u, v\}$. Let $x$ be the vertex after $p_{out}$ on $P$, and let $y$ be the vertex after $p_l$ on $Q$. The vertices $x$ and $y$ must be distinct vertices in $R$ and thus there must be two vertex disjoint paths $P_x$ and $Q_y$ rooted at $x$ and $y$, respectively, so that $V(P_x) \cup V(Q_y) = R$. We build an $r$-out-branching $T$ from $(T' \setminus \{u, v\}) \cup P_x \cup Q_y$ by letting $x$ be the child of $p_{out}$ and $y$ be the child of $p_{in}$. In this case, $T'$ and $T$ have the same number of leaves outside of $V(P)$ and $T'$ has at most two leaves in $V(P_u)$, while both the leaf of $P_u$ and the leaf of $Q_v$ are leaves in $T$. Hence, $T$ has at least $k$ leaves. ∎

We say that 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 $D$. It is easy to observe from the description of the reduction rules that we can apply them in polynomial time, resulting in the following lemma.

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

## 4. POLYNOMIAL KERNEL: BOUNDING A REDUCED NO-INSTANCE

In this section, 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 let $T$ 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 $P = p_1 p_2 \dots p_l$ 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 $p_l r_2$ 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 $P = p_1 p_2 \dots p_l$ 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 $P$.*

PROOF. Let $S$ be the set of vertices in $P \setminus \{p_l\}$ that are endpoints of arcs originating outside of $P$. For the sake of contradiction suppose that there are at least $k+1$ vertices in $P$ that are endpoints of arcs originating outside of $P$. Then $|S| \geq k$. By Lemma 4.1 there exists a path from the root $r$ to every vertex in $S$, that avoids vertices of $P \setminus \{p_l\}$ as an intermediate vertex. Using these paths we can build an $r$-out-tree with every vertex in $S$ as a leaf. This $r$-out-tree can be extended to a $r$-out-branching with at least $k$ leaves by Proposition 2.1, contradicting the fact that $D$ is a NO-instance. ■

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 $P$.

— Out-neighbors of $P$ that are not in $V(P')$.
— 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 $P_1, P_2, \dots, P_t$ be the remaining subpaths of $P'$ when vertices in $Y$ are removed. For every $i \leq t$, $P_i = v_{i1}v_{i2} \dots v_{iq}$. We define the vertex set $Z$ containing $v_{i1}$ if $|P_i| = 1$ and otherwise the two last vertices of each path $P_i$. The number of vertices with out-degree at least 2 in $T$ is at most $k-2$ 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 $P_i = v_{i1}v_{i2} \dots v_{iq}$, $1 \leq i \leq t$,$3 \leq q$, there is either an arc $u_iv_{iq-1}$ or an arc $u_iv_{iq}$, where $u_i \notin V(P_i)$.*

The claim holds, because the removal of arc $v_{iq-2}v_{iq-1}$ does not disconnect the root $r$ from both $v_{iq-1}$ and $v_{iq}$—otherwise Rule 3 would have been applicable to our reduced instance. Without loss of generality, let us 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 $P_i = v_{i1}v_{i2} \dots v_{iq}$, $1 \leq i \leq t$, we associate an interval $I_i = v_{i1}v_{i2} \dots v_{iq-2}$ 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 $P'$. Observe that 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 conclude 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 $\mathcal{P}_i$.

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 $D$ and a BFS-tree $T$ of $D$ rooted at $r$, let $P = p_1 p_2 \ldots p_l$ 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 $P \setminus \{p_l\}$.

*Definition* 4.4. A subforest $F = (V(P) \setminus \{p_l\}, A(F))$ 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 $p_i p_j \in A(F)$ 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 $p_i p_j \in A(F)$ and $i > j$, then for all $l > q > i$, $p_q p_j \notin A(D)$.

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 have to prove that a nice forest always exists.

LEMMA 4.5. *There is a nice forest in $P \setminus \{p_l\}$.*

PROOF. We define a subgraph $F$ of $D[V(P) \setminus \{p_l\}]$ as follows: The vertex set of $F$ is $V(P) \setminus \{p_l\}$ and an arc $p_t p_s$ is in $A(F)$ if $p_s \notin S$ and $1 \leq t < l$ is the largest index such that $p_t p_s \in A(D)$.

Let us argue that $F$ satisfies all three conditions from the definition of the nice forest.

Graph $F$ satisfies condition (c) because selection of $t$ as the largest number such that $p_t p_s \in A(D)$ ensures that $p_q p_s \notin A(D)$ for all $t < q < l$. Condition (b) is ensured by the fact that for each arc $p_t p_s$ in $F$ we have $p_s \notin S$, this ensures that there is no in-arc from the outside of $P \setminus \{p_l\}$ ending in $p_s$. If $t < s$ then $t = s - 1$ and $p_t p_s$ is the only in-arc to $p_s$ as $1 \leq t < l$ is the maximum number such that $p_t p_s \in A(D)$, $P$ do not have any forward arcs, and there are no arcs from the outside of $P \setminus \{p_l\}$ ending in $p_s$. Finally, to prove (a), we have to show that $F$ is a forest. Suppose for a contradiction that there is a cycle $C$ in $F$. By definition of $F$, every vertex has in-degree at most 1, so $C$ must be a directed cycle. Since every vertex of $S$ has in-degree 0 in $F$, we have that $C \cap S = \emptyset$. Consider the highest numbered vertex $p_i$ of $C$. Since $P$ has no forward arcs, we have that $p_{i-1}$ is the predecessor of $p_i$ in $C$. The construction of $F$ implies that there are no arcs $p_q p_i$ in $A(D)$, where $l > q > i$. Also, $p_i$ does not have any in-arcs from outside of $P \setminus \{p_l\}$. Thus, $p_{i-1}$ disconnects $p_i$ from the root. Hence, by Rule 2 $p_i p_{i-1} \notin A(D)$. Let $p_j$ be the predecessor of $p_{i-1}$ in $C$. Since $p_i p_{i-1} \notin A(D)$ and $p_i$ is the highest numbered vertex in $C$, we have that $j < i - 1$. Hence $j = i - 2$. Path $P$ does not have forward arcs, $i - 2$ is the highest number such that $p_{i-2} p_{i-1} \in A(D)$, and there are no in-arcs from outside of $P \setminus \{p_l\}$ to $p_{i-2}$. As a consequence $p_{i-2} p_{i-1}$ is the only in-arc to $p_{i-1}$. This contradicts the fact that $D$ is a reduced instance because the arc $p_{i-2} p_{i-1}$ disconnects $p_{i-1}$ and $p_i$ from the root $r$ implying that Rule 3 can be applied. Thus $F$ is a forest and since every vertex in this forest except for vertices in $S$ have in-degree 1, we have that every tree of $F$ is rooted in some vertex of $S$. This completes the proof that $F$ satisfies properties (a)–(c), and thus is a nice forest. ∎

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 $F$.*

PROOF. By the proof of Corollary 4.2, there is an $r$-out-tree $T_S$ with $(V(T_S) \cap V(P)) \subseteq (S \cup \{p_l\})$ and $(A(T_S) \cap A(P)) = \emptyset$, such that all vertices in $S$ are leaves of $T_S$. We build an $r$-out-tree $T_F = (V(T_S) \cup V(P), A(T_S) \cup A(F))$. Notice that every leaf of $F$ is a leaf of $T_F$. Since $D$ is a NO-instance, we have that $T_F$ has at most $k - 1$ leaves and $k - 2$ vertices with out-degree at least 2. Thus, $F$ has at most $k - 1$ leaves and at most $k - 2$ vertices with out-degree at least 2. Hence, the number of vertices in $F$ whose parent in $F$ has out-degree at least 2 is at most $2k - 2$. Finally, by Corollary 4.2, $|S| \leq k$. Adding up these upper bounds yields that there are at most $k - 1 + k - 2 + 2k - 2 + k = 5(k - 1)$ 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. First, however, we need some more observations about the interplay between $P$ and $F$.

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

PROOF. As $T$ is a BFS-tree, it has no forward arcs. So, any vertex set $X = \{p_1, p_2, \dots, p_q\}$ with $q < |V(P)|$, the arc $p_q p_{q+1}$ is the only arc in $D$ from a vertex in $X$ to a vertex in $V(P) \setminus X$. ∎

COROLLARY 4.7. *No arc $p_ip_{i+1}$ is a forward arc of $F$.*

PROOF. If $p_ip_{i+1}$ is a forward arc of $F$, then there is a path from $p_i$ to $p_{i+1}$ in $F$. By Observation 1, $p_ip_{i+1}$ is the unique path from $p_i$ to $p_{i+1}$ in $D[V(P)]$. Hence $p_ip_{i+1} \in A(F)$ contradicting the fact that it is a forward arc. ∎

OBSERVATION 2. *Let $p_tp_j$ be an arc in $A(F)$ such that neither $p_t$ nor $p_j$ are key vertices, and $t \in \{j-1, j+1, \ldots, l\}$. Then for all $q > t$, $p_qp_j \notin A(D)$.*

Observation 2 follows directly from the definitions of a nice forest and key vertices.

OBSERVATION 3. *If neither $p_i$ nor $p_{i+1}$ are key vertices, then either $p_ip_{i+1} \notin A(F)$ or $p_{i+1}p_{i+2} \notin A(F)$.*

PROOF. Assume for contradiction that $p_ip_{i+1} \in A(F)$ and $p_{i+1}p_{i+2} \in A(F)$. Since neither $p_i$ nor $p_{i+1}$ are key vertices, both $p_{i+1}$ and $p_{i+2}$ must have in-degree 1 in $D$. Then the arc $p_ip_{i+1}$ disconnects both $p_{i+1}$ and $p_{i+2}$ from the root $r$ and Rule 3 can be applied, contradicting the fact that $D$ is a reduced instance. ∎

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

LEMMA 4.8. *$V(P')$ induces a directed path in $F$.*

PROOF. We first prove that for any arc $p_ip_{i+1} \in A(P')$ such that $p_ip_{i+1} \notin A(F)$, there is a path from $p_{i+1}$ to $p_i$ in $F$. Suppose for contradiction that there is no path from $p_{i+1}$ to $p_i$ in $F$, and let $x$ be the parent of $p_{i+1}$ in $F$. Then $p_ip_{i+1}$ is not a backward arc of $F$ and hence, $F' = (F \setminus xp_{i+1}) \cup \{p_ip_{i+1}\}$ is a forest of out-trees rooted at vertices in $S$. Also, since $p_{i+1}$ is not a key vertex, $x$ has out-degree 1 in $F$ and thus $x$ is a leaf in $F'$. Since $p_i$ is not a leaf in $F$, $F'$ has one more leaf than $F$. Now, every vertex with out-degree at least 2 in $F$ has out-degree at least 2 in $F'$. Additionally, $p_i$ has out-degree 2 in $F'$. Hence $F'$ is a nice forest of $P$ with more leaves than $F$, contradicting the choice of $F$.

Now, notice that by Observation 1, any path in $D[V(P)]$ from a vertex $u \in V(P')$ to a vertex $v \in V(P')$ that contains a vertex $w \notin V(P')$ must contain either the arc $p_{x-1}p_x$ or the arc $p_yp_{y+1}$. Since neither of those two arcs are arcs of $F$, it follows that for any arc $p_ip_{i+1} \in A(P')$ such that $p_ip_{i+1} \notin A(F)$, there is a path from $p_{i+1}$ to $p_i$ in $F[V(P')]$. Hence $F[V(P')]$ is weakly connected, that is, the underlying undirected graph is connected. Since every vertex in $V(P')$ has in-degree 1 and out-degree 1 in $F$, we conclude that $F[V(P')]$ is a directed path. ∎

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

OBSERVATION 4. *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 $Q'$.*

PROOF. Suppose for contradiction that $p_i$ appears before $p_j$ in $Q'$. By Observation 1, $p_ip_{i+1}p_{i+2}\ldots p_j$ is the unique path from $p_i$ to $p_j$ in $D[V(P')]$. This path contains both the arc $p_ip_{i+1}$ and $p_{i+1}p_{i+2}$ contradicting Observation 3. ∎

LEMMA 4.9. *All arcs of $D[V(P')]$ are contained in $A(P') \cup A(F)$.*

PROOF. Since $P$ has no forward arcs, it is enough to prove that any arc $p_jp_i \in A(D[V(P')])$ with $i < j$ is an arc of $F$. Suppose this is not the case and let $p_q$ be the parent of $p_i$ in $F$. We know that $p_i$ has in-degree at least 2 in $D$ and also since $p_i$ is not a key vertex $p_q$ has in-degree one in $F$. Hence by definition of $F$ being a nice forest, we have that for every $t > q$, $p_tp_i \notin A(D)$. It follows that $i < j < q$. By Lemma 4.8, $F[V(P')]$

is a directed path $Q'$ containing both $p_i$ and $p_j$. If $p_j$ appears after $p_i$ in $Q'$, Observation 4 implies that $i = j - 1$ and that $p_j$ has in-degree 1 in $D$ since $F$ is a nice forest. Thus, $p_i$ separates $p_j$ from the root and Rule 2 can be applied to $p_jp_i$ contradicting the fact that $D$ is a reduced instance. Hence, $p_j$ appears before $p_i$ in $Q'$.

Since $p_j$ is an ancestor of $p_i$ in $F$ and $p_q$ is the parent of $p_i$ in $F$, $p_j$ is an ancestor of $p_q$ in $F$ and hence $p_q \in V(Q') = V(P')$. Now, $p_j$ comes before $p_q$ in $Q'$ and $j < q$; so, Observation 4 implies that $q = j + 1$ and that $p_q$ has in-degree 1 in $D$ since $F$ is a nice forest. Thus, $p_j$ separates $p_q$ from the root $r$ and both $p_jp_i$ and $p_qp_i$ are arcs of $D$. Hence, Rule 4 can be applied to remove the arc $p_qp_i$, contradicting the fact that $D$ is a reduced instance. ∎

LEMMA 4.10. *If $|P'| \geq 3$, then there are exactly 2 vertices in $P'$ that are endpoints of arcs starting outside of $P'$.*

PROOF. By Observation 1, $p_{x-1}p_x$ is the only arc between $\{p_1, p_2, \ldots, p_{x-1}\}$ and $P'$. By Lemma 4.8, $F[V(P')]$ is a directed path $Q'$. Let $p_q$ be the first vertex on $Q'$ and notice that the parent of $p_q$ in $F$ is outside of $V(P')$. Observation 4 implies that $q \geq y - 1$. Hence $p_q$ and $p_x$ are two distinct vertices that are endpoints of arcs starting outside of $P'$. It remains to prove that they are the only such vertices. Let $p_i$ be any vertex in $P' \setminus \{p_x, p_q\}$. By Lemma 4.8, $V(P')$ induces a directed path $Q'$ in $F$, and since $p_q$ is the first vertex of $Q'$, the parent of $p_i$ in $F$ is in $V(P')$. Observation 2 yields then that $p_tp_i \notin A(D)$ for any $t > y$. ∎

OBSERVATION 5. *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 $D[V(P')]$.*

PROOF. By Lemma 4.8, $Q'$ is a directed path $f_1f_2 \ldots f_{|P'|}$ and let $Q'[f_1f_i]$ be the path $f_1f_2 \ldots f_i$. We prove that for any $i < |Q'|$, $f_if_{i+1}$ is the only arc from $V(Q'[f_1f_i])$ to $V(Q'[f_{i+1}f_{|P'|}])$. By Lemma 4.9, all arcs of $D[V(P')]$ are either arcs of $P'$ or arcs of $Q'$. Since $Q'$ is a path, $f_if_{i+1}$ is the only arc from $V(Q'[f_1f_i])$ to $V(Q'[f_{i+1}f_{|P'|}])$ in $Q'$. By Corollary 4.7, there are no arcs from $V(Q'[f_1f_i])$ to $V(Q'[f_{i+1}f_{|P'|}])$ in $P'$, except possibly for $f_if_{i+1}$. ∎

LEMMA 4.11. *For any vertex $x \notin V(P')$, there are at most 2 vertices in $P'$ with arcs to $x$.*

PROOF. Suppose there are 3 vertices $p_a, p_b, p_c$ in $V(P')$ such that $a < b < c$ and such that $p_ax, p_bx, p_cx \in A(D)$. By Lemma 4.8, $Q' = F[V(P')]$ is a directed path. If $p_a$ appears before $p_b$ in $Q'$ then Observation 4 implies that $a + 1 = b$ and that $p_b$ has in-degree 1 in $D$. Then, $p_a$ separates $p_b$ from the root and hence Rule 4 can be applied to remove the arc $p_bx$ contradicting the fact that $D$ is a reduced instance. Hence, $p_b$ appears before $p_a$ in $Q'$. By an identical argument, $p_c$ appears before $p_b$ in $Q'$.

Let $P_b$ be a path in $D$ from the root to $p_b$ and let $u$ be the last vertex in $P_b$ outside of $V(P')$. Let $v$ be the vertex in $P_b$ after $u$. By Lemma 4.10, $v$ is either $p_x$ or the first vertex $p_q$ of $Q'$. If $v = p_x$, then Observation 1 implies that $P_b$ contains $p_a$, whereas if $v = p_q$, by Observation 5, $P_b$ contains $p_c$. Thus the set $\{p_a, p_c\}$ separates $p_b$ from the root and hence Rule 4 can be applied to remove the arc $p_bx$ contradicting the fact that $D$ is a reduced instance. ∎

COROLLARY 4.12. *There are at most $14(k - 1)$ vertices in $P'$ with out-neighbors outside of $P'$.*

PROOF. By Lemma 4.3, there are at most $7(k-1)$ vertices that are endpoints of arcs originating in $P'$. By Lemma 4.11, each such vertex is the endpoint of at most two arcs from vertices in $P'$. ∎

LEMMA 4.13. $|P'| \leq 154(k-1)+10$.

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 4.12, $|X| \leq 14(k-1)$. Removing vertex set $X$ leave $14(k-1)+1$ paths and the union of these paths contains at least $154(k-1)+10-14(k-1)$ vertices. Hence, one of these paths contains at least $11$ vertices. By Observation 3, 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 4.8, $F[V(P'')]$ is a directed path $Q''$. Let $p_q$ and $p_t$ be the first and last vertices of $Q''$, respectively. By Lemma 4.10, $p_a$ and $p_q$ are the only vertices with in-arcs from outside of $P''$. By Observation 4, $p_q \in \{p_{b-1}, p_b\}$ 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 4.9 implies that $p_b$ and $p_t$ are the only vertices of $P'$ with out-arcs to the outside of $P''$. By Lemma 1, the path $P''$ is the unique out-branching of $D[V(P'')]$ rooted at $p_a$. By Observation 5, the path $Q''$ is the unique out-branching of $D[V(P'')]$ rooted at $p_q$ and ending in $p_t$. By Observation 4, $p_{b-2}$ 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 4.14. *Let $D$ be a reduced* NO*-instance to* ROOTED $k$-LEAF OUT-BRANCHING. *Then* $|V(D)| = O(k^3)$. *More specifically,* $|V(D)| \leq 1540k^3$.

PROOF. Let $T$ be a BFS-tree of $D$. $T$ has at most $k-1$ leaves and at most $k-2$ inner vertices with out-degree at least $2$. The remaining vertices can be partitioned into at most $2k-3$ paths $P_1 \dots P_t$ with all vertices having out-degree $1$ in $T$. We prove that for every $q \in \{1, \dots, t\}$, $|P_q| = O(k^2)$. Let $F$ be a nice forest of $P_q$ with the maximum number of leaves. Remember that the last vertex of the path is not included in the nice forest definition. By Lemma 4.6, $F$ has at most $5(k-1)$ key vertices. Let $p_i$ and $p_j$ be consecutive key vertices of $F$ on $P_q$. By Observation 3, there is a path $P' = p_x p_{x+1} \dots p_y$ containing no key vertices, with $x \leq i+1$ and $y \geq j-1$, such that neither $p_{x-1}p_x$ nor $p_y p_{y+1}$ are arcs of $F$. By Lemma 4.13, $|P'| \leq 154(k-1)+10$ so $|P_q| \leq (5(k-1)+1)(154(k-1)+10)+3(5(k-1))$. Hence, $|V(D)| \leq 2k(5k(154(k-1)+10+3)) \leq 1540k^3 = O(k^3)$. ∎

Lemma 4.14 results in a polynomial kernel for the problem ROOTED $k$-LEAF OUT-BRANCHING as follows.

THEOREM 4.15. *Both* ROOTED $k$-LEAF OUT-BRANCHING *and* ROOTED $k$-LEAF OUT-TREE *admit a kernel with* $O(k^3)$ *vertices.*

PROOF. Let $D$ be the reduced instance of ROOTED $k$-LEAF OUT-BRANCHING obtained in polynomial time using Lemma 3.5. If the number of vertices of $D$ is more than $1540k^3$, then return YES. Otherwise, we have an instance with $O(k^3)$ vertices. The correctness of this step follows from Lemma 4.14 which shows that any reduced NO-instance to ROOTED $k$-LEAF OUT-BRANCHING has $O(k^3)$ vertices. The result for ROOTED $k$-LEAF OUT-TREE follows similarly. ∎

Theorem 4.15 implies the following result about Turing kernelization.

COROLLARY 4.16. *Both* $k$-LEAF OUT-BRANCHING *and* $k$-LEAF OUT-TREE *admit a Turing kernel with* $O(k^3)$ *vertices.*

PROOF. We only give the proof for $k$-LEAF OUT-BRANCHING. The proof for $k$-LEAF OUT-TREE is similar. Let $D$ be a digraph on $n$ vertices. We apply the kernelization algorithm for ROOTED $k$-LEAF OUT-BRANCHING described in Theorem 4.15 for every vertex of $D$ as a root, and obtain $n$ graphs with $O(k^3)$ vertices, such that at least one of them has a $k$-leaf out-branching if and only if $D$ does. Clearly, this algorithm runs in

time polynomial in $n$. Now using the $O(k^3)$-sized oracle for $k$-LEAF OUT-BRANCHING we can solve the problem in linear time. This shows that $k$-LEAF OUT-BRANCHING admits a Turing kernel with $O(k^3)$ vertices, concluding the proof. ∎

## 5. KERNELIZATION LOWER BOUNDS

In the last section we gave a polynomial 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. [2009] that states that any *compositional* parameterized problem does not have a polynomial kernel unless $coNP$ is in $NP/poly$.

Let us outline first the proof of the main result of this section that $k$-LEAF OUT-BRANCHING has no polynomial kernel. The proof is done in several steps. We start from using the framework of Bodlaender et al. [2009] to establish that $k$-LEAF OUT-TREE has no polynomial kernel unless $coNP \subseteq NP/poly$ (Theorem 5.3). The second step is to show that a polynomial kernel for $k$-LEAF OUT-BRANCHING would yield a polynomial-sized kernel for $k$-LEAF OUT-TREE. To obtain this goal, we take $n$ copies of the graph corresponding to different guesses of roots, and for each such graph run the kernelization algorithm from Theorem 4.15 for ROOTED $k$-LEAF OUT-TREE. Since decision versions of both problems, $k$-LEAF OUT-BRANCHING and ROOTED $k$-LEAF OUT-TREE, are NP-complete under Karp reduction, there is a $k^{O(1)}$-time algorithm mapping each of the $n$ instances of ROOTED $k$-LEAF OUT-TREE to an instance of $k$-LEAF OUT-BRANCHING of size $k^{O(1)}$. We want to compose these graphs in such a way that a $k$-leaf out-branching of the composition can be computed from the maximum of out-branchings of its summands. To obtain such composition, we have to work with graphs with specific properties. But then we must prove that $k$-LEAF OUT-BRANCHING remains NP-complete under Karp reduction on this special class of digraphs (Lemma 5.4).

Before we proceed with the proofs, we need the following definition.

*Definition* 5.1 (*Composition [Bodlaender et al. 2009]*). A composition algorithm for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that

— 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 [Bodlaender et al. 2009], which we need for our purpose.

THEOREM 5.2 ([BODLAENDER ET AL. 2009]). *Let $L$ be a compositional parameterized language whose unparameterized version $\widetilde{L}$ is* NP*-complete. Unless $coNP \subseteq NP/poly$, there is no polynomial kernel for $L$.*

THEOREM 5.3. $k$-LEAF OUT-TREE *has no polynomial kernel unless $coNP \subseteq NP/poly$.*

PROOF. The problem is NP-complete [Alon et al. 2007]. We prove that it is compositional and thus, Theorem 5.2 will imply the statement of the theorem. A simple com-

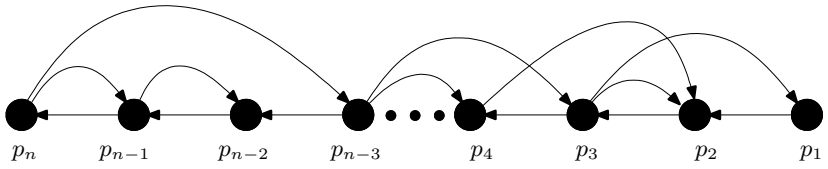


Fig. 3. Vertex $p_1$ is the bottom and vertex $p_n$ is the top of the willows. The willow is nice because $p_np_{n-1}$ and $p_np_{n-2}$ are arcs of $D$, these arcs are the only arcs of $A_2$ incident to $p_{n-1}$ or $p_{n-2}$, and arcs of $A_2$ give an out-branching of the graph.

position algorithm for this problem is as follows. On input $(D_1, k), (D_2, k), \ldots, (D_t, k)$ output the instance $(D, k)$ where $D$ is the disjoint union of $D_1, \ldots, D_t$. Since an out-tree must be completely contained in a connected component of the underlying undirected graph of $D$, $(D, k)$ is a YES-instance to $k$-LEAF OUT-TREE if and only if any out of $(D_i, k)$, $1 \leq i \leq t$, is. This concludes the proof. ∎

A *willow* graph [Drescher and Vetta 2010] $D = (V, A_1 \cup A_2)$ is a directed graph such that $D' = (V, A_1)$ is a directed path $P = p_1p_2 \ldots p_n$ 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 $p_np_{n-1}$ and $p_np_{n-2}$ 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. See Fig. 3 for an example of a nice willow graph.

OBSERVATION 6. *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 $p_n$.*

PROOF. Let $P = p_1p_2 \ldots p_n$ be the stem of $D$ and suppose for contradiction that there is an out-branching $T$ with the maximum number of leaves rooted at $p_i$, $i < n$. Since $D$ is a nice willow, $D' = (V, A_2)$ has a $p_n$-out-branching $T'$. Since every arc of $A_2$ is a backward arc of $P$, $T'[\{v_j : j \geq i\}]$ is a $p_n$-out-branching of $D[\{v_j : j \geq i\}]$. Then,

$$T'' = (V, \{v_xv_y \in A(T') : y \geq i\} \cup \{v_xv_y \in A(T) : y < i\})$$

is an out-branching of $D$. If $i = n - 1$, then $p_n$ is not a leaf of $T$, since the only arcs going out of the set $\{p_n, p_{n-1}\}$ start in $p_n$. Thus, in this case, all leaves of $T$ are leaves of $T''$ and $p_{n-1}$ is a leaf of $T''$ and not a leaf of $T$, contradicting the fact that $T$ has the maximum number of leaves. ∎

LEMMA 5.4. *$k$-LEAF OUT-TREE in nice willow graphs is NP-complete under Karp reductions.*

PROOF. We reduce from the well known NP-complete SET COVER problem [Karp 1972]. A *set cover* of a universe $U$ is a family $\mathcal{F}'$ of sets over $U$ such that every element of $u$ appears in some set in $\mathcal{F}'$. In the SET COVER problem, one is given a family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of sets over a universe $U$, $|U| = n$, together with a number $b \leq m$ and one is asked whether there is a set cover $\mathcal{F}' \subset \mathcal{F}$ with $|\mathcal{F}'| \leq b$ of $U$. In our reduction. we will assume that every element of $U$ is contained in at least one set in $\mathcal{F}$. We will also assume that $b \leq m - 2$. These assumptions are safe, because if either of them does not hold, the SET COVER instance can be resolved in polynomial time. From an instance of SET COVER, we build a digraph $D = (V, A_1 \cup A_2)$ as follows. The vertex set $V$ of $D$ is comprised of (1) a root $r$, (2) vertices $s_i$ for each $1 \leq i \leq m$ representing the sets in $\mathcal{F}$, (3) vertices $e_i$, $1 \leq i \leq n$ representing elements in $U$ and (4) two vertices $p$ and $p'$.

The arc set $A_2$ is defined as follows. There is an arc from $r$ to each vertex $s_i$, $1 \leq i \leq m$ and there is an arc from a vertex $s_i$ representing a set to a vertex $e_j$ representing an element if $e_j \in S_i$. Furthermore, $rp$ and $rp'$ are arcs in $A_2$. Finally, we let

$$A_1 = \{e_{i+1}e_i : 1 \leq i < n\} \cup \{s_{i+1}s_i : 1 \leq i < m\} \cup \{e_1 s_m, s_1 p, pp', p'r\}.$$

This concludes the description of $D$. We now proceed to prove that there is a set cover $\mathcal{F}' \subset \mathcal{F}$ with $|\mathcal{F}'| \leq b$ if and only if there is an out-branching in $D$ with at least $n + m + 2 - b$ leaves.

Suppose that there is a set cover $\mathcal{F}' \subset \mathcal{F}$ with $|\mathcal{F}'| \leq b$. We build a directed tree $T$ rooted at $r$ as follows. Every vertex $s_i$, $1 \leq i \leq m$, $p$ and $p'$ has $r$ as their parent. For every element $e_j$, $1 \leq i \leq n$, we choose the parent of $e_j$ to be $s_i$ such that $e_j \in S_i$ and $S_i \in \mathcal{F}'$ and for every $i' < i$, either $S_{i'} \notin |\mathcal{F}'|$ or $e_j \notin S_{i'}$. Since the only inner nodes of $T$ except for the root $r$ are vertices representing sets in the set cover, $T$ is an out-branching of $D$ with at least $n + m + 2 - b$ leaves.

In the other direction, suppose that there is an out-branching $T$ of $D$ with at least $n + m + 2 - b$ leaves, and suppose that $T$ has the most leaves out of all out-branchings of $D$. Since $D$ is a nice willow with $r$ as top vertex, Observation 6 implies that $T$ is an $r$-out-branching of $D$. Now, if there is an arc $e_{i+1}e_i \in A(T)$ then let $s_j$ be a vertex such that $e_i \in S_j$. Then, $T' = (T \setminus e_{i+1}e_i) \cup s_j e_i$ is an $r$-out-branching of $D$ with as many leaves as $T$. Hence, without loss of generality, for every $i$ between 1 and $n$, the parent of $e_i$ in $T$ is some $s_j$. Let $\mathcal{F}' = \{S_i : s_i \text{ is an inner vertex of } T\}$. $\mathcal{F}'$ is a set cover of $U$ with size at most $n + m + 2 - (n + m + 2 - b) = b$, concluding the proof. ∎

THEOREM 5.5. *$k$-LEAF OUT-BRANCHING has no polynomial kernel unless $coNP \subseteq NP/poly$.*

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 of $k$-LEAF OUT-TREE. For every vertex $v \in V$, we produce an instance $(D, v, k)$ of 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 for ROOTED $k$-LEAF OUT-TREE for some $v \in V$. By Theorem 4.15, 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 $(D_1, v_1, k)$, $(D_2, v_2, k)$, ..., $(D_n, v_n, k)$ with $|V(D_i)| = O(k^3)$ for each $i \leq n$. By Lemma 5.4, $k$-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 one 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 if and only if $(D', b_{max})$ is. Finally, 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 fixed constants $c_3$ and $c_4$. Hence, we conclude that if $k$-LEAF OUT-BRANCHING has a polynomial kernel, then so does $k$-LEAF OUT-TREE. Thus, Theorem 5.3 implies that $k$-LEAF OUT-BRANCHING has no polynomial kernel unless $coNP \subseteq NP/poly$. ∎

## 6. CONCLUSION AND DISCUSSIONS

In this paper, we demonstrated that Turing kernelization is a more powerful technique than Karp kernelization. We showed that while $k$-LEAF OUT-BRANCHING and $k$-LEAF OUT-TREE do not have a polynomial kernels, unless an unlikely collapse of complexity classes occurs, they do have $n$ independent polynomial kernels. Daligault and Thomassé [2009] have recently improved on the bounds of our 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.

— Which other problems admit a Turing kernelization like the cubic vertex kernels for $k$-LEAF OUT-BRANCHING and $k$-LEAF OUT-TREE obtained here? Is there a framework to rule out the possibility of $|I|^{O(1)}$ polynomial kernels similar to the framework developed in [Bodlaender et al. 2009]?
— Does there exist a problem for which we do not have a linear Karp kernel, but which does have linear kernels from the viewpoint of Turing kernelization?
— Can the recent results on lower bounds for kernels by Dell and van Melkebeek [2010] be used to prove a lower bound on sizes of kernels for the rooted $k$-Leaf Out-Branching?

## REFERENCES


ALON, N., FOMIN, F. V., GUTIN, G., KRILEVICH, M., AND SAURABH, S. 2009. Spanning directed trees with many leaves. *SIAM Journal of Discrete Mathematics 23*, 466–476.

ALON, N., FOMIN, F. V., GUTIN, G., KRIVELEVICH, M., AND SAURABH, S. 2007. Parameterized algorithms for directed maximum leaf problems. In *Automata, Languages and Programming, 34th International Colloquium, ICALP*, L. Arge, C. Cachin, T.Jurdzinski, and A. Tarlecki, Eds. LNCS Series, vol. 4596. Springer, 352–362.

BINKELE-RAIBLE, D. AND FERNAU, H. 2010. A faster exact algorithm for the directed maximum leaf spanning tree problem. In *Computer Science in Russia CSR*, F. Ablayev and E. W. Mayr, Eds. LNCS Series, vol. 6072. Springer, 328–339.

BODLAENDER, H. L. 2009. Kernelization: New upper and lower bound techniques. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC*, J. Chen and F. V. Fomin, Eds. LNCS Series, vol. 5917. Springer, 17–37.

BODLAENDER, H. L., DEMAINE, E. D., FELLOWS, M. R., GUO, J., HERMELIN, D., LOKSHTANOV, D., MLLER, M., RAMAN, V., VAN ROOIJ, J., AND ROSAMOND, F. A. 2008. Open problems in parameterized and exact computation – iwpec 2008. Tech. Rep. UU-CS-2008-017, Department of Informatics and Computing Sciences, Utrecht University.

BODLAENDER, H. L., DOWNEY, R. G., FELLOWS, M. R., AND HERMELIN, D. 2009. On problems without polynomial kernels. *Journal of Computer and System Sciences 75*, 423–434.

BODLAENDER, H. L., FOMIN, F. V., LOKSHTANOV, D., PENNINKX, E., SAURABH, S., AND THILIKOS, D. M. 2009. (Meta) kernelization. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*. IEEE Computer Society, 629–638.

BONSMA, P. S., BRUEGGEMANN, T., AND WOEGINGER, G. J. 2003. A faster FPT algorithm for finding spanning trees with many leaves. In *Mathematical Foundations of Computer Science 2003, MFCS*, B. Rovan and P. Vojtáś, Eds. LNCS Series, vol. 2747. Springer, 259–268.

BONSMA, P. S. AND DORN, F. 2008. Tight bounds and a fast FPT algorithm for Directed Max-Leaf Spanning Tree. In *Algorithms — ESA 2008, 16th Annual European Symposium*, D. Halperin and K. Mehlhorn, Eds. LNCS Series, vol. 5193. Springer, 222–233.

CHEN, J., FERNAU, H., KANJ, Y. A., AND XIA, G. 2007. Parametric duality and kernelization: lower bounds and upper bounds on kernel size. *SIAM Journal on Computing 37*, 1077–1108.

CHEN, J., KANJ, I. A., AND JIA, W. 2001. Vertex cover: further observations and further improvements. *Journal of Algorithms 41*, 280–301.

DALIGAULT, J., GUTIN, G., KIM, E. J., AND YEO, A. 2010. FPT algorithms and kernels for the directed $k$-leaf problem. *Journal of Computer and System Sciences 76,* 2, 144–152.

DALIGAULT, J. AND THOMASSÉ, S. 2009. On finding directed trees with many leaves. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC*, J. Chen and F. V. Fomin, Eds. LNCS Series, vol. 5917. Springer, 86–97.

DELL, H. AND VAN MELKEBEEK, D. 2010. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *ACM Symposium on Theory of Computing, STOC 2010*, L. J. Schulman, Ed. ACM, 251–260.

DING, G., JOHNSON, T., AND SEYMOUR, P. 2001. Spanning trees with many leaves. *Journal of Graph Theory*, 189–197.

DOWNEY, R. G. AND FELLOWS, M. R. 1999. *Parameterized Complexity*. Springer.

DRESCHER, M. AND VETTA, A. 2010. An approximation algorithm for the maximum leaf spanning arborescence problem. *ACM Trans. Algorithms 6,* 3, 1–18.

ESTIVILL-CASTRO, V., FELLOWS, M. R., LANGSTON, M. A., AND ROSAMOND, F. A. 2005. FPT is P-time extremal structure I. In *Algorithms and Complexity in Durham ACiD 2005*, H. Broersma, M. Johnson, and S. Szeider, Eds. Texts in Algorithmics Series, vol. 4. King's College Publications, 1–41.

FELLOWS, M. R. 2006. The lost continent of polynomial time: Preprocessing and kernelization. In *Parameterized and Exact Computation, Second International Workshop, IWPEC*, H. L. Bodlaender and M. A. Langston, Eds. LNCS Series, vol. 4169. Springer, 276–277.

FELLOWS, M. R., MCCARTIN, C., ROSAMOND, F. A., AND STEGE, U. 2000. Coordinatized kernels and catalytic reductions: an improved FPT algorithm for Max Leaf Spanning Tree and other problems. In *Foundations of Software Technology and Theoretical Computer Science, 20th Conference, FST TCS*, S. Kapoor and S. Prasad, Eds. LNCS Series, vol. 1974. Springer, 240–251.

FERNAU, H., FOMIN, F. V., LOKSHTANOV, D., RAIBLE, D., SAURABH, S., AND VILLANGER, Y. 2009. Kernel(s) for problems with no kernel: on out-trees with many leaves. In *Symposium on Theoretical Aspects of Computer Science STACS*, S. Albers and J.-Y. Marion, Eds. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Germany, 421–432.

FLUM, J. AND GROHE, M. 2006. *Parameterized Complexity Theory*. Text in Theoretical Computer Science. Springer.

FOMIN, F. V., GRANDONI, F., AND KRATSCH, D. 2008. Solving connected dominating set faster than $2^n$. *Algorithmica 52*, 2, 153–166.

FOMIN, F. V., LOKSHTANOV, D., SAURABH, S., AND THILIKOS, D. M. 2010. Bidimensionality and kernels. In *ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, M. Charikar, Ed. SIAM, 503–510.

GALBIATI, G., MAFFIOLI, F., AND MORZENTI, A. 1994. A short note on the approximability of the Maximum Leaf Spanning Tree problem. *Information Processing Letters 52*, 45–49.

GRIGGS, J. R., KLEITMAN, D. J., AND SHASTRI, A. 1989. Spanning trees with many leaves in cubic graphs. *13*, 669–695.

GRIGGS, J. R. AND WU, M. 1992. Spanning trees in graphs of minimum degree 4 or 5. *Discrete Mathematics*.

GUO, J. AND NIEDERMEIER, R. 2007. Invitation to data reduction and problem kernelization. *SIGACT News 38,* 1, 31–45.

KARP, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum Press, 85–103.

KLEITMAN, D. J. AND WEST, D. B. 1991. Spanning trees with many leaves. *SIAM Journal of Discrete Mathematics 4,* 1, 99–106.

KNEIS, J., LANGER, A., AND ROSSMANITH, P. 2008. A new algorithm for finding trees with many leaves. In *Algorithms and Computation, ISAAC*, S.-H. Hong, H. Nagamochi, and T. Fukunaga, Eds. LNCS Series, vol. 5369. Springer, 270–281.

LU, H.-I. AND RAVI, R. 1998. Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms 29*, 132–141.

NIEDERMEIER, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

RAIBLE, D. AND FERNAU, H. 2010. An amortized search tree analysis for $k$-LEAF SPANNING TREE. In *SOFSEM 2010: Theory and Practice of Computer Science*, J. van Leeuwen, A. Muscholl, D. Peleg, J. Pokorný, and B. Rumpe, Eds. LNCS Series, vol. 5901. Springer, 672–684.

SOLIS-OBA, R. 1998. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In *Algorithms—ESA '98, 6th Annual European Symposium*, G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, Eds. LNCS Series, vol. 1461. Springer, 441–452.

THOMASSÉ, S. 2010. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms 6,* 2.