A Faster Exact Algorithm for the Directed Maximum Leaf Spanning Tree Problem

Conference Paper · November 2009

DOI: 10.1007/978-3-642-13182-0_31 · Source: arXiv

2 authors:

Daniel Binkele-Raible
Itemis

Henning Fernau
Universität Trier

A Faster Exact Algorithm for the Directed Maximum Leaf Spanning Tree Problem

Daniel Raible & Henning Fernau

Univ.Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany
{raible,fernau}@informatik.uni-trier.de

Abstract. Given a directed graph $G = (V, A)$, the Directed Maximum Leaf Spanning Tree problem asks to compute a directed spanning tree (i.e., an out-branching) with as many leaves as possible. By designing a Branch-and-Reduced algorithm combined with the *Measure&Conquer* technique for running time analysis, we show that the problem can be solved in time $\mathcal{O}^*(1.9043^n)$ using polynomial space. Hitherto, there have been only few examples. Provided exponential space this run time upper bound can be lowered to $\mathcal{O}^*(1.8139^n)$.

1 Introduction

We investigate the following problem Directed Maximum Leaf Spanning Tree (DMLST)
Given: A directed graph $G(V, A)$.
Task: Find a directed spanning tree for G with the maximum number of leaves.

Alternatively, we can find an out-branching with the maximum number of leaves. Here an out-branching in a directed graph is a spanning tree T in the underlying undirected graph, but the arcs are directed from the root to the leaves, which are the vertices of out-degree zero with respect to T. The terms out-branching and directed spanning tree are equivalent.

1.1 Known Results.

The undirected version of the problem already has been widely studied with regard to its approximability. There is a 2-approximation running in polynomial time by R. Solis-Oba [13]. In almost linear time H.-I. Lu and R. Ravi [10] provide a 3-approximation. P.S. Bonsma and F. Zickfeld [2] could show that the problem is $\frac{3}{2}$-approximable when the input is restricted to cubic graphs. J. Daligault and S. Thomassé [4] described a 92-approximation algorithm together with an $\mathcal{O}(k^2)$-kernel for the Directed Maximum Leaf Spanning Tree problem.
This problem has also drawn notable attention in the field of parameterized algorithms. Here the problem is known as *directed k-leaf spanning tree* where k is a lower bound on the number of leaves in the directed spanning tree. The algorithm of J. Kneis, A. Langer and P. Rossmanith [8] solves this problem in

time $\mathcal{O}^*(4^k)$[1]. Moreover, in J. Daligault *et al.* [3] an upper-bound of $\mathcal{O}^*(3.72^k)$ is achieved. The same authors could also analyze their algorithm with respect to the input size n. This implies a running time upper bound of $\mathcal{O}^*(1.9973^n)$. D. Raible and H. Fernau [11] improved this running time to $\mathcal{O}^*(3.4575^k)$ in the more special case of undirected graphs.
F.V. Fomin, F. Grandoni and D. Kratsch [6] gave an exact, non-parameterized algorithm with run time $\mathcal{O}^*(1.9407^n)$ for the undirected version. H. Fernau *et al.* [5] improved this upper bound to $\mathcal{O}^*(1.8966^n)$. I. Koutis and R. Williams [9] could derive a randomized $\mathcal{O}^*(2^k)$-algorithm for the undirected version. Using an observation of V. Raman and S. Saurabh [12] this implies a randomized algorithm with running time $\mathcal{O}^*(1.7088^n)$.

1.2 Our Achievements.

The main result in this paper improves the current best upper of$\mathcal{O}^*(1.9973^n)$ by [3]. We can achieve a new bound of $\mathcal{O}^*(1.9043^n)$. Our algorithm is inspired by the one of [5]. However, this algorithm cannot be simply transferred to the directed version. Starting from an initial root the algorithm grows a tree T. The branching process takes place by deciding whether the vertices neighbored to the tree will become final leaves or internal vertices. A crucial ingredient of the algorithm was also to create *floating leaves*, i.e., vertices which are final leaves in the future solution but still have to be attached to the T, the tree which is grown. This concept has been already used in [5] and partly by [3]. In the undirected case we guarantee that in the bottleneck case we can generate at least two such leaves. In the directed version there is a situation where only one can be created. Especially for this problem we had to find a workaround.

1.3 Preliminaries, Terminology & Notation

We consider directed graphs $G(V, A)$ in the course of our algorithm, where V is the vertex set and A the arc set. The *in-neighborhood* of a vertex $v \in V$ is $N^-_{V'}(v) = \{u \in V' \mid (u, v) \in A\}$ and, analogously, its *out-neighborhood* is $N^+_{V'}(v) := \{u \in V' \mid (v, u)\}$. The *in- and out-degrees* of v are $d^-_{V'}(v) := |N^-_{V'}(v)|$ and $d^+_{V'}(v) := |N^+_{V'}(v)|$ and its *degree* is $d_{V'}(v) = d^-_{V'}(v) + d^+_{V'}(v)$. If $V' = V$ then we might suppress the subscript. For $V' \subseteq V$ we let $N^+(V) := \bigcup_{v \in V'} N^+(v)$ and $N^-(V')$ is defined analogously.
Let $A(V') := \{(u, v) \in A \mid \exists u, v \in V'\}$, $N^+_A(v) := \{(v, u) \in A \mid u \in N^+_V(v)\}$ and $N^-_A(v) := \{(u, v) \in A \mid u \in N^-_V(v)\}$. Given a graph $G = (V, A)$ and a graph $G' = (V', A')$, G' is a *subgraph* of G if $V' \subseteq V$ and $A' \subseteq A$. The subgraph of G induced by a vertex set $X \subseteq V$ is denoted by $G(X)$ and is defined by $G(X) = (X, A')$ where $A' = A(X)$. The *subgraph* of G induced by an arc set $Y \subseteq A$ is denoted by $G(Y)$ and is defined by $G(Y) = (\tilde{V}, V(Y))$ where $V(Y) = \{u \in V \mid \exists(u, v) \in Y \vee \exists(v, u) \in Y\}$.
A *directed path* of length ℓ in G is a set of pairwise different vertices $v_1, \dots, v_\ell$

[1] The notation $\mathcal{O}^*()$ suppresses polynomial factors.

such that $(v_i, v_{i+1}) \in A$ for $1 \leq i < \ell$. A subgraph $H(V_H, A_H)$ of G is called a *directed tree* if there is a unique root $r \in V_H$ such that there is a unique directed path P from r to every $v \in V_H \setminus \{r\}$ under the restriction that its arc set obeys $A(P) \subseteq A_H$. Speaking figuratively, in a directed tree the arcs are directed from the parent to the child. If for a directed tree $H = (V_H, A_H)$ that is a subgraph of $G(V, A)$ we have $V = V_H$ we call it *spanning directed tree* of G. The terms *out-tree* and *out-branching* are sometimes used for directed tree and spanning directed tree, respectively. The *leaves* of a directed tree $H = (V_H, A_H)$ are the vertices u such that $d^-_{V_H}(u) = d_{V_H}(u) = 1$. In $leaves(H)$ all leaves of a tree H are comprised and $internal(H) := V(H) \setminus leaves(H)$. The unique vertex v such that $N^-_{V_H}(u) = \{v\}$ for a tree-vertex will be called *parent* of u. A vertex $v \in V_H$ such that $d_{V_H}(v) \geq 2$ will be called *internal*. Let $T(V_T, A_T)$ and $T'(V_{T'}, A_{T'})$ be two trees. T' *extends* T, written $T' \succeq T$, iff $V_T \subseteq V_{T'}$, $A_T \subseteq A_{T'}$. Simplistically, we will consider a tree T also as a set of arcs $T \subseteq A$ such that $G(T)$ is a directed tree. The notions of $\succeq$ and $leaves(T)$ carry over canonically.
An *arc-cut set* is a set of arcs $B \subset A$ such that $G(A \setminus B)$ is a digraph which is not connected. We suppose that $|V| \geq 2$. The function $\chi()$ returns 1 if its argument evaluates to true and 0 otherwise.

1.4 Basic Idea of the Algorithm

First we formally re-define our problem:
Rooted Directed Maximum Leaf Spanning Tree (RDMLST)
Given: A directed graph $G(V, A)$ and a vertex $r \in V$.
Task: Find a spanning directed tree $T' \subseteq A$ such that $|leaves(T')|$ is maximum and $d^-_T(r) = 0$.

Once we have an algorithm for RDMLST it is easy to see that it can be used to solve DMLST. As a initial step we will consider every vertex as a possible root r of the final solution. This yields a total of n cases.
Then in the course of the algorithm for RDMLST we will gradually extend a out-tree $T \subseteq A$, which is predetermined to be a subgraph in the final out-branching. Let $V_T := V(T)$ and $\overline{V}_T := V \setminus V_T$. We will also maintain a mapping $lab : V \to \{\text{free}, \text{IN}, \text{LN}, \text{BN}, \text{FL}\} =: D$, which assigns different roles to the vertices. If $lab(v) = \text{IN}$ then v is already fixed to be internal, if $lab(v) = \text{LN}$ then it will be a leaf. If $lab(v) = \text{BN}$ then v already has a parent in T, but can be leaf or internal in the final solution. In general we will decide this by branching on such BN-vertices. If $lab(v) = \text{FL}$ then v is constrained to be a leaf but has not yet been attached to the tree T. Such vertices are called *floating leaves*. If $lab(v) = \text{free}$ then $v \notin V_T$ and nothing has been fixed or v yet. For a *label* $Z \in D$ and $v \in V$ we will often write $v \in Z$ when we mean $lab(v) = Z$. Vertices in IN or LN will also be called *internal nodes* or *leaf nodes*, respectively. A given tree T' defines a labeling $V_{T'} \to D$ to which we refer by $lab_{T'}$.. Let $\text{IN}_{T'} := \{v \in V_{T'} \mid d^+_{T'}(v) \geq 1\}$, $\text{LN}_{T'} := \{v \in V_{T'} \mid d^+_{T'}(v) = 0\}$ and $\text{BN}_{T'} = V_{T'} \setminus (\text{IN}_{T'} \cup \text{LN}_{T'})$. Then for any $ID \in D \setminus \{\text{FL}, \text{free}\}$ we have $ID_{T'} = lab^{-1}(ID)$. We always assure that lab_T and lab are the same on V_T. The subscript might be hence suppressed if $T' = T$.

If $T' \succ T$, then we assume that $\mathrm{IN}_T \subseteq \mathrm{IN}_{T'}$ and $\mathrm{LN}_T \subseteq \mathrm{LN}_{T'}$. So, the labels IN and LN remain once they are fixed. For the remaining labels we have the following possible transitions: FL $\to$ LN, BN $\to$ {LN, IN} and free $\to D \setminus \{\text{free}\}$. Let $\mathrm{BN}_i = \{v \in \mathrm{BN} \mid d^+(v) = i\}$, $\mathrm{free}_i = \{v \in \mathrm{free} \mid d^-(v) = i\}$ for $i \geq 1$, $\mathrm{BN}_{\geq \ell} := \cup_{j=\ell}^{n} \mathrm{BN}_j$ and $\mathrm{free}_{\geq \ell} := \cup_{j=\ell}^{n} \mathrm{free}_j$.

2 The Polynomial Part

2.1 Halting Rules

First we specify halting rules. If one of these rules applies the algorithm halts. Then it either returns a solution or answers that none can be built in the according branch of the search tree.

(H1) If there exists a $v \in \mathrm{free} \cup \mathrm{FL}$ with $d^-(v) = 0$. Halt and answer NO.
(H2) If $\mathrm{BN} = \emptyset$. Halt. A spanning tree has been constructed if $\mathrm{free} \cup \mathrm{FL} = \emptyset$. If so return $|\mathrm{LN}|$.
(H3) If there is a bridge $e := (u, v) \in A \setminus T$ which splits the graph in at least two connected components of size at least two and $v \in \mathrm{FL}$. Halt and answer NO.

2.2 Reduction rules

We state a set of six reduction rules in the following. Similar reduction rules for the undirected version have already appeared in [5,11]. We assume that the halting rules are already checked exhaustively

(R1) Let $v \in V$. If $lab(v) = \mathrm{FL}$ then remove $N_A^+(v)$. If $lab(v) = \mathrm{BN}$ then remove $N_A^-(v) \setminus T$.
(R2) If there exists a vertex $v \in \mathrm{BN}$ with $d^+(v) = 0$ then set $\mathrm{lab}(v) := \mathrm{LN}$.
(R3) If there exists a vertex $v \in \mathrm{free}$ with $d(v) = 1$ then set $\mathrm{lab}(v) := \mathrm{FL}$.
(R4) If $v \in \mathrm{LN}$ then remove $N_A(v) \setminus T$.
(R5) Let $u \in \mathrm{BN}$ such that $N_A^+(u)$ is a an arc-cut set. Then $\mathrm{lab}(u) := \mathrm{IN}$ and for all $x \in N^+(u) \cap \mathrm{FL}$ set $\mathrm{lab}(x) := \mathrm{LN}$, and for all $x \in N^+(u) \cap \mathrm{free}$ set $\mathrm{lab}(x) := \mathrm{BN}$.
(R6) If there is an arc $(a, b) \in A$ with $a, b \in \mathrm{free}$ and $G(A \setminus \{u, v\})$ consist of two strongly connected components of vertex-size greater than one. Then contract (a, b).

Proposition 1. *The reduction rules are sound.*

Proof. **(R1)** A floating leaf v cannot be a parent anymore. Thus, it is valid to remove $N_A^+(v)$. If $v \in \mathrm{BN}$ then v already has a parent in T. Thus, no arc in $N^-(v) \setminus T$ will ever be part of a tree $T' \succeq T$.
(R2) The vertex v cannot be a parent anymore. Thus, setting $lab(v) := \mathrm{LN}$ is sound.
(R3) The vertex v must be a leaf in any tree $T' \succeq T$.

(R4) The only arcs present in any tree $T' \succeq T$ will be $N_A(v) \cap T$. Thus, $N_A(v) \setminus T$ can be removed.

(R5) As $N_A^+(v)$ is an arc-cut set, setting $v \in$ LN would cut off a component which cannot be reached from the root r. Thus, $v \in$ IN is constrained.

(R6) Let G^* be the graph after contracting (h, u). If G^* has a spanning tree with k leaves, then also G. On the other hand note that in every spanning tree $T' \succeq T$ for G we have that $h, u \in$ IN and $(h, u) \in T'$. Hence, the tree $T^\#$ evolved by contracting (h, u) in T' is a spanning tree with k leaves in G^*.

3 The Exponential Part

3.1 Branching rules

If $N^+(internal(T)) \subseteq internal(T) \cup leaves(T)$, we call T an *inner-maximal* directed tree. We make use of the following fact:

Lemma 1 ([8] Lemma 4.2). *If there is a tree T' with $leaves(T') \geq k$ such that $T' \succeq T$ and $x \in internal(T')$ then there is a tree T'' with $leaves(T'') \geq k$ such that $T'' \succeq T$, $x \in internal(T'')$ and $\{(x, u) \in A\} \subseteq T''$*

See the Algorithm 1 which describes the branching rules. As mentioned before, the search tree evolves by branching on BN-vertices. For some $v \in$ BN we will set either $lab(v) =$ LN or $lab(v) =$ IN. In the second case we adjoin the vertices $N_A^+(v) \setminus T$ as BN-nodes to the partial spanning tree T. This is justified by Lemma 1. Thus, during the whole algorithm we only consider inner-maximal trees. Right in the beginning we therefore have $A(\{r\} \cup N^+(r))$ as a initial tree where r is the vertex chosen as the root.

We also introduce an abbreviating notation for the different cases generated by branching: $\langle v \in \text{LN}; v \in \text{IN}\rangle$ means that we recursively consider the two cases were v becomes a leaf node and an internal node. The semicolon works as a delimiter between the different cases. Of course, more complicated expression like $\langle v \in \text{BN}, x \in \text{BN}; v \in \text{IN}, x \in \text{LN}; v \in \text{LN}\rangle$ are possible, which generalize straight-forward.

3.2 Correctness of the algorithm

In the following we are going to prove a lemma which is crucial for the correctness and the running time.

Lemma 2. *Let $T \subseteq A$ be a given tree such that $v \in BN_T$ and $N^+(v) = \{x_1, x_2\}$. Let $T', T^* \subseteq A$ be optimal solutions with $T', T^* \succeq T$ under the restriction that $lab_{T'}(v) = LN$, and $lab_{T^*}(v) = IN$ and $lab_{T^*}(x_1) = lab_{T^*}(x_2) = LN$.*

1. *If there is a vertex $u \neq v$ with $N^+(u) = \{x_1, x_2\}$. Then $|leaves(T')| \geq |leaves(T^*)|$.*

Data: A directed graph $G = (V, A)$ and a tree $T \subseteq A$.
Result: A spanning tree T' with the maximum number of leaves such $T' \succeq T$
Check if a halting rule applies.
Apply the reduction rules exhaustively.
if $BN_1 \neq \emptyset$ **then**
    Choose some $v \in \mathrm{BN}_1$.
    Let $P = \{v_0, v_1, \ldots, v_k\}$ be a path of maximum length such that (1) $v_0 = v$, for all $1 \leq i \leq k-1$ (2) $d^+_{\overline{P_{i-1}}}(v_i) = 1$ (where $P_{i-1} = \{v_0, \ldots, v_{i-1}\}$) and (3) $P \setminus \mathrm{free} \subseteq \{v_0, v_k\}$
    if $d^+_{\overline{P_{i-1}}}(v_k) = 0$ **then**
        Put $v \in \mathrm{LN}$ (B1)
    else
        $\langle v \in \mathrm{IN}, v_1, \ldots, v_k \in \mathrm{IN}; v \in \mathrm{LN}\rangle$ (B2)
else
    Choose a vertex $v \in \mathrm{BN}$ with maximum out-degree.
    if *a)* $d^+(v) \geq 3$ ***or*** *b)*$(N^+(v) = \{x_1, x_2\}$ *and* $N^+(v) \subseteq FL)$ **then**
        $\langle v \in \mathrm{IN}; v \in \mathrm{LN}\rangle$ and in case *b)* apply `makeleaves`(x_1, x_2) in the 1st branch. (B3)
    else if $N^+(v) = \{x_1, x_2\}$ **then**
        if *for* $z \in (\{x_1, x_2\} \cap \mathit{free})$ *we have*
        $|N^+(z) \setminus N^+(v)| = 0$ ***or*** *(B4.1)*
        $N^+_A(z)$ *is an arc-cut set* **or** *(B4.2)*
        $N^+(z) \setminus N^+(v) = \{v_1\}$. *(B4.3)*
        then
            $\langle v \in \mathrm{IN}; v \in \mathrm{LN}\rangle$ (B4)
    else if $N^+(v) = \{x_1, x_2\}$, $x_1 \in \mathit{free}, x_2 \in FL$ **then**
        $\langle v \in \mathrm{IN}, x_1 \in \mathrm{IN}; v \in \mathrm{IN}, x_1 \in \mathrm{LN}; v \in \mathrm{LN}\rangle$ and apply `makeleaves`(x_1, x_2) in the 2nd branch. (B5)
    else if $N^+(v) = \{x_1, x_2\}$, $x_1, x_2 \in \mathit{free}$, $\exists z \in (N^-(x_1) \cap N^-(x_2)) \setminus \{v\}$ **then**
        $\langle v \in \mathrm{IN}, x_1 \in \mathrm{IN}; v \in \mathrm{IN}, x_1 \in \mathrm{LN}, x_2 \in \mathrm{IN}; v \in \mathrm{LN}\rangle$ (B6)
    else if $N^+(v) = \{x_1, x_2\}$, $x_1, x_2 \in \mathit{free}$, $|(N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}| \geq 2$ **then**
        $\langle v \in \mathrm{IN}, x_1 \in \mathrm{IN}; v \in \mathrm{IN}, x_1 \in \mathrm{LN}, x_2 \in \mathrm{IN}; v \in \mathrm{IN}, x_1 \in \mathrm{LN}, x_2 \in \mathrm{LN}; v \in \mathrm{LN}\rangle$ and apply `makeleaves`(x_1, x_2) in the 3rd branch. (B7)
    else
        $\langle v \in \mathrm{IN}; v \in \mathrm{LN}\rangle$ (B8)

Algorithm 1: An Algorithm for solving RDMLST

begin
    $\forall u \in [(N^-(x_1) \cup N^-(x_2)) \setminus \{x_1, x_2, v\}] \cap \mathrm{free}$ set $u \in \mathrm{FL}$;
    $\forall u \in [(N^-(x_1) \cup N^-(x_2)) \setminus \{x_1, x_2, v\}] \cap \mathrm{BN}$ set $u \in \mathrm{LN}$;
end

Procedure `makeleaves`(x_1, x_2)

2. *Assume that $d^-(x_i) \geq 2$ ($i = 1, 2$). Assume that there exists some $u \in (N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}$ such that $lab_{T^*}(u) = IN$. Then $|leaves(T')| \geq |leaves(T^*)|$.*

Proof. 1. Let $T^+ := (T^* \setminus \{(v, x_1), (v, x_2)\}) \cup \{(u, x_1), (u, x_2)\}$. We have $lab_{T^+}(v) =$ LN and u is the only vertex besides v where $lab_{T^*}(u) \neq lab_{T^+}(u)$ is possible. Hence, u is the only vertex where we could have $lab_{T^*}(u) =$ LN such that $lab_{T^+}(u) =$ IN. Thus, we can conclude $|leaves(T^+)| \geq |leaves(T^*)|$. As T' is optimal under the restriction that $v \in$ LN it follows $|leaves(T')| \geq |leaves(T^+)| \geq |leaves(T^*)|$.
2. W.l.o.g. we have $u \in N^-(x_1) \setminus \{v, x_2\}$. Let $q \in N^-(x_2) \setminus \{v\}$ and $T^+ := (T^* \setminus \{(v, x_1), (v, x_2)\}) \cup \{(u, x_1), (q, x_2)\}$. We have $lab_{T^+}(v) =$ LN, $lab_{T^+}(u) = lab_{T^*}(u) =$ IN and q is the only vertex besides v where we could have $lab_{T^*}(q) \neq lab_{T^+}(q)$ (i.e., $lab_{T^*}(q) =$ LN and $lab_{T^+}(q) =$ IN). Therefore $|leaves(T')| \geq |leaves(T^+)| \geq |leaves(T^*)|$.

□

Correctness of the Different Branching Cases First note that **(H2)** takes care of the case it indeed an out-branching has been built. If so the number of its leaves is returned.

Below we will argue that each branching case in Algorithm 1 is correct in a way that it preserves at least one optimal solution. Cases (B4) and (B8) do not have to be considered in detail as these are simple binary and exhaustive branchings.

(B1) Suppose there is an optimal extension $T' \succeq T$ such that $lab_{T'}(v) = lab_{T'}(v_0) =$ IN. Due to the structure of P there must be an i, $0 < i \leq k$ such that $(v_j, v_{j-1}) \in T'$ for $0 < j \leq i$, i.e., $v, v_1, \ldots v_{i-1} \in$ IN and $v_i \in$ LN. W.l.o.g., we choose T' in a way that i is minimum but T' is still optimal (✢). By **(R5)** there must be a vertex v_z, $0 < z \leq i$, such that there is an arc (q, v_z) with $q \notin P$. Now consider $T'' = (T' \setminus \{(v_{z-1}, v_z)\}) \cup \{q, v_z\}$. In T'' the vertex v_{z-1} is a leaf and therefore $|leaves(T'')| \geq |leaves(T')|$. Additionally, we have that $z - 1 < i$ which is a contradiction to the choice of T' (✢).

(B2) Note that $lab(v_k) \in \{\text{BN}, \text{FL}\}$ is not possible due to **(R1)** and, thus, $lab(v_k) =$ free. By the above arguments from **(B1)** we can exclude the case that $v, v_1, \ldots v_{i-1} \in$ IN and $v_i \in$ LN ($i \leq k$). Thus, under the restriction that we set $v \in$ IN, the only remaining possibility is also to set $v_1, \ldots v_k \in$ IN.

(B3) *b)* When we set $v \in$ IN then the two vertices in $N^+(v)$ will become leaf nodes (i.e., become part of LN). Thus, Lemma 2.2 applies (Note that **(R5)** does not apply and therefore $(N^+(x_1) \cup N^+(x_2)) \setminus \{v, x_1, x_2\} \neq \emptyset$). This means that that every vertex in $(N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}$ can be assumed a to be leaf node in the final solution. This justifies to apply `makeleaves`(x_1, x_2).

(B5) The branching is exhaustively with respect to v and x_1. Nevertheless, in the second branch `makeleaves`(x_1, x_2) is carried out. This is justified by Lemma 2.2 as by setting $v \in$ IN and $x_1 \in$ LN, x_2 will be attached to v as a LN-node and **(R5)** does not apply.

(B6) In this case we neglect the possibility that $v \in \mathrm{IN}, x_1, x_2 \in \mathrm{LN}$. But due to Lemma 2.1 a no worse solution can be found in the recursively considered case where we set $v \in \mathrm{LN}$. This shows that the considered cases are sufficient.

(B7) Similarly, as in case (B3) we can justify by Lemma 2.2 the application of $\texttt{makeleaves}(x_1, x_2)$ in the third branch.

Further branching cases will not be considered as their correctness is clear due to exhaustive branching.

3.3 Analysis of the Running Time

The Measure To analyze the running-time we follow the *Measure&Conquer*-approach (see [7]) and use the following measure:

$$\mu(G) = \sum_{i=1}^{n} \epsilon_i^{\mathrm{BN}} |\mathrm{BN}_i| + \sum_{i=1}^{n} \epsilon_i^{\mathrm{free}} |\mathrm{free}_i| + \epsilon^{\mathrm{FL}} |\mathrm{FL}|$$

The concrete values are $\epsilon^{\mathrm{FL}} = 0.2251$, $\epsilon_1^{\mathrm{BN}} = 0.6668$, $\epsilon_i^{\mathrm{BN}} = 0.7749$ for $i \geq 2$, $\epsilon_1^{\mathrm{free}} = 0.9762$ and $\epsilon_2^{\mathrm{free}} = 0.9935$. Also let $\epsilon_j^{\mathrm{free}} = 1$ for $j \geq 3$ and
$\eta = \min\{\epsilon^{\mathrm{FL}}, (1 - \epsilon_1^{\mathrm{BN}}), (1 - \epsilon_2^{\mathrm{BN}}), (\epsilon_2^{\mathrm{free}} - \epsilon_1^{\mathrm{BN}}), (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}), (\epsilon_1^{\mathrm{free}} - \epsilon_1^{\mathrm{BN}}), (\epsilon_1^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})\} = \epsilon_1^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}} = 0.2013$.

For $i \geq 2$ let $\Delta_i^{\mathrm{free}} = \epsilon_i^{\mathrm{free}} - \epsilon_{i-1}^{\mathrm{free}}$ and $\Delta_1^{\mathrm{free}} = \epsilon_1^{\mathrm{free}}$. Thus, $\Delta_{i+1}^{\mathrm{free}} \leq \Delta_i^{\mathrm{free}}$ with $\Delta_s^{\mathrm{free}} = 0$ for $s \geq 4$.

Run Time Analysis of the Different Branching Cases
In the following we state for every branching case by how much μ will be reduced. Especially, Δ_i states the amount by which the i-th branch decreases μ. If v is the vertex chosen by Algorithm 1 the it is true that for all $x \in N^+(v)$ we have $d^-(x) \geq 2$ by **(R5)** (✜).

(B2) $\langle v \in \mathrm{IN}, v_1, \ldots, v_k \in \mathrm{IN}, v \in \mathrm{LN}\rangle$
Recall that $d^+_{\overline{P_{k-1}}}(v_k) \geq 2$ and $v_k \in$ free by **(R1)**. Then we must have that $v_1 \in \mathrm{free}_{\geq 2}$ by **(R5)**.
1. v becomes IN-node; $v_1, \ldots, v_k$ become IN-nodes; the free vertices in $N^+(v_k)$ become BN-nodes, the floating leaves in $N^+(v_k)$ become LN-nodes:
$\Delta_1 = \epsilon_1^{\mathrm{BN}} + \sum_{i=2}^{k} \epsilon_1^{\mathrm{free}} + \chi(v_1 \in \mathrm{free}_2) \cdot \epsilon_2^{\mathrm{free}} + \chi(v_1 \in \mathrm{free}_{\geq 3}) \cdot \epsilon_3^{\mathrm{free}} + 2 \cdot \eta$
2. v becomes LN-node; the degree of v_1 is reduced:
$\Delta_2 = \epsilon_1^{\mathrm{BN}} + \sum_{i=2}^{3} \chi(v_1 \in \mathrm{free}_i) \cdot \Delta_i^{\mathrm{free}}$

(B3) $\langle v \in \mathrm{IN}; v \in \mathrm{LN}\rangle$.
Case *a)*
1. v becomes IN-node; the free out-neighbors of v become BN-nodes; the FL out-neighbors of v becomes LN-nodes:
$\Delta_1 = \epsilon_2^{\mathrm{BN}} + \sum_{x \in N^+(v) \cap \mathrm{free}_{\geq 3}} (1 - \epsilon_2^{\mathrm{BN}}) + \sum_{x \in N^+(v) \cap \mathrm{free}_2} (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}) + \sum_{y \in N^+(v) \cap \mathrm{FL}} \epsilon^{\mathrm{FL}}$

2. v becomes LN-node; the in-degree of the free out-neighbors of v is decreased; $\mathit{\Delta}_2 = \epsilon_2^{\mathrm{BN}} + \sum_{i=2}^{3} |N^+(v) \cap \mathrm{free}_i| \cdot \mathit{\Delta}_i^{\mathrm{free}}$

Case *b)*

Recall that v is a BN of maximum out-degree, thus $d^+(z) \leq d^+(v) = 2$ for all $z \in \mathrm{BN}$. On the other hand $\mathrm{BN}_1 = \emptyset$ which implies $\mathrm{BN} = \mathrm{BN}_2$ from this point on. Hence, we have $N^+(v) = \{x_1, x_2\}$, $d^-(x_i) \geq 2$, $(i = 1, 2)$ and $|(N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}| \geq 1$ by (✜), in the following branching cases. Therefore the additional amount of $min\{\epsilon_1^{\mathrm{free}} - \epsilon^{\mathrm{FL}}, \epsilon_2^{\mathrm{BN}}\}$ in the first branch is justified by the application of `makeleaves`(x_1, x_2). Note that by (✜) at least one free-node becomes a FL-node, or one BN-node becomes a LN-node. Also due to **(R1)** we have that $N^+(x_i) \cap \mathrm{BN} = \emptyset$.

1. v becomes IN-node; the FL out-neighbors of v become LN-nodes; the vertices in $[N^-(x_1) \cup N^-(x_2) \setminus \{v, x_1, x_2\}] \cap \mathrm{BN}$ become LN-nodes; the vertices in $[N^-(x_1) \cup N^-(x_2) \setminus \{v, x_1, x_2\}] \cap \mathrm{free}$ become FL-nodes.
   $\mathit{\Delta}_1 = \epsilon_2^{\mathrm{BN}} + 2 \cdot \epsilon^{\mathrm{FL}} + min\{\epsilon_1^{\mathrm{free}} - \epsilon^{\mathrm{FL}}, \epsilon_2^{\mathrm{BN}}\}$
2. v becomes LN; $\mathit{\Delta}_2 = \epsilon_2^{\mathrm{BN}}$.

(B4) $\langle v \in \mathrm{IN}; v \in \mathrm{LN} \rangle$.

(B4.1): 1. v becomes IN-node; z becomes LN-node by **(R1)**, **(R2)** or both **R4)**; The vertex $q \in \{x_1, x_2\} \setminus \{z\}$ becomes LN-node or BN-node (depending on $q \in \mathrm{FL}$ or $q \in \mathrm{free}$)
$\mathit{\Delta}_1 = \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{free}} + \min\{\epsilon^{\mathrm{FL}}, (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})\}$
2. z becomes LN-node;
$\mathit{\Delta}_2 = \epsilon_2^{\mathrm{BN}}$

(B4.2): 1. v becomes IN-node; (z, h) $N_A^+(z)$ is an arc-cut. Thus, z becomes IN-node as **(R5)** applies; The vertex $q \in \{x_1, x_2\} \setminus \{z\}$ becomes LN-node or BN-node (depending on $q \in \mathrm{FL}$ or $q \in \mathrm{free}$)
$\mathit{\Delta}_1 = \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{free}} + \min\{\epsilon^{\mathrm{FL}}, (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})\}$
2. z becomes LN-node;
$\mathit{\Delta}_2 = \epsilon_2^{\mathrm{BN}}$

Note that in all following branching cases we have $N^+(x_i) \cap \mathrm{free}_1 = \emptyset$ $(i = 1, 2)$ by this case.

(B4.3): We have $|N^+(z) \setminus N^+(v)| = 1$. Thus, in the next recursive call after the first branch and the exhaustive application of **(R1)**, either **(R6)**, case (B2) or (B1) applies. **(R5)** does not apply due to (B4.2) being ranked higher. Note that the application of any other reduction rule does not change the situation. If (B2) applies we can analyze the current case together with its succeeding one. If (B2) applies in the case we set $v \in \mathrm{IN}$ we deduce that $v_0, v_1, \ldots, v_k \in free$ where $z = v_0 = x_1$ (w.l.o.g., we assumed $z = x_1$). Observe that $v_1 \in \mathrm{free}_{\geq 2}$ as (B4.2) does not apply.

1. v becomes IN-node; x_1 becomes LN-node; x_2 becomes FL- or BN-node (depending on whether $x_2 \in \mathrm{free}$ or $x_2 \in \mathrm{FL}$; the degree of v_1 drops:

$$\mathit{\Delta}_{11} = \epsilon_2^{\mathrm{BN}} + \chi(x_1 \in \mathrm{free}_{\geq 3}) \cdot \epsilon_3^{\mathrm{free}} + \chi(x_1 \in \mathrm{free}_2) \cdot \epsilon_2^{\mathrm{free}} +$$

$\chi(x_2 \in \text{free}_{\geq 3}) \cdot (\epsilon_3^{\text{free}} - \epsilon_2^{\text{BN}}) + \chi(x_2 \in \text{free}_2) \cdot (\epsilon_2^{\text{free}} - \epsilon_2^{\text{BN}}) + \chi(x_2 \in \text{FL}) \cdot \epsilon^{\text{FL}} + \sum_{i=2}^{3} \chi(v_1 \in \text{free}_i) \cdot \Delta_i^{\text{free}}$

2. v becomes IN-node, $x_1, v_1 \in \text{IN}, \ldots, v_k$ become IN-nodes; the free vertices in $N^+(v_k)$ become BN-nodes, the floating leaves in $N^+(v_k)$ become LN-nodes:

   $\Delta_{12} = \epsilon_2^{\text{BN}} + \chi(x_1 \in \text{free}_{\geq 3}) \cdot \epsilon_3^{\text{free}} + \chi(x_1 \in \text{free}_2) \cdot \epsilon_2^{\text{free}} + \chi(x_2 \in \text{free}_{\geq 3}) \cdot (\epsilon_3^{\text{free}} - \epsilon_2^{\text{BN}}) + \chi(x_2 \in \text{free}_2) \cdot (\epsilon_2^{\text{free}} - \epsilon_2^{\text{BN}}) + \chi(x_2 \in \text{FL}) \cdot \epsilon^{\text{FL}} +$
   $\chi(v_1 \in \text{free}_2) \cdot \epsilon_2^{\text{free}} + \chi(v_1 \in \text{free}_{\geq 3}) \cdot \epsilon_3^{\text{free}} + \sum_{i=2}^{k} \epsilon_1^{\text{free}} + 2\eta$
3. v becomes LN-node: the degrees of x_1 and x_2 drop:

   $\Delta_2 = \epsilon_2^{\text{BN}} + \sum_{\ell=2}^{\max_{h \in \{1,2\}} d^-(x_h)} \sum_{j=1}^{2} \chi(x_j \in \text{free}_\ell) \cdot \Delta_\ell^{\text{free}}.$

If case (B1) applies to v_1 the reduction in both branches is as least as great as in (B4.1)/(B4.2).
If **(R6)** applies after the first branch (somewhere in the graph) we get $\Delta_1 = \epsilon_2^{\text{BN}} + (\epsilon_2^{\text{free}} - \epsilon_1^{\text{BN}}) + \epsilon_1^{\text{free}} + \min\{\epsilon^{\text{FL}}, (\epsilon_2^{\text{free}} - \epsilon_2^{\text{BN}})\}$ and $\Delta_2 = \epsilon_2^{\text{BN}}$. Here the amount of ϵ_1^{free} in Δ_1 originates from an **(R6)** application.

(B5) $\langle v \in \text{IN}, x_1 \in \text{IN}; v \in \text{IN}, x_1 \in \text{LN}; v \in \text{LN}\rangle$

1. v and x_1 become IN-nodes; x_2 becomes a FL-node; the vertices in $N^+(x_1) \cap$ free become BN-nodes; the vertices in $N^+(x_1) \cap \text{FL}$ become LN-nodes;

   $\Delta_1 = \epsilon_2^{\text{BN}} + \epsilon_2^{\text{free}} + \epsilon^{\text{FL}} + \sum_{x \in N^+(x_1) \cap \text{free}} (\epsilon_2^{\text{free}} - \epsilon_2^{\text{BN}}) + \sum_{x \in N^+(x_1) \cap \text{FL}} \epsilon^{\text{FL}}$

2. v becomes IN-node; x_1 becomes LN-node; x_2 becomes LN-node; after applying `makeleaves`(x_1, x_2) the vertices in $[N^-(x_1) \cup N^-(x_2) \setminus \{v, x_1, x_2\}] \cap \text{BN}$ become LN-nodes and the vertices in $[N^-(x_1) \cup N^-(x_2) \setminus \{v, x_1, x_2\}] \cap$ free become FL-nodes:

   $\Delta_2 = \epsilon_2^{\text{BN}} + \epsilon_2^{\text{free}} + \epsilon^{\text{FL}} + \min\{\epsilon_1^{\text{free}} - \epsilon^{\text{FL}}, \epsilon_2^{\text{BN}}\}$
3. v becomes LN: $\Delta_3 = \epsilon_2^{\text{BN}}$

The amount of $\min\{\epsilon^{\text{FL}}, (\epsilon_1^{\text{free}} - \epsilon_2^{\text{BN}})\}$ in the second branch is due to (✥).

(B6) $\langle v \in \text{IN}, x_1 \in \text{IN}; v \in \text{IN}, x_1 \in \text{LN}, x_2 \in \text{IN}; v \in \text{LN}\rangle$ The branching vector can be derived by considering items 1,2 and 4 of (B7) and the reductions Δ_1, Δ_2 and Δ_4 in μ obtained in each item.

(B7) $\langle v \in \text{IN}, x_1 \in \text{IN}; v \in \text{IN}, x_1 \in \text{LN}, x_2 \in \text{IN}; v \in IN, x_1 \in \text{LN}, x_2 \in \text{LN}; v \in \text{LN}\rangle$
Note that if $N_A^+(x_1)$ or $N_A^+(x_2)$ is an arc-cut set then (B4.2) applies. Thus, all the branching cases must be applicable.
Moreover due to the previous branching case (B4.3) we have $|N^+(x_1) \setminus N^+(v)| = |N^+(x_1) \setminus \{x_2\}| \geq 2$ and $|N^+(x_2) \setminus N^+(v)| = |N^+(x_2) \setminus \{x_1\}| \geq 2$ (✶).
Note that $N^-(x_1) \cap N^-(x_2) = \{v\}$ due to (B6).

For $i \in \{1, 2\}$ let $fl_i = |\{x \in N^+(x_i) \setminus N^+(v) \mid x \in FL\}|$, $fr_i^{\geq 3} = |\{u \in N^+(x_i) \setminus N^+(v) \mid u \in \text{free}_{\geq 3}\}|$ and $fr_i^2 = |\{u \in N^+(x_i) \setminus N^+(v) \mid u \in \text{free}_2\}|$. Observe that for $i \in \{1, 2\}$ we have $(fl_i + fr_i^{\geq 3} + fr_i^2) \geq 2$ due to (✶).

1. v becomes IN; x_1 becomes IN; x_2 becomes BN; the free out-neighbors of x_1 become BN; the FL out-neighbors of x_1 become LN;
   $\Delta_1 = \epsilon_2^{\mathrm{BN}} + \chi(x_1 \in \mathrm{free}_{\geq 3}) + \chi(x_1 \in \mathrm{free}_2) \cdot \epsilon_2^{\mathrm{free}} + \chi(x_2 \in \mathrm{free}_{\geq 3}) \cdot (\epsilon_3^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}) + \chi(x_2 \in \mathrm{free}_2) \cdot (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})$
   $+ (fl_1 \cdot \epsilon^{\mathrm{FL}} + fr_1^{\geq 3} \cdot (\epsilon_3^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}) + fr_1^2 \cdot (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}))$
2. v becomes IN; x_1 becomes LN; x_2 becomes IN; the free out-neighbors of x_2 becomes BN; the FL out-neighbors of x_2 become LN;
   $\Delta_2 = \epsilon_2^{\mathrm{BN}} + (\sum_{i=1}^{2}[\chi(x_i \in \mathrm{free}_{\geq 3}) \cdot \epsilon_3^{\mathrm{free}} + \chi(x_i \in \mathrm{free}_2) \cdot \epsilon_2^{\mathrm{free}}])$
   $+ (fl_2 \cdot \epsilon^{\mathrm{FL}} + fr_2^{\geq 3} \cdot (\epsilon_3^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}) + fr_2^2 \cdot (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}}))$
3. v becomes IN; x_1 becomes LN; x_2 becomes LN; the free in-neighbors of x_1 become FL; the BN in-neighbors of x_1 become LN; the free in-neighbors of x_2 become FL; the BN in-neighbors of x_2 become LN:

   $\Delta_3 = \epsilon_2^{\mathrm{BN}} + [\sum_{i=1}^{2}(\chi(x_i \in \mathrm{free}_{\geq 3}) \cdot \epsilon_3^{\mathrm{free}} + \chi(x_i \in \mathrm{free}_2) \cdot \epsilon_2^{\mathrm{free}})]$
   $+ \max\{2, (d^-(x_1) + d^-(x_2) - 4)\} \cdot \min\{\epsilon_1^{\mathrm{free}} - \epsilon^{\mathrm{FL}}, \epsilon_2^{\mathrm{BN}}\}$

   Note that the additional amount of $\max\{2, (d^-(x_1)+d^-(x_2)-4)\} \cdot \{\epsilon_2^{\mathrm{free}} - \epsilon^{\mathrm{FL}}, \epsilon_2^{\mathrm{BN}}\}$ is justified by Lemma 2.2 and by the fact that $d^-(x_i) \geq 2$ and $N^-(x_1) \cap N^-(x_2) = \{v\}$ due to (B6). Thus, we have $|N^-(x_1) \cup N^-(x_2) \setminus \{x_1, x_2, v\}| \geq \max\{2, (d^-(x_1) + d^-(x_2) - 4)\}$.
4. v becomes LN; the degrees of x_1 and x_2 drop:
   $\Delta_4 = \epsilon_2^{\mathrm{BN}} + \sum_{j=2}^{\max_{\ell \in \{1,2\}}\{d^-(x_\ell)\}} \sum_{i=1}^{2}(\chi(d^-(x_i) = j) \cdot \Delta_j^{\mathrm{free}})$

(B8) Observe that in the second branch we can apply **(R6)**. Due to the non-applicability of **(R5)** and the fact that (B7) is ranked higher in priority we have $|(N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}| = 1$. Especially, (B6) cannot be applied by which we derive that $N^-(x_1) \cap N^-(x_2) = \{v\}$. Thus, due to this we have the situation in Figure 1.
So, w.l.o.g, there are arcs $(q, x_1), (x_1, x_2) \in A$, where $\{q\} = (N^-(x_1) \cup N^-(x_2)) \setminus \{v, x_1, x_2\}$, because we can rely on $d^-(x_i) \geq 2$ $(i = 1, 2)$ by (✜).

1. Firstly, assume that $q \in$ free.
   (a) v becomes IN; x_1 and x_2 becomes BN:
   $\Delta_1 = \epsilon_2^{\mathrm{BN}} + 2 \cdot (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})$
   (b) The arc (q, x_1) will be contracted by **(R6)** when we v becomes LN, as x_1 and x_2 only can be reached by using (q, x_1):
   $\Delta_2 = \epsilon_2^{\mathrm{BN}} + \epsilon_1^{\mathrm{free}}$.
2. Secondly, assume $q \in$ BN. Then $q \in \mathrm{BN}_2$ due to the branching priorities.
   (a) v becomes IN; x_1 and x_2 become BN:
   $\Delta_1 = \epsilon_2^{\mathrm{BN}} + 2 \cdot (\epsilon_2^{\mathrm{free}} - \epsilon_2^{\mathrm{BN}})$
   (b) Then after setting $v \in$ LN, rule **(R5)** will make q internal and subsequently also x_1:
   $\Delta_2 = \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{free}} + \epsilon_2^{\mathrm{BN}}$.
   This amount is justified by the changing roles of the vertices in $N^+(q) \cup \{q\}$.

By the above case analysis we are able to conclude:

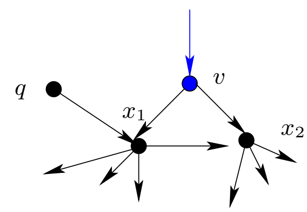

Fig. 1. The only situation which can occur in branching case (B8). The blue arc is contained in T.

Theorem 1. Directed Maximum Leaf Spanning Tree *can be solved in* $\mathcal{O}^*(1.9043^n)$ *steps.*

The proven run time bound admits only a small gap to the bound of $\mathcal{O}^*(1.8966^n)$ for the undirected version. It seems that we can benefit from degree two vertices only on a small scale in contrast to the undirected problem version. Speaking loosely if $v \in \mathrm{BN}_2$ and $x \in N(v)$ we can follow a WIN/WIN approach in the undirected version. Either $d(x)$ is quite big then we will add many vertices to BN or FL when v and subsequently x become internal. If $d(x)$ is small, say two, then by setting $v \in \mathrm{LN}$ the vertex x becomes a FL-node. This implies also an extra reduction of the measure. We point out that in the directed case the in- and out-degree of a vertex generally is not related. Thus, the approach described for the undirected problem remains barred for the directed version.

4 Conclusions

4.1 An Approach Using Exponential Space

The algorithm of J. Kneis *et al.* [8] can also be read in an exact non-parameterized way. It is not hard to see that it yields a running time of $\mathcal{O}^*(2^n)$. Alternatively, keep the cases (B1) and (B2) of Algorithm 1 and substitute all following cases by a simple branch on some BN-node. Using n as a measure we see that $\mathcal{O}^*(2^n)$ is an upper bound.

We are going to use the technique of memoization to obtain an improved running time. Let $SG^\alpha := \{G(V') \mid V' \subseteq V, |V'| \leq \alpha \cdot n\}$ where $\alpha = 0.141$. Then we aim to create the following table L indexed by some $G' \in SG^\alpha$ and some $V_{\mathrm{BN}} \subseteq V(G')$:

$L[G', V_{\mathrm{BN}}] = T'$ such that $|leaves(T')| = \min_{\tilde{T} \subseteq \mathcal{L}} |leaves(\tilde{T})|$ where

$\mathcal{L} = \{\tilde{T} \mid \tilde{T}$ is directed spanning tree for $G'_{\mathrm{BN}}$with root $r'\}$ and $G'_{\mathrm{BN}} = (V(G') \cup \{r', y\}, A(G') \cup (\{(r', y)\} \cup_{u \in V_{\mathrm{BN}}} (r', u))$ and r', y are new vertices.

Entries where such a directed spanning tree $\tilde{T}$ does not exits (e.g. if $V_{\mathrm{BN}} = \emptyset$) get the value $\emptyset$. This table can be filled up in time $\mathcal{O}^*(\binom{n}{\alpha \cdot n} \cdot 2^{\alpha n} \cdot 1.9043^{\alpha n}) \subseteq \mathcal{O}^*(1.8139^n)$. This running time is composed of enumerating SG^α, then by cycling through all possibilities for V_{BN} and finally solving the problem on instance

G'_{BN} with Algorithm 1.

Theorem 2. Directed Maximum Leaf Spanning Tree *can be solved in time* $\mathcal{O}^*(1.8139^n)$ *consuming* $\mathcal{O}^*(1.6563^n)$ *space.*

Proof. Run the above mentioned $\mathcal{O}^*(2^n)$-algorithm until $|G^r| \le \alpha \cdot n$ with $G^r := V \setminus internal(T)$. Then let $T^e = L[G^r, V(G^r) \cap \mathrm{BN}_T]$. Note that the vertex $r \in V(T^e)$ must be internal and $y \in leaves(T^e)$. By Lemma 1 we can assume that $A(\{r\} \cup N^+(r)) \subseteq T^e$. Now identify the vertices $BN_T \cap V(T^e)$ with $V(G^r) \cap \mathrm{BN}_T$ and delete r an y to a directed spanning tree $\hat{T}$ for the original graph G. Or more formally let $\hat{T} := T \cup (T^e \setminus A(\{r\} \cup N^+(r))$. Observe that $\hat{T}$ extends T to optimality. $\square$

Note that in the first phase we cannot substitute the $\mathcal{O}^*(2^n)$-algorithm by Algorithm 1. It might be the case that **(R6)** generates graphs which are not vertex-induced subgraphs of G.

4.2 Résumé

The paper at hand presented an algorithm which solves the Directed Maximum Leaf Spanning Tree problem in time $\mathcal{O}^*(1.9043^n)$. Although this algorithm follows the same line of attack as the one of [5] the algorithm itself differs notably. The approach of [5] does not simply carry over. To achieve our run time bound we had to develop new algorithmic ideas. This is reflected by the greater number of branching cases.

References

1. J. Blum, M. Ding, A. Thaeler, and X. Cheng. Connected dominating set in sensor networks and MANETs, *Handbook of Combinatorial Optimization, Vol. B*, pp. 329–369. Springer, Heidelberg, 2005.
2. P. S. Bonsma and F. Zickfeld. A 3/2-approximation algorithm for finding spanning trees with many leaves in cubic graphs. In *WG*, *LNCS* 5344:66–77. Springer, Heidelberg, 2008.
3. J. Daligault, G. Gutin, E. J. Kim, and A. Yeo. FPT algorithms and kernels for the directed k-leaf problem. In *Journal of Computer and System Sciences*, 2009. http://dx.doi.org/10.1016/j.jcss.2009.06.005
4. J. Daligault and S. Thomassé. On finding directed trees with many leaves. In *IWPEC*, 2009, to appear.
5. H. Fernau, A. Langer, M. Liedloff, J. Kneis, D. Kratsch, D. Raible and P. Rossmanith. An exact algorithm for the Maximum Leaf Spanning Tree problem. In *IWPEC*, 2009, to appear.
6. F.V. Fomin, F. Grandoni and D. Kratsch. Solving Connected Dominating Set Faster than 2^n. *Algorithmica*, 52(2):153–166, 2008.
7. F.V. Fomin, F. Grandoni and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56(5),2009.

8. J. Kneis, A. Langer, and P. Rossmanith. A new algorithm for finding trees with many leaves. In *ISAAC*, *LNCS* 5369:270–281. Springer, Heidelberg, 2008.
9. I. Koutis and R. Williams. Limits and Applications of Group Algebras for Parameterized Problems. In *ICALP (1)*, *LNCS* 5555:653–664. Springer, Heidelberg, 2009.
10. H.-I Lu and R. Ravi. Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms* 29:132–141, 1998.
11. D. Raible and H. Fernau. An Amortized Search Tree Analysis for k-Leaf Spanning Tree In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, 2010, to appear.
12. V. Raman and S. Saurabh. Parameterized algorithms for feedback set problems and their duals in tournaments, In *Theoretical Computer Science*, 351(3):446–458, 2006.
13. R. Solis-Oba. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In *ESA*, *LNCS* 1461:441–452. Springer, Heidelberg, 1998.
14. M. T. Thai, F. Wang, D. Liu, S. Zhu, and D.-Z. Du. Connected dominating sets in wireless networks different transmission ranges. *IEEE Trans. Mobile Computing* 6:1–10, 2007.