An exact algorithm for the Maximum Leaf Spanning Tree problem

Henning Fernau[1], Joachim Kneis[2], Dieter Kratsch[3], Alexander Langer[2], Mathieu Liedloff[4], Daniel Raible[1], and Peter Rossmanith[2]

[1] Universität Trier, FB 4—Abteilung Informatik, D-54286 Trier, Germany. {fernau,raible}@uni-trier.de

[2] Department of Computer Science, RWTH Aachen University, Germany.[‡] {kneis,langer,rossmani}@cs.rwth-aachen.de

[3] Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. kratsch@univ-metz.fr

[4] Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, 45067 Orléans Cedex 2, France. liedloff@univ-orleans.fr

Abstract. Given an undirected graph with n nodes, the Maximum Leaf Spanning Tree problem is to find a spanning tree with as many leaves as possible. When parameterized in the number of leaves k, this problem can be solved in time $O(4^k \text{poly}(n))$ using a simple branching algorithm introduced by a subset of the authors [12]. Daligault, Gutin, Kim, and Yeo [6] improved the branching and obtained a running time of $O(3.72^k \text{poly}(n))$. In this paper, we study the problem from an exponential time viewpoint, where it is equivalent to the Connected Dominating Set problem. Here, Fomin, Grandoni, and Kratsch showed how to break the $\Omega(2^n)$ barrier and proposed an $O(1.9407^n)$-time algorithm [10]. In light of some useful properties of [12] and [6], we present a branching algorithm whose running time of $O(1.8966^n)$ has been analyzed using the Measure-and-Conquer technique. Finally we provide a lower bound of $\Omega(1.4422^n)$ for the worst case running time of our algorithm.

1 Introduction

The Maximum Leaf Spanning Tree (MLST) problem, i.e., finding a spanning tree with as many leaves as possible, is one of the classical NP-complete problems [11]. Ongoing research on this topic is motivated by the fact that variants of this problem occur frequently in real life applications. For example, some broadcasting problems in network design ask to minimize the number of broadcasting nodes, which must be connected to a single root. This translates nicely to finding a spanning tree with many leaves and few internal nodes. There are a lot of results dealing with this topics, e.g., [5, 13–15].

The Maximum Leaf Spanning Tree problem is equivalent to the Connected Dominating Set problem, where one shall find a minimum set of connected nodes dominating the whole graph: It is easy to see that the internal

[‡] Partially supported by the DFG under grant RO 927/7.

nodes of a spanning tree with k leaves are a connected dominating set of size $|V| - k$ and vice versa.

Known results. In the field of exact exponential time algorithms, there is only the paper by Fomin, Grandoni, and Kratsch [10], where they present an algorithm with a runtime of $O(1.9407^n)$. This result was the first that improved over the trivial $\Omega(2^n)$ barrier, when n is the number of nodes. There is however a long research history for this problem in the field of parameterized complexity, see [1, 7, 9, 3, 8, 2, 4]. The currently fastest published algorithm is due to Kneis, Langer, and Rossmanith [12] with a runtime bounded by $O^*(4^k)$, which has been further improved to $O^*(3.72^k)$ by Daligault, Gutin, Kim, and Yeo in an yet to appear article (a preliminary version can be found in [6]), whose improvements are also used in our exact algorithm.

Our results. In the next sections we solve the MLST problem in time $O(1.8966^n)$, improving the result of [10]. The algorithm presented here is based on the parameterized one [12], which basically repeatedly branches on leaves of a subtree of the graph in order to decide whether it can remain a leaf or must become an internal node. If we analyze the running time as a function of n, we find that branching on nodes of small degree (with two possible successors) becomes the worst case resulting in a bad running time. This resembles the worst case of the parameterized algorithm, and the changes in [6] are based on improving exactly this case. We use a similar approach for our exact algorithm. We mark nodes as leaves as early as possible even when they are not yet attached to an internal node. In the Measure-and-Conquer analysis, this balances the bad cases against the better cases, i.e., the better cases "lend" some running time to the bad cases for an overall improvement. However, this approach requires a rather complicated measure and an involved analysis.

2 Preliminaries

Let $G = (V, E)$ be a simple, undirected graph. We denote by n the number of its vertices and by m the number of its edges. Given a vertex $v \in V$, the set of its neighbors is defined by $N(v) = \{\, u \in V \mid \{u, v\} \in E \,\}$. The closed neighborhood of v is $N[v] = \{v\} \cup N(v)$. Given a subset $S \subset V$, we define $N(S)$ as the set $\bigcup_{v \in S} N(v) \setminus S$ and for a $X \subset V$, we define $N_X(S) = N(S) \cap X$. We write $H \subseteq G$ if H is a subgraph of G.

A tree $T = (V_T, E_T)$ is a *subtree* of G (or a *tree in* G) if $T \subseteq G$. The tree T is a *spanning tree* of G if furthermore $V_T = V$. As usual, a node of degree 1 in T is called a *leaf* and all other nodes are called *internal* nodes. Wlog, we assume each spanning tree contains at least one internal node. Once we fix some arbitrary node, wlog an internal node, as the root of the tree, we can also speak of *parents* of nodes within this tree. A spanning tree is a maximum leaf spanning tree (MLST) if there is no spanning tree with a larger number of leaves.

In the following, we identify trees $T = (V_T, E_T)$ with the bipartition of V_T into the sets of internal nodes and leaves, denoted as internal(T) and leaves(T), respectively. Although there might be multiple subtrees of G sharing the same set of internal nodes and leaves, either both are subtrees of some optimal solution for MLST or none of them is (recall that G is undirected).

We assume the reader is familiar with the concepts of branching algorithms, branching vectors and their corresponding branching number, and the Measure-and-Conquer approach.

3 A new exact algorithm

The algorithm partitions the set of vertices of G into the sets of *free vertices* (Free), *floating leaves* (FL), *branching nodes* (BN), *leaf nodes* (LN), and *internal nodes* (IN), where the latter three form the nodes of some tree $T \subseteq G$. Initially, all vertices are in the set Free, i.e., the tree is empty.

The key idea of the algorithm is to recursively build a subtree $T \subseteq G$ with $V_T = \mathrm{IN} \cup \mathrm{BN} \cup \mathrm{LN}$, $\mathrm{internal}(T) = \mathrm{IN}$ and $\mathrm{leaves}(T) = \mathrm{BN} \cup \mathrm{LN}$, which might in some *branch* eventually turn into a spanning tree T' of G.

Definition 1. *Let $G = (V, E)$ be a graph, let $\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL} \subseteq V$ be disjoint sets of vertices, and let $x_1, \ldots, x_l \in V$. By $x_1 \to X_1, \ldots, x_l \to X_l$, where each X_i is one of* IN, BN, LN *or* FL, *we denote the* branch *that corresponds to moving each x_i to the respective set X_i, and additionally, if $X_i = \mathrm{IN}$, all $y \in N_{\mathrm{Free}}(x_i)$ to* BN *and all $y \in N_{\mathrm{FL}}(x_i)$ to* LN. *The notation is extended to sets $Y \to X$ in a straightforward manner. The recursive* branching *over multiple branches is denoted by*

$$\langle x_1 \to X_1, \ldots, x_l \to X_l \mid\mid \ldots \mid\mid x_1' \to X_1', \ldots, x_{l'}' \to X_{l'}' \rangle.$$

In particular, whenever Algorithm $\mathcal{M}$ decides that some node $x \in V$ becomes an internal node, all of its neighbors are directly attached to the tree, which is never worse than connecting these neighbors through some other nodes [12]. However, vertices of T that are in LN will always remain leaves in subsequent calls, whereas the status of a vertex in BN is still subject to change. Similarly, vertices in $\mathrm{FL} \subseteq V \setminus V_T$ will be leaves in the spanning tree T', but their parents in T' have not yet been determined. This is formally defined as follows.

Definition 2. *Let $G = (V, E)$ be a graph, and let $\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL} \subseteq V$ be disjoint sets of vertices and $T \subseteq G$ be a tree. We say T* extends *$(\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$ iff $\mathrm{IN} \subseteq \mathrm{internal}(T)$, $\mathrm{LN} \subseteq \mathrm{leaves}(T)$, $\mathrm{BN} \subseteq \mathrm{internal}(T) \cup \mathrm{leaves}(T)$, and $\mathrm{FL} \cap \mathrm{internal}(T) = \emptyset$.*

If $N(\mathrm{internal}(T)) \subseteq \mathrm{internal}(T) \cup \mathrm{leaves}(T)$, we call T an inner-maximal *tree. A node $v \in \mathrm{Free} \cup \mathrm{FL}$ is* unreachable, *if there is no path $uv_1 \ldots v_t v$, where $t \geq 0$, $u \in \mathrm{BN}$ and $v_i \in \mathrm{Free}$ for all $1 \leq i \leq t$.*

For any $v \in V \setminus (\mathrm{IN} \cup \mathrm{LN})$, we define its *degree* $d(v)$ as $d(v) = |N(v) \cap (\mathrm{Free} \cup \mathrm{FL})|$ if $v \in \mathrm{BN}$, as $d(v) = |N(v) \cap (\mathrm{Free} \cup \mathrm{FL} \cup \mathrm{BN})|$ if $v \in \mathrm{Free}$, and as $d(v) = |N(v) \cap (\mathrm{Free} \cup \mathrm{BN})|$ if $v \in \mathrm{FL}$.

Our algorithm uses the following reduction rules.

Definition 3. *Let $G = (V, E)$ be a graph and let* $\mathrm{IN} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{Free}$ *be a partition of V. We define the following reduction rules:*

(R1) *If there exist two adjacent vertices $u, v \in V$ such that $u, v \in \mathrm{FL}$ or $u, v \in \mathrm{BN}$, then remove the edge $\{u, v\}$.*
(R2) *If there exists a vertex $v \in \mathrm{BN}$ with $d(v) = 0$, then insert v into* LN *instead.*
(R3) *If there exists a vertex $v \in \mathrm{Free}$ with $d(v) = 1$, then insert v into* FL *instead.*
(R4) *If there exists a vertex $v \in \mathrm{Free}$ with no neighbors in $\mathrm{Free} \cup \mathrm{FL}$, then insert v into* FL *instead.*
(R5) *If there exists a triangle $\{x, y, z\}$ with $d(x) = 2$ and $x \in \mathrm{Free}$, then insert x into* FL *instead.*
(R6) *If there exists a vertex $u \in BN$ which is a cut vertex, then apply $u \to \mathrm{IN}$.*
(R7) *If there exist two adjacent vertices $u, v \in V$ such that $u \in \mathrm{LN}$ and $v \in V \setminus \mathrm{IN}$, then remove the edge $\{u, v\}$.*

The correctness proofs of reduction rules are straightforward and details are not given in this extended abstract due to space limitations.

The halting and branching rules are described in Algorithm $\mathcal{M}$ (see Figure 1). Their correctness is shown in the following Section. The running-time analysis is provided in Section 5.

4 Correctness of the algorithm

The following lemma will ease the forthcoming correctness proof. It enables us to turn some nodes into additional floating leaves in some special cases. A similar technique has already been used in [6].

Lemma 1. *Let $G = (V, E)$ be a graph, T a tree in G and $v \in leaves(T)$ such that $N(v) \setminus V(T) = \{x_1, x_2\}$. If every optimal spanning tree $T' \supseteq T$ is such that v is an internal node and each x_i is a leaf in T', then there is also some optimal spanning tree where additionally each $w \in N(\{x_1, x_2\}) \setminus (\mathrm{internal}(T) \cup \{v\})$ is a leaf.*

Lemma 2. *Algorithm $\mathcal{M}$ solves the* Maximum Leaf Spanning Tree *problem if called with $\mathrm{BN} = \{r\}$ and $\mathrm{IN} = \mathrm{LN} = \mathrm{FL} = \emptyset$, where r is the root of some optimal spanning tree.*

Proof. The reduction rules update a partition $\mathcal{P} = (\mathrm{Free}, \mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$ to a partition $\mathcal{P}' = (\mathrm{Free}', \mathrm{IN}', \mathrm{BN}', \mathrm{LN}', \mathrm{FL}')$ so that any maximum leaf spanning tree T' that extends $\mathcal{P}'$ has at least as many leaves as any spanning tree T extending $\mathcal{P}$. Note that given some disjoint subsets IN, BN, LN, FL, the subset Free is uniquely determined by $V \setminus (\mathrm{IN} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL})$. Thus, we omit the explicit notion of the set Free.

In the following, $(\mathrm{IN} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL})_{x_1 \to X_1, \ldots, x_l \to X_l}$ denotes the partition $(\mathrm{Free}', \mathrm{IN}', \mathrm{BN}', \mathrm{LN}', \mathrm{FL}')$ obtained from $(\mathrm{Free}, \mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$ by the algorithm

```
Algorithm $\mathcal{M}$
Input: A graph $G = (V, E)$, $\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL} \subseteq V$
-----------------------------------------------------------------------------------------
Reduce $G$ according to the reduction rules
**if** there is some unreachable $v \in \mathrm{Free} \cup \mathrm{FL}$ **then** return 0
**if** $V = \mathrm{IN} \cup \mathrm{LN}$ **then** return $|\mathrm{LN}|$
Choose a vertex $v \in \mathrm{BN}$ of maximum degree
**if** $d(v) \geq 3$ **or** $(d(v) = 2$ and $N_{\mathrm{FL}}(v) \neq \emptyset)$ **then**
    $\langle v \to \mathrm{LN} \mid\mid v \to \mathrm{IN} \rangle$                                   (B1)
**else if** $d(v) = 2$ **then**
    Let $\{x_1, x_2\} = N_{\mathrm{Free}}(v)$ such that $d(x_1) \leq d(x_2)$
    **if** $d(x_1) = 2$ **then**
        Let $\{z\} = N(x_1) \setminus \{v\}$
        **if** $z \in \mathrm{Free}$ **then**
            $\langle v \to \mathrm{LN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{IN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{LN} \rangle$       (B2)
        **else if** $z \in \mathrm{FL}$ **then** $\langle v \to \mathrm{IN} \rangle$
    **else if** $(N(x_1) \cap N(x_2)) \setminus \mathrm{FL} = \{v\}$ **and** $\forall z \in (N_{\mathrm{FL}}(x_1) \cap N_{\mathrm{FL}}(x_2))$,
        $d(z) \geq 3$ **then**
        $\langle v \to \mathrm{LN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{IN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{LN}, x_2 \to \mathrm{IN} \mid\mid$       (B3)
         $v \to \mathrm{IN}, x_1 \to \mathrm{LN}, x_2 \to \mathrm{LN}, N_{\mathrm{Free}}(x_1, x_2) \to \mathrm{FL}, N_{\mathrm{BN}}(x_1, x_2) \to \mathrm{LN} \rangle$
    **else** $(N(x_1) \cap N(x_2)) \setminus \mathrm{FL} \neq \{v\}$ **then**
            $\langle v \to \mathrm{LN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{IN} \mid\mid v \to \mathrm{IN}, x_1 \to \mathrm{LN}, x_2 \to \mathrm{IN} \rangle$      (B4)
**else if** $d(v) = 1$ **then**
    Let $P = (v = v_0, v_1, \ldots, v_k)$ be a maximum path such that
        $d(v_i) = 2$, $1 \leq i \leq k$, $v_1, \ldots, v_k \in \mathrm{Free}$
    Let $z \in N(v_k) \setminus V(P)$
    **if** $z \in \mathrm{FL}$ **and** $d(z) = 1$ **then** $\langle v_0, \ldots, v_k \to \mathrm{IN}, z \to \mathrm{LN} \rangle$
    **else if** $z \in \mathrm{FL}$ **and** $d(z) > 1$ **then** $\langle v_0, \ldots, v_{k-1} \to \mathrm{IN}, v_k \to \mathrm{LN} \rangle$
    **else if** $z \in \mathrm{BN}$ **then** $\langle v \to \mathrm{LN} \rangle$
    **else if** $z \in \mathrm{Free}$ **then** $\langle v_0, \ldots, v_k \to \mathrm{IN}, z \to \mathrm{IN} \mid\mid v \to \mathrm{LN} \rangle$        (B5)
```

Fig. 1. An algorithm for Maximum Leaf Spanning Tree. The notation $\langle v \to \mathrm{IN} \mid\mid v \to \mathrm{LN} \rangle$ denotes the corresponding recursive branches, e.g., in this case v either becomes an internal node or a leaf (see Definition 1).

in the $x_1 \to X_1, \ldots, x_l \to X_l$ branch. In particular, whenever Algorithm $\mathcal{M}$ decides that some nodes $X \subseteq \mathrm{BN} \cup \mathrm{Free}$ become internal nodes, all nodes in $N(X) \cap \mathrm{Free}$ become new branching nodes (BN) and all nodes in $N(X) \cap \mathrm{FL}$ become leaves (LN). Hence, Algorithm $\mathcal{M}$ always computes an inner-maximal tree. It thus remains to show that if there is some spanning tree T with k leaves that extends the current $(\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$, then Algorithm $\mathcal{M}$ calls itself with an new $(\mathrm{IN}', \mathrm{BN}', \mathrm{LN}', \mathrm{FL}')$ such that there is some spanning tree T' with k leaves that extends $(\mathrm{IN}', \mathrm{BN}', \mathrm{LN}', \mathrm{FL}')$ as well.

We prove this by induction. For the base step, any spanning tree extends $(\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$ with $\mathrm{BN} = \{r\}$ and $\mathrm{IN} = \mathrm{LN} = \mathrm{FL} = \emptyset$, where the root r is the only branching node. Now let T be a spanning tree with k leaves that extends $(\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \mathrm{FL})$, and let $v \in \mathrm{BN}$ be of maximum degree.

– If $d(v) \geq 3$ or $d(v) = 2$ and $N_{\text{FL}}(v) \neq \emptyset$, then Algorithm $\mathcal{M}$ calls itself recursively in (B1). Since v is either an internal node or a leaf in any spanning tree, T extends either $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN}}$ or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{LN}}$.
– If $d(v) = 2$, $N_{\text{Free}}(v) = \{x_1, x_2\}$ and $N(x_1) \setminus \{v\} = \{z\}$, such that $z \in \text{FL}$, we do not need to branch, since x_1 must somehow be connected to the tree in any solution extending $(\text{IN}, \text{BN}, \text{LN}, \text{FL})$, and v is the only choice. If otherwise $z \in \text{Free}$, then T either extends $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{LN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{LN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{IN}}$, because if v is not a leaf in T, then it is an internal node and x_1 is either leaf or internal node.
– In the case where $d(v) = 2$, $3 \leq d(x_1) \leq d(x_2)$ and $N(x_1) \cap N(x_2) \cap (\text{Free} \cup \text{BN}) = \{v\}$, the algorithm branches on all possibilities whether v, x_1 and x_2 are internal nodes or leaves. If there is some $z \in (N_{\text{FL}}(x_1) \cap N_{\text{FL}}(x_2))$ with $d(z) \leq 2$, not both x_1 and x_2 can be leaves and we skip the last branch (which yields (B4)). Otherwise, Lemma 1 guarantees that in the last branch where v must be an internal node and x_1 and x_2 are leaves, we can assume that all other nodes neighbors of x_1 and x_2 are leaves in some optimal solution as well. Hence there is a tree that extends either $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{LN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{IN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{LN},x_2\to\text{IN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{LN},x_2\to\text{LN},N_{\text{Free}}(x_1,x_2)\to\text{FL},N_{\text{BN}}(x_1,x_2)\to\text{LN}}$.
– In the case where $d(v) = 2$, $3 \leq d(x_1) \leq d(x_2)$ and $N(x_1) \cap N(x_2) \cap (\text{Free} \cup \text{BN}) \neq \{v\}$ we can assume that if v is an internal node in every optimal solution, either x_1 or x_2 is an internal node as well. Otherwise we could connect x_1 and x_2 to $z \in (N(x_1) \cap N(x_2)) \setminus \text{FL}$ instead of v, which might destroy the leaf z, that is connected somehow else, but yields the new leaf v. Since z is either a branching node or a free node, this is still allowed. Hence, there is also some optimal solution that extends $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{LN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{IN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{IN},x_1\to\text{LN},x_2\to\text{IN}}$.
– Finally, if $d(v) = 1$, let $P = (v = v_0, v_1, \ldots, v_k)$ be a maximum path such that $d(v_i) = 2$, $1 \leq i \leq k$, $v_1, \ldots, v_k \in \text{Free}$ and let $z \in (N(v_k) \setminus V(P))$, as described in Algorithm $\mathcal{M}$. If $z \in \text{FL}$ and $d(z) = 1$, all nodes in P must be internal nodes in any spanning tree that extends $(\text{IN}, \text{BN}, \text{LN}, \text{FL})$, because there is no other way to connect z. If otherwise $d(z) > 1$, there is always an inner-maximal solution where v_k is a leaf by a simple exchange argument.
If on the other hand $z \in \text{BN}$, then the nodes in P must either be connected through v or through z, and hence we can just decide to make v a leaf, again by a simple exchange argument.
Now assume $z \in \text{Free}$. Since T is inner-maximal we know by [12], that there is some inner-maximal T' that extends either $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v\to\text{LN}}$, or $(\text{IN}, \text{BN}, \text{LN}, \text{FL})_{v,v_1,\ldots,v_k,z\to\text{IN}}$ in this case.

Since this concludes a complete distinction of all possible $d(v)$, the claim follows by induction. □

5 Analysis of the running-time

To analyze the running-time, we use the following measure:

$$\mu(G) = \sum_{i=1}^{n} \epsilon_i^{\mathrm{BN}} |\mathrm{BN}_i| + \sum_{i=2}^{n} \epsilon_i^{\mathrm{Free}} |\mathrm{Free}_i| + \sum_{i=2}^{n} \epsilon_i^{\mathrm{FL}} |\mathrm{FL}_i|,$$

where BN_i (resp. Free_i and FL_i) denotes the set of vertices in BN (resp. Free and FL) with degree i, and the values of the ϵ's are chosen in $[0, 1]$ so that $\mu(G) \leq n$, more precisely:

- $\epsilon_0^{\mathrm{Free}} = \epsilon_1^{\mathrm{Free}} = 0$, $\epsilon_2^{\mathrm{Free}} = 0.731975$, $\epsilon_3^{\mathrm{Free}} = 0.946609$, and $\epsilon_i^{\mathrm{Free}} = 1$ for all $i \geq 4$;
- $\epsilon_0^{\mathrm{BN}} = 0$, $\epsilon_1^{\mathrm{BN}} = 0.661662$, $\epsilon_i^{\mathrm{BN}} = 0.730838$ for all $i \geq 2$;
- $\epsilon_0^{\mathrm{FL}} = \epsilon_1^{\mathrm{FL}} = 0$, $\epsilon_2^{\mathrm{FL}} = 0.331595$, $\epsilon_3^{\mathrm{FL}} = 0.494066$, and $\epsilon_i^{\mathrm{FL}} = 0.628886$ for all $i \geq 4$.

Lemma 3. *Let $G = (V, E)$ be a graph and let $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $d(v) \geq 3$ or $d(v) = 2$ and there is some $u \in N_{\mathrm{FL}}(v)$. Then branching according to (B1) yields a branching number less than 1.8966.*

Proof. By the reduction rules (R3) and (R6), we have $d(u) \geq 2$ for all $u \in \mathrm{Free} \cup \mathrm{FL}$.

1. In the first branch, v is added to the internal nodes. Thus, all nodes in $N_{\mathrm{Free}}(v)$ are added to the branching nodes. This reduces the degree of all these nodes by at least one, since the edge to v is not counted anymore. Moreover, all nodes in $N_{\mathrm{FL}}(v)$ are now leaf nodes. Thus, the measure decreases by at least
$$\Delta_1 = \epsilon_{d(v)}^{\mathrm{BN}} + \sum_{x \in N_{\mathrm{Free}}(v)} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{BN}}) + \sum_{y \in N_{\mathrm{FL}}(v)} \epsilon_{d(y)}^{\mathrm{FL}}.$$
2. In the second branch, v becomes a leaf. Therefore, the degree of all nodes in $N_{\mathrm{Free} \cup \mathrm{FL}}(v)$ decreases by one, as the edge to v is removed. This implies a change in the measure of at least
$$\Delta_2 = \epsilon_{d(v)}^{\mathrm{BN}} + \sum_{x \in N_{\mathrm{Free}}(v)} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{Free}}) + \sum_{y \in N_{\mathrm{FL}}(v)} (\epsilon_{d(y)}^{\mathrm{FL}} - \epsilon_{d(y)-1}^{\mathrm{FL}}).$$

Since higher degrees only imply a higher change, it is now sufficient to test all combinations where $d(v) = 3$ or $d(v) = 2$ and there is some $u \in N_{\mathrm{FL}}(v)$. For all other nodes $u \in N_{\mathrm{Free} \cup \mathrm{FL}}(v)$, we can similarly assume $2 \leq d(u) \leq 5$. The worst case (branching vector $(1.538324, 0.730838)$, branching number less than 1.8966) occurs at $d(v) = 3$, where v has three free neighbors of degree at least five. $\square$

Lemma 4. *Let $G = (V, E)$ be a graph and let $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $d(v) = 2$ and there is some $x_1 \in N_{\mathrm{Free}}(v)$ with $d(x_1) = 2$ and the remaining $z \in N(x_1) \setminus \{v\}$ is contained in Free. Then branching according to (B2) yields a branching number less than 1.8966.*

Proof. By the reduction rule (R5), we know that $z \neq x_2$. Moreover, (R3) implies $d(z) \geq 2$.

1. Again, v becomes leaf in the first branch. Similar to Lemma 3, this implies a change in the measure of at least
$$\begin{aligned}\Delta_1 &= \epsilon_2^{\mathrm{BN}} + (\epsilon_2^{\mathrm{Free}} - \epsilon_1^{\mathrm{FL}}) + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{Free}}) \\ &= \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{Free}} + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{Free}}),\end{aligned}$$
because x_1 becomes a floating leaf of degree one and the degree of x_2 decreases by one.
2. In the second branch, both v and x_1 become internal nodes, which implies that z and x_2 become branching nodes. Again, $d(z)$ and $d(x_2)$ decrease by one. The measure decreases by at least
$$\Delta_2 = \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{Free}} + (\epsilon_{d(z)}^{\mathrm{Free}} - \epsilon_{d(z)-1}^{\mathrm{BN}}) + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{BN}}).$$
3. In the third branch, v becomes an internal node and x_1 becomes a leaf connected to v. Thus, x_2 is now a branching node and $d(x_2)$ decreases. Moreover, $d(z)$ decreases by one as well. This implies that the measure is reduced by at least
$$\Delta_3 = \epsilon_2^{\mathrm{BN}} + \epsilon_2^{\mathrm{Free}} + (\epsilon_{d(z)}^{\mathrm{Free}} - \epsilon_{d(z)-1}^{\mathrm{Free}}) + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{BN}}).$$

Since $d(v) = d(x_1) = 2$, we need to try all possible combinations of $d(z)$ and $d(x_2)$, both between 2 and 5. Here, the worst case is $d(z) = d(x_2) = 5$ (1.8965 for branching vector $(1.462813, 1.731975, 2.001137)$). □

Lemma 5. *Let $G = (V, E)$ be a graph and let $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $N_{\mathrm{Free}}(v) = \{x_1, x_2\}$ with $3 \leq d(x_1) \leq d(x_2)$ and let $(N(x_1) \cap N(x_2)) \setminus \mathrm{FL} = \{v\}$. Finally, let $x_1 \notin N(x_2)$. Then branching according to (B3) yields a branching number less than* 1.8966.

Proof. 1. In the first branch, v becomes a leaf, which yields
$$\Delta_1 = \epsilon_2^{\mathrm{BN}} + (\epsilon_{d(x_1)}^{\mathrm{Free}} - \epsilon_{d(x_1)-1}^{\mathrm{Free}}) + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{Free}}).$$

2. In the second branch, v and x_1 become internal nodes. As a consequence, x_2 becomes a branching leaf and its degree decreases by one. Furthermore, the degree of all nodes in $N_{\mathrm{Free}\cup\mathrm{FL}}(x_1)$ decreases by one. We gain at least
$$\begin{aligned}\Delta_2 = \epsilon_2^{\mathrm{BN}} + \epsilon_{d(x_1)}^{\mathrm{Free}} + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{BN}}) + \sum_{x \in N_{\mathrm{Free}}(x_1)} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{BN}}) \\ + \sum_{y \in N_{\mathrm{FL}}(x_1)} \epsilon_{d(y)}^{\mathrm{FL}} + \sum_{z \in N_{\mathrm{BN}}(x_1)\setminus\{v\}} (\epsilon_{d(z)}^{\mathrm{BN}} - \epsilon_{d(z)-1}^{\mathrm{BN}}).\end{aligned}$$

3. In the third branch, v and x_2 become internal nodes, while x_1 becomes a leaf. Thus, the degree decreases by one for all nodes in $N_{\mathrm{Free}\cup\mathrm{FL}}(x_1)$ as well as for all nodes in $N_{\mathrm{BN}}(x_2)$. Moreover, all nodes in $N_{\mathrm{Free}}(x_2)$ become branching nodes and all nodes in $N_{\mathrm{FL}}(x_2)$ become leaves. Since $(N(x_1)\cap N(x_2))\setminus\mathrm{FL} = \emptyset$, the measure decreases by at least

$$\begin{aligned}\Delta_3 = \epsilon_2^{\mathrm{BN}} + \epsilon_{d(x_1)}^{\mathrm{Free}} + \epsilon_{d(x_2)}^{\mathrm{Free}} + \sum_{x\in N_{\mathrm{Free}}(x_1)} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{Free}})\\ + \sum_{y\in N_{\mathrm{FL}}(x_1)\setminus N(x_2)} (\epsilon_{d(y)}^{\mathrm{FL}} - \epsilon_{d(y)-1}^{\mathrm{FL}}) + \sum_{z\in N_{\mathrm{BN}}(\{x_1,x_2\})\setminus\{v\}} (\epsilon_{d(z)}^{\mathrm{BN}} - \epsilon_{d(z)-1}^{\mathrm{BN}})\\ + \sum_{x'\in N_{\mathrm{Free}}(x_2)} (\epsilon_{d(x')}^{\mathrm{Free}} - \epsilon_{d(x')-1}^{\mathrm{BN}}) + \sum_{y'\in N_{\mathrm{FL}}(x_2)} \epsilon_{d(y')}^{\mathrm{FL}}.\end{aligned}$$

4. In the last branch, v becomes an internal node, x_1 and x_2 become leaves, and all nodes in $N_{\mathrm{Free}}(\{x_1,x_2\})$ become floating leaves. Moreover, all nodes in $N_{\mathrm{BN}}(\{x_1,x_2\})$ become leaves as well and finally, the degree decreases by at least one for all $u \in N_{\mathrm{FL}}(\{x_1,x_2\})$. This implies that the measure decreases by at least

$$\begin{aligned}\Delta_4 = \epsilon_2^{\mathrm{BN}} + \epsilon_{d(x_1)}^{\mathrm{Free}} + \epsilon_{d(x_2)}^{\mathrm{Free}} + \sum_{x\in N_{\mathrm{Free}}(\{x_1,x_2\})} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{FL}})\\ + \sum_{y\in N_{\mathrm{FL}}(\{x_1,x_2\})\setminus(N(x_1)\cap N(x_2))} (\epsilon_{d(y)}^{\mathrm{FL}} - \epsilon_{d(y)-1}^{\mathrm{FL}})\\ + \sum_{y\in \mathrm{FL}\cap N(x_1)\cap N(x_2)} (\epsilon_{d(y)}^{\mathrm{FL}} - \epsilon_{d(y)-2}^{\mathrm{FL}})\\ + \sum_{z\in N_{\mathrm{BN}}(\{x_1,x_2\})\setminus\{v\}} \epsilon_{d(z)}^{\mathrm{BN}}.\end{aligned}$$

Again, we have to compute all possible neighborhoods. This requires us to test all $3 \le d(x_1) \le d(x_2) \le 5$, all $1 \le d(u) \le 2$ for all $u \in N_{\mathrm{BN}}(\{x_1,x_2\})$, all $2 \le d(u) \le 5$ for each $u \in N_{\mathrm{FL}}(\{x_1,x_2\})$ and finally all $2 \le d(u) \le 5$ for all $u \in N_{\mathrm{Free}}(\{x_1,x_2\})$. Note that it is sufficient to assume that all floating leaves are of degree at least two. Otherwise, some of the branches yield new instances that will be solved in polynomial time, because they are obvious "No" instances. Thus, the exponential parts of the runtime only depend on the other branches, which yields a much better runtime bound, even if some floating leaves are of degree one. Similarly, we can assume that floating leaves of degree two are not contained in $N(x_1)\cap N(x_2)$, because otherwise the last branch (both, x_1 and x_2 are in LN) is found to be a "No" instance in polynomial time.

It turns out that the largest branching number in this case is smaller than 1.8506 with a branching vector $(0.730838, 2.476690, 3.216207, 8.218955)$ for the case $d(x_1) = d(x_2) = 5$, $N_{\mathrm{Free}}(x_1) = \{u\}$ with $d(u) = 5$, $N_{\mathrm{FL}}(x_1) = \emptyset$, $N_{\mathrm{BN}}(x_1) = \{u_1,u_2,u_3\}$ with $d(u_1) = d(u_2) = d(u_3) = 2$, $N_{\mathrm{Free}}(x_2) = \{w\}$ with

$d(w) = 2$, $N_{\mathrm{FL}}(x_2) = \emptyset$, and $N_{\mathrm{BN}}(x_2) = \{w_1, w_2, w_3\}$ with $d(w_1) = d(w_2) = d(w_3) = 2$. □

Lemma 6. *Let $G = (V, E)$ be a graph and let* $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ *be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $N_{\mathrm{Free}}(v) = \{x_1, x_2\}$ with $3 \leq d(x_1) \leq d(x_2)$ and let $(N(x_1) \cap N(x_2)) \setminus \mathrm{FL} = \{v\}$. Finally, let $x_1 \in N(x_2)$. Then branching according to (B3) yields a branching number less than* 1.8966.

The proof is very similar to the previous lemma, we only need to make sure that the edge between x_1 and x_2 is not counted twice.

Lemma 7. *Let $G = (V, E)$ be a graph and let* $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ *be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $N_{\mathrm{Free}}(v) = \{x_1, x_2\}$ with $3 \leq d(x_1) \leq d(x_2)$ and let $(N(x_1) \cap N(x_2)) \setminus \mathrm{FL} \neq \{v\}$. Then branching according to (B4) yields a branching number less than* 1.8966.

Proof. Similar to Lemma 5 and Lemma 6, x_1 and x_2 can possibly be neighbors.

1. In the first branch, v becomes a leaf. Similar to above, we obtain at least
$$\varDelta_1 = \epsilon_2^{\mathrm{BN}} + (\epsilon_{d(x_1)}^{\mathrm{Free}} - \epsilon_{d(x_1)-1}^{\mathrm{Free}}) + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{Free}}).$$
2. In the second branch, v and x_1 become internal nodes. As a consequence, the degree decreases for all nodes in $N_{\mathrm{Free} \cup \mathrm{FL}}(\{v, x_1\})$ and these nodes turn into branching nodes or leaves, respectively. The measure decreases by at least
$$\begin{aligned}\varDelta_2 = \epsilon_2^{\mathrm{BN}} + \epsilon_{d(x_1)}^{\mathrm{Free}} + (\epsilon_{d(x_2)}^{\mathrm{Free}} - \epsilon_{d(x_2)-1}^{\mathrm{BN}}) + \sum_{x \in N_{\mathrm{Free}}(x_1) \setminus \{x_2\}} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{BN}}) \\ + \sum_{y \in N_{\mathrm{FL}}(x_1)} \epsilon_{d(y)}^{\mathrm{FL}} + \sum_{z \in N_{\mathrm{BN}}(x_1) \setminus \{v\}} (\epsilon_{d(z)}^{\mathrm{BN}} - \epsilon_{d(z)-1}^{\mathrm{BN}}).\end{aligned}$$
Note that when $x_2 \in N(x_1)$, $d(x_2)$ decreases even more. However, this estimation is good enough to obtain the claimed bounds.
3. In the last branch, v and x_2 become internal nodes and x_1 becomes a leaf. As usual, the measure decreases by at least
$$\begin{aligned}\varDelta_3 = \epsilon_2^{\mathrm{BN}} + \epsilon_{d(x_1)}^{\mathrm{Free}} + \epsilon_{d(x_2)}^{\mathrm{Free}} + \sum_{x \in N_{\mathrm{Free}}(x_1) \setminus \{x_2\}} (\epsilon_{d(x)}^{\mathrm{Free}} - \epsilon_{d(x)-1}^{\mathrm{Free}}) \\ + \sum_{y \in N_{\mathrm{FL}}(x_1)} (\epsilon_{d(y)}^{\mathrm{FL}} - \epsilon_{d(y)-1}^{\mathrm{FL}}) + \sum_{z \in N_{\mathrm{BN}}(x_1) \setminus \{v\}} (\epsilon_{d(z)}^{\mathrm{BN}} - \epsilon_{d(z)-1}^{\mathrm{BN}}).\end{aligned}$$

In all three cases, we only analyzed how the neighbors of x_1 are affected and omitted the neighbors of x_2. Thus, we do not have to distinguish between nodes in $N(x_1) \setminus N(x_2)$ and $N(x_1) \cap N(x_2)$. Similar to previous lemmas, we can safely assume that $d(u) \geq 2$ for all floating leaves $u \in N(x_1)$.

In order to compute all possible branching vectors, we need to test all $3 \le d(x_1) \le d(x_2) \le 5$. Furthermore, we need to try all $1 \le d(u) \le 2$ for all $u \in N_{\mathrm{BN}}(x_1)$, all $2 \le d(u) \le 5$ for each $u \in N_{\mathrm{FL}}(x_1)$ and finally all $2 \le d(u) \le 5$ for all $u \in N_{\mathrm{Free}}(x_1)$.

The worst case of 1.8966 (branching vector $(0.730838, 2.407514, 2.869190)$) occurs when $d(x_1) = d(x_2) = 5$, $N_{\mathrm{Free}}(x_1) = \{u_1, u_2\}$ with $d(u_1) = d(u_2) = 5$, $N_{\mathrm{FL}}(x_1) = \emptyset$, and $N_{\mathrm{BN}}(x_1) = \{u'_1 u'_2\}$ with $d(u'_1) = d(u'_2) = 2$. □

Lemma 8. *Let $G = (V, E)$ be a graph and let* $\mathrm{Free} \cup \mathrm{BN} \cup \mathrm{LN} \cup \mathrm{FL} \cup \mathrm{IN}$ *be a partition of V. Moreover let $v \in \mathrm{BN}$ such that $d(v) = 1$. Then branching according to (B5) yields a branching number less than* 1.8966.

Proof. Let $v_1, \dots, v_k$ and $z \in V$ as described in Algorithm $\mathcal{M}$ and recall that $d(z) \ge 3$ and $z \in \mathrm{Free}$.

1. In the first branch, v becomes an internal node as well as all $v_1, \dots, v_k$ and z. This implies that the measure decreases by at least
$$\Delta_1 = \epsilon_1^{\mathrm{BN}} + k\epsilon_2^{\mathrm{Free}} + \epsilon_{d(z)}^{\mathrm{Free}}.$$
2. In the other branch, v becomes a leaf. If $k = 0$, then the degree of z will decrease and otherwise the node v_1 becomes a floating leaf of degree one. Therefore, we gain at least
$$\Delta_2 = \epsilon_1^{\mathrm{BN}} + \min(\epsilon_{d(z)}^{\mathrm{Free}} - \epsilon_{d(z)-1}^{\mathrm{Free}}, \epsilon_2^{\mathrm{Free}}).$$

The worst case occurs when $d(z) = 5$ and $k = 0$ with a branching vector of $(1.661662, 0.661662)$ and a branching number less than 1.8966. □

From the above lemmas as well as from Lemma 2, which guarantees the correctness of our algorithm, we can conclude our main result.

Theorem 1. *The given algorithm solves the* Maximum Leaf Spanning Tree *problem in time $O(1.8966^n)$.*

It is well known that the current worst case running-time analysis, even based on Measure-and-Conquer, overestimate the upper bounds. The following Theorem gives a lower bound on this worst-case running-time enlighting the reader on the prevision of the analysis. We recall here that Fomin et al. [10] present an algorithm solving the problem whose worst-case running time is upper bounded by $O(1.9407^n)$ and they provide a lower bound of $\Omega(1.3195^n)$.

Theorem 2. *There is lower bound of $\Omega(3^{n/3}) = \Omega(1.4422^n)$ for the worst case running time of our algorithm.*

6 Concluding remarks

We improved the running time required to solve the MLST problem. Since the algorithm is based on a parameterized algorithm that works for directed graphs as well, it would be interesting to study the running time for directed graphs. However, some details that helped to improve the running time of our algorithm work only for undirected graphs. We believe that small modifications might be sufficient to gain similar runtime bounds for directed graphs.

References

1. H. L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993.
2. P. Bonsma. *Sparse cuts, matching-cuts and leafy trees in graphs.* PhD thesis, University of Twente, the Netherlands, 2006.
3. P. S. Bonsma, T. Brüggemann, and G. J. Woeginger. A faster FPT algorithm for finding spanning trees with many leaves. In *Proc. of 28th MFCS*, volume 2747 of *LNCS*, pages 259–268. Springer, 2003.
4. P. S. Bonsma and F. Zickfeld. Spanning trees with many leaves in graphs without diamonds and blossoms. In *Proc. of 8th LATIN*, number 4957 in LNCS, pages 531–543. Springer, 2008.
5. F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):908–920, 2004.
6. J. Daligault, G. Gutin, E. J. Kim, and A. Yeo. FPT Algorithms and Kernels for the Directed k-Leaf Problem. CoRR abs/0810.4946, 2008.
7. R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, pages 219–244. Boston: Birkhäuser, 1995.
8. V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-time extremal structure I. In *Proc. of 1st ACiD*, pages 1–41, 2005.
9. M. R. Fellows, C. McCartin, F. A. Rosamond, and U. Stege. Coordinatized kernels and catalytic reductions: An improved FPT algorithm for max leaf spanning tree and other problems. In *Proc. of 20th FSTTCS*, pages 240–251. Springer-Verlag, 2000.
10. F. V. Fomin, F. Grandoni, and D. Kratsch. Solving connected dominating set faster than 2^n. *Algorithmica*, 52(2):153–166, 2008.
11. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.
12. J. Kneis, A. Langer, and P. Rossmanith. A new algorithm for finding trees with many leaves. In *Proc. of 19th ISAAC*, number 5369 in LNCS, pages 270–281. Springer, 2008.
13. W. Liang. Constructing minimum-energy broadcast trees in wireless ad hoc networks. In *Proc. of 3rd MOBIHOC*, pages 112–122, 2002.
14. M. A. Park, J. Willson, C. Wang, M. Thai, W. Wu, and A. Farago. A dominating and absorbent set in a wireless ad-hoc network with different transmission ranges. In *Proc. of 8th MOBIHOC*, pages 22–31, New York, NY, USA, 2007. ACM.
15. M. Thai, F. Wang, D. Liu, S. Zhu, and D.Z. Du. Connected dominating sets in wireless networks with different transmission ranges. *IEEE Trans. Mobil. Comp.*, 6(7):721–730, 2007.