Algorithmica
DOI 10.1007/s00453-013-9827-7

# Better Approximation Algorithms for the Maximum Internal Spanning Tree Problem

**Martin Knauer · Joachim Spoerhase**

Received: 31 October 2012 / Accepted: 12 August 2013

**Abstract** We examine the problem of determining a spanning tree of a given graph such that the number of internal nodes is maximum. The best approximation algorithm known so far for this problem is due to Prieto and Sloper and has a ratio of 2. For graphs without pendant nodes, Salamon has lowered this factor to $\frac{7}{4}$ by means of local search. However, the approximative behaviour of his algorithm on general graphs has remained open. In this paper we show that a simplified and faster version of Salamon's algorithm yields a $\frac{5}{3}$-approximation even on general graphs. In addition to this, we investigate a node weighted variant of the problem for which Salamon achieved a ratio of $2 \cdot \Delta(G) - 3$. Extending Salamon's approach we obtain a factor of $3 + \epsilon$ for any $\epsilon > 0$. We complement our results with worst case instances showing that our analyses are tight.

**Keywords** Approximation algorithm · Spanning tree

## 1 Introduction

The MAXIMUM INTERNAL SPANNING TREE problem (MAXIST) consists in finding a spanning tree of a given graph such that the number of internal nodes is maximized. MAXIST is a natural optimization version of the HAMILTONIAN PATH problem and can be motivated by the design of cost-efficient communication networks [11]. Fernau et al. [1] describe a different application in the area of water supply networks where spanning trees with many internal nodes minimize turbulences in the supply network. The problem is closely related to the MINIMUM LEAF SPANNING

A preliminary version of this article appeared in Proc. 11th Algorithms and Data Structures Symposium (WADS 2009) [5].

M. Knauer · J. Spoerhase (✉)
Chair of Computer Science I, University of Würzburg, Am Hubland, 97074 Würzburg, Germany
e-mail: joachim.spoerhase@uni-wuerzburg.de

TREE problem (MINLST) which asks for a spanning tree with a minimum number of leaves. Although MINLST and MAXIST lead to the same constructive problem, they have different approximability properties.

## 1.1 Previous Work

Lu and Ravi [6] introduced MINLST and showed that it admits no constant factor approximation algorithm unless some NP-hard problem can be solved in deterministic quasi-polynomial time. Flandrin et al. [3] investigated conditions for the existence of spanning trees with few leaves. Interestingly, the aforementioned non-approximability result for MINLST does *not* carry over to MAXIST since the latter has a different objective function. Indeed, Prieto and Sloper [8] gave an efficient 2-approximation algorithm for MAXIST based on local search. Later, Salamon and Wiener [11] developed a simple linear time algorithm, which is basically a slight modification of depth first search, and showed that it yields a 2-approximation, too. Moreover, they obtained that MAXIST is approximable within factors $\frac{3}{2}$ and $\frac{6}{5}$ on claw-free and cubic graphs, respectively. The fixed parameter complexity of MAXIST was first studied by Prieto and Sloper [8]. They gave an $O(k^2)$ kernel which proves that the problem is fixed parameter tractable. Here, $k$ denotes the number of internal nodes. Recently, Fomin et al. [4] provided an improved $O(k)$ kernel for the problem. Fernau et al. [1] presented exact exponential time algorithms. In particular they gave $O^*(2^n)$ and $O^*(1.8612^n)$ algorithms for solving MAXIST on general graphs and graphs of maximum degree three, respectively. Here, the $O^*$-notation disregards polynomial factors. See the overview of Salamon [10] for more results on MAXIST.

The existence of efficient approximation algorithms with a ratio better than 2 has been an open problem so far [11]. Recently some progress was made towards answering this question: Salamon [9] suggested a local search based algorithm for MAXIST and proved by means of linear programming techniques that it terminates with a $\frac{7}{4}$-approximation on graphs without pendant nodes (nodes of degree 1). Although this algorithm applies also to arbitrary graphs, its approximative behaviour has remained open for the general case.

Additionally, Salamon [9] investigated a node weighted version of the problem (MAXWIST) for which he provided a local search algorithm with an approximation ratio of $2 \cdot \Delta(G) - 3$ where $\Delta(G)$ denotes the maximum degree of $G$.

## 1.2 Our Contribution

We will show that Salamon's algorithm reaches a $\frac{5}{3}$-approximation even on *general graphs*, which constitutes a considerable improvement over the previously known factor of $\frac{7}{4}$ known for graphs without pendant nodes [9]. Thereby, we also surpass the approximation ratio of 2 for the problem on general graphs [8, 11] which is often a critical bound for combinatorial optimization problems. Moreover, it turns out that a substantially smaller neighborhood structure in the local search is sufficient to guarantee the approximation ratio. This leads to a better running time.

We will also tackle the weighted case. By extending the neighborhood of Salamon's algorithm we are able to approximate MAXWIST within a factor of $3 + \epsilon$ for any $\epsilon > 0$, which is the first constant-factor approximation for MaxWIST and greatly improves upon the previously best degree-based bound.

To achieve our improved results, we introduce a new technique of analyzing edge-flipping local search for MaxIST, which differs significantly from previous methods [8, 9, 11]. Roughly speaking, we show that certain leaves in a local optimum are always associated with a sufficiently large number of internal nodes so-called *companions*. The effectiveness of our approach is underlined by the fact that our analyses are tight. We also demonstrate that Salamon's original algorithm [9] does not perform better than ours in the worst case although it uses a significantly larger neighborhood. We believe that our method could also be helpful for analyzing local search algorithms for similar problems such as MAXIMUM LEAF SPANNING TREE or MINIMUM BRANCH NODE SPANNING TREE. Chimani and Spoerhase [2] recently described a local search algorithm for the related MAXIMUM PATH NODE SPANNING TREE.

## 1.3 Organization of the Paper

The paper is organized as follows: In Sect. 2 we will present our algorithm LOST-LIGHT which constitutes a substantial simplification of the algorithm LOST proposed by Salamon [9]. Section 3 is devoted to the analysis of the approximation ratio of our algorithm. Here we will prove the factor of $\frac{5}{3}$ which is the heart of this paper. The node weighted version of MAXIST will then be examined in Sect. 4. We will complete our results with suitable worst case instances in Sect. 5 demonstrating that the analyses of our algorithms are best possible.

## 1.4 Preliminaries

Let $G = (V, E)$ be an undirected, simple graph and $G'$ be a subgraph of $G$. Then we denote by $E(G')$ the set of edges of $G'$. We say that $v$ is a *$G'$-neighbor* of $u$, if $u$ and $v$ are adjacent in $G'$. If $v$ is not a $G'$-neighbor of $u$ then $u$ and $v$ are called *$G'$-independent*. By $d_{G'}(u)$ we denote the *degree* of $u$ in $G'$. Let $T$ be a spanning tree of $G$. Then any node $u$ with degree $d_T(u) = 1$ is called a *$T$-leaf* or *leaf of $T$*. All other nodes of $T$ are referred to as *$T$-internal nodes* or alternatively *internal nodes of $T$*. The set of leaves and the set of internal nodes of $T$ are denoted by $L(T)$ and $I(T)$, respectively. By $P_T(u, v)$ we denote the unique path between nodes $u, v$ in $T$ and by $u^{\to v}$ the unique $T$-neighbor of $u$ on $P_T(u, v)$. A *$T$-branching* is a node $u$ in $T$ with degree $d_T(u) \geq 3$. If $T$ is not a path and $l$ is a leaf of $T$ then $b(l)$ is the unique branching which is closest to $l$ in $T$. Moreover, $l$ is said to be *$x$-supported*, if node $x$ does not lie on path $P_T(l, b(l))$ and $x$ is $G$-adjacent (but not $T$-adjacent) to $l$. The *length* of a path is the number of its edges.

If $S' \subseteq S$ is a subset of an arbitrary set $S$ weighted by a function $c: S \to \mathbb{Q}$ then let us use the notation $c(S') := \sum_{s \in S'} c(s)$.

## 1.5 Problem Definition

An instance of the MAXIMUM INTERNAL SPANNING TREE problem (MAXIST) is a connected and undirected graph $G = (V, E)$. The goal is to find a spanning tree $T$ of $G$ such that $|I(T)|$ is maximum among all spanning trees of $G$.

An instance of the MAXIMUM WEIGHTED INTERNAL SPANNING TREE problem (MAXWIST) is a connected, undirected graph $G = (V, E)$ whose nodes are weighted by a function $c: V \to \mathbb{Q}^+$. The goal is to find a spanning tree $T$ of $G$ such that $c(I(T))$ is maximum among all spanning trees of $G$.

## 2 The Algorithm

Salamon [9] suggests a local search algorithm, named LOST (locally optimal spanning tree), for approximating MAXIST. This algorithm maintains a spanning tree $T$ which is initialized with a depth-first search tree of the input graph $G = (V, E)$. Thereafter, it performs a sequence of local improvement operations, taken from a fixed set of so called *rules*. Each of these rules consists of a precondition and an action which replaces edges of $T$ with non-tree edges of $G$. The algorithm terminates when no more rule is applicable.

Our algorithm LOSTLIGHT applies the same framework but it employs only five of the 14 rules used by LOST. This is reflected in a running time reduced by a linear factor. In terms of the worst case ratio, we do not lose anything when dropping the additional rules of LOST, as we shall see in Sect. 5. Our algorithm may be initialized with an arbitrary spanning tree to achieve the claimed performance guarantee.

Let us adopt the following technical conventions: From now on, the notations $u^{\to v}$ and $b(l)$ will always be interpreted with respect to the spanning tree $T$ maintained by LOSTLIGHT. This is particularly important to avoid confusions, since later we will also deal with trees different from $T$ such as the global optimum $T^*$. Moreover, we will assume that $T$ is not already a path which ensures that $b(l)$ is defined for every leaf of $T$.

LOSTLIGHT uses the following five rules (confer Fig. 1):

Rule 1 If there are two $G$-adjacent leaves $l_1, l_2$ in $T$ then add edge $(l_1, l_2)$ and remove $(b(l_1), b(l_1)^{\to l_1})$.

Rule 2.1 If there is an $x$-supported leaf $l$ in $T$ such that $x^{\to l}$ is a branching in $T$ then add $(l, x)$ and remove $(x, x^{\to l})$.

Rule 2.2 If there are two $T$-leaves $l_1, l_2$ and a node $x$ such that $x^{\to l_1}$ has degree two in $T$ and edges $(l_1, x), (l_2, x^{\to l_1})$ are in $E - E(T)$ then add $(l_1, x)$ and remove $(x, x^{\to l_1})$. Afterwards apply Rule 1 to $l_2$ and the new leaf $x^{\to l_1}$.

Rule 3.1 If there is an $x$-supported leaf $l$ in $T$ such that $b(l)^{\to x}$ is a branching then add $(l, x)$ and remove $(b(l), b(l)^{\to x})$.

Rule 3.2 If there is an $x$-supported leaf $l_1$ and a leaf $l_2$ such that $d_T(b(l_1)^{\to x}) = 2$ and $(l_2, b(l_1)^{\to x}) \in E - E(T)$ then add $(l_1, x)$ and remove $(b(l_1), b(l_1)^{\to x})$. Afterwards apply Rule 1 to $l_2$ and the new leaf $b(l_1)^{\to x}$.

Notice that Salamon [9] additionally requires $l_1$ to be $x$-supported in Rule 2.2. However, it is easy to verify that this restricted Rule 2.2 is applicable to leaf $l_2$, node $u := x^{\to l_1}$ and $u^{\to l_2} = x$ if $l_1$ is *not* $x$-supported. Hence, our extended rule does not add any power to the local search. Therefore, we may drop this limitation which keeps the presentation simpler.

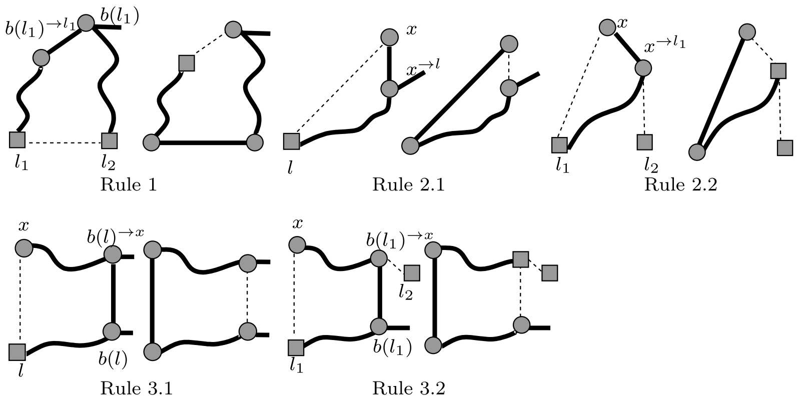

**Fig. 1** The rules of LOSTLIGHT. *Squares represent* $T$-leaves. *Thick* and *dashed straight lines* mark edges of $T$ and non-tree edges, respectively. *Wiggly lines* depict paths in $T$. In the constellations depicted for Rules 2.2, 3.2 we apply Rule 1 afterwards

It is easy to see that each of the Rules 1–3.2 increases the number of internal nodes in $T$. Thus the algorithm performs at most $|V|$ iterations. Each of the rules, in turn, can be carried out in $O(|V|^2)$ time as shown in [9]. We obtain:

**Lemma 1** *Algorithm* LOSTLIGHT *runs in time* $O(|V|^3)$.

By comparison, algorithm LOST needs time $O(|V|^4)$. This is basically because some of its additional rules do not discard leaves which may lead to overall $\Omega(|V|^2)$ iterations.

## 3 The Analysis

In this section we will analyse the quality of spanning trees generated by LOSTLIGHT. We will show that each spanning tree $T$, which is locally optimal with respect to the Rules 1–3.2, is a $\frac{5}{3}$-approximation. More formally, if spanning tree $T^*$ is globally optimal then $|I(T^*)| \le \frac{5}{3}|I(T)|$ holds.

### 3.1 Covered and Uncovered Leaves

Let $T^*$ be an optimal solution for MAXIST and $T$ an output of LOSTLIGHT. Let further $r$ be an arbitrary internal node of $T$ and consider $T^*$ as rooted at $r$. A $T$-leaf $l$ is said to be *covered*, if there is a $T$-leaf $l'$ such that $l'$ is a descendant of $l$ in $T^*$. Otherwise, $l$ is called *uncovered*. We denote the set of covered and the set of uncovered leaves by $C$ and $U$, respectively.

Consider an uncovered leaf $l$ and the subtree $T_l^*$ of $T^*$ hanging from $l$. Since $l$ is uncovered, $T_l^*$ cannot contain any $T$-leaf except for $l$ itself. Therefore, if $l'$ is another uncovered leaf then $T_l^*$ and $T_{l'}^*$ are disjoint. Since any subtree hanging from some uncovered leaf contains at least one $T^*$-leaf we conclude that $|U| \le |L(T^*)|$. Hence

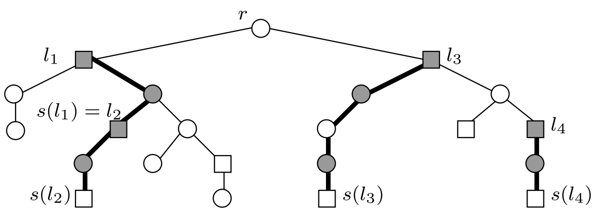

**Fig. 2** Illustration of $T^*$ with $T$-leaves represented by *squares. The grey squares* $l_1, \dots, l_4$ are the covered leaves. The paths $P(l_i)$ are marked by *thick lines*. *Grey circles* are path companions

$$\begin{aligned}|I(T^*)| &= (|I(T)| + |L(T)|) - |L(T^*)| \\ &\le |I(T)| + |L(T)| - |U| \\ &= |I(T)| + |C|.\end{aligned}$$

Dividing both hands by $|I(T)|$ we obtain the following central inequality:

$$\frac{|I(T^*)|}{|I(T)|} \le 1 + \frac{|C|}{|I(T)|}. \tag{1}$$

Inequality (1) says that in order to obtain a good approximation we have to bound $|I(T)|$ from below in terms of $|C|$. Indeed, as a result of our analysis, we will show that $|I(T)| \ge \frac{3}{2}|C|$ which yields the desired approximation ratio of $\frac{5}{3}$.

The central idea of our proof consists in identifying for each covered leaf $l$ a set $I(l)$ of distinctive internal nodes referred to as *companions of* $l$. For technical reasons we will divide the companions of $l$ into sets $I_\mathrm{p}(l)$ and $I_\mathrm{a}(l)$ of so called *path companions* and *auxiliary companions*. Showing that the total number of companions is at least $\frac{3}{2}|C|$ we obtain our factor of $\frac{5}{3}$ by inequality (1). The main difficulty will be that the companion sets $I(\cdot)$ need not be disjoint. To keep the overall set of companions from collapsing we will bound the *frequency* of the companions in the family of companion sets.

## 3.2 Path Companions

Our first step in identifying distinctive companions for any covered leaf $l$ consists in assigning to $l$ a unique subpath $P(l)$ of $T^*$. We will then single out certain internal nodes from $P(l)$ forming the set $I_\mathrm{p}(l)$ of path companions of $l$. The path $P(l)$ will also play a crucial role in the definition of auxiliary companions in the next section.

Consider an arbitrary covered leaf $l$. We pick a $T$-leaf, denoted by $s(l)$, such that $s(l)$ is a descendant of $l$ in $T^*$ and there are no other $T$-leaves on path $P_{T^*}(l, s(l))$ except for $l$ and $s(l)$. We set $P(l) = P_{T^*}(l, s(l))$. Clearly there may be more than one $T$-leaf satisfying the conditions imposed on $s(l)$ but we need only one of them. For an illustration of this path construction confer Fig. 2.

Since Rule 1 is not applicable in $T$ we can state that any pair of distinct $T$-leaves is $G$-independent. Therefore, it is impossible that some $P(l)$ consists of only one single edge. Moreover, it is easy to see that the union $F$ of the paths $P(\cdot)$ constitutes a subforest of $T^*$ having no $F$-branchings. We condense the foregoing considerations in the following lemma:

**Lemma 2** *Let $l$ and $l'$ be distinct covered leaves. Then $P(l)$ has always length at least* 2. *Moreover, if $P(l)$ and $P(l')$ share common nodes then either $l = s(l')$ or $l' = s(l)$.*

We are now ready to define our path companions (confer also Fig. 2):

**Definition 1** (Path companions) Let $l$ be a covered leaf. The *upper path companion of $l$* is the $T^*$-neighbor of $l$ on path $P(l)$. The $T^*$-neighbor of $s(l)$ on $P(l)$ is called the *lower path companion of $l$*. The set $I_{\mathrm{p}}(l)$ of *path companions of $l$* consists of the upper and the lower path companion of $l$.

Observe that in the above definition the upper and the lower path companions of a covered leaf may coincide, i.e., a leaf may have only one path companion. As a byproduct of Lemma 2, we obtain that $T$ is a 2-approximation: Since no $I_{\mathrm{p}}(\cdot)$ is empty and all those sets are pairwisely disjoint we conclude that there are at least $|C|$ companions and so $|I(T^*)| \leq 2 \cdot |I(T)|$ by inequality (1).

## 3.3 Auxiliary Companions

In the sequel we are going to improve the above ratio of 2 by identifying for each covered leaf $l$ a set $I_{\mathrm{a}}(l)$ of additional companions which we call *auxiliary companions* of $l$. The definition of auxiliary companions is inspired by the specific structure of our local optimum $T$ reflected in the following two simple observations:

**Observation 1** *Let $l$ be a $T$-leaf and $(l, x) \in E - E(T)$. Then $x^{\to l}$ has degree two in $T$ and is $G$-independent from any leaf in $L(T) - l$.*

*Proof* The claim follows immediately since Rules 2.1 and 2.2 are not applicable. □

**Observation 2** *Let $l$ be an $x$-supported $T$-leaf then $b(l)^{\to x}$ has degree two in $T$ and is $G$-independent from any leaf of $T$.*

*Proof* The claim follows immediately since Rules 2.1, 3.1, and 3.2 are not applicable. □

Both of the above properties predict the existence of certain degree-two nodes which we will now associate with covered leaves in an (almost) unique way (confer Fig. 3):

**Definition 2** (Auxiliary companions) Let $l$ be a covered leaf. Then the set $I_{\mathrm{a}}(l)$ of *auxiliary companions of $l$* consists of the following internal nodes:

(i) Nodes of the form $x^{\to l}$ if $x$ is the *upper* path companion of $l$ but not $T$-adjacent to $l$.
(ii) Nodes of the form $x^{\to s(l)}$ if $x$ is the *only* path companion of $l$ but not $T$-adjacent to $s(l)$.
(iii) Nodes of the form $b(l)^{\to x}$ if $b(l)$ is the *upper* path companion of $l$, and $l$ is $x$-supported.

Note that if $l$ has only one path companion $x$ then $x$ is in particular the upper path companion of $l$ and therefore any of the three cases in Definition 2 may apply to $l$.

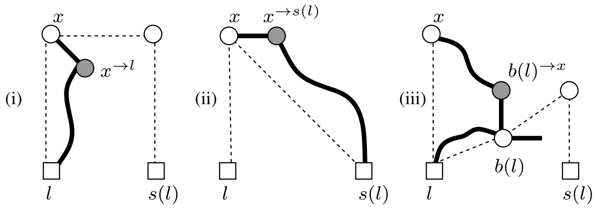

**Fig. 3** Illustration of $T$ with auxiliary companions represented by *grey nodes*. *Thick* and *dashed lines* mark edges of $T$ and $T^*$, respectively. *Wiggly lines* depict paths of $T$

At first glance Definition 2 might appear somewhat confusing. But a closer look reveals that all auxiliary companions share the following nice properties which guarantee (confer next section) that they never occur in more than two sets $I_{\mathrm{a}}(\cdot)$.

**Lemma 3** *Let $l$ be a covered leaf and $u \in I_{\mathrm{a}}(l)$ be an auxiliary companion of $l$. Then $u$ satisfies the following conditions*:

(i) *$u$ is not a path companion of $l$,*
(ii) *$u$ is $G$-adjacent to at most one $T$-leaf, and this is always in $\{l, s(l)\}$,*
(iii) *$u$ has degree $d_T(u) = 2$,*
(iv) *$u$ is $T$-adjacent to the upper path companion of $l$.*

*Proof* Statements (ii) and (iii) are immediate consequences of Observations 1 and 2. Statement (iv) follows immediately from the construction of auxiliary companions. In order to show statement (i), assume that $u$ is simultaneously an auxiliary and a path companion of $l$. Consider first that $u$ complies with Definition 2(i), i.e., $u$ equals $x^{\to l}$ where $x$ is the upper path companion of $l$. Since $u$ cannot equal $x$ it must be the lower path companion of $l$ and therefore be $T^*$-adjacent to $s(l)$. As $l$ is not $T$-adjacent to $x$ this contradicts Observation 1. Obviously $u$ cannot comply with Definition 2(ii) as in this case $u$ is $T$-adjacent and therefore not equal to the only path companion of $l$. Also the remaining case that $u$ complies with Definition 2(iii) is easily ruled out with the help of Observation 2 since $u$ is $T^*$-adjacent to some leaf of $T$. □

## 3.4 Counting the Companions

In this section we will show that there are at least $\frac{3}{2}|C|$ companions which implies the desired approximation factor of $\frac{5}{3}$ for LOSTLIGHT.

To this end consider the family $\mathcal{I} := \{I_{\mathrm{a}}(l), I_{\mathrm{p}}(l) \mid l \in C\}$ of all companion sets $I_{\mathrm{p}}(\cdot)$ and $I_{\mathrm{a}}(\cdot)$. Our goal is to count the number of all companions $I_{\mathrm{c}} := \bigcup \mathcal{I}$. Of course it is not sufficient to consider the sum $\sum_{J \in \mathcal{I}} |J|$ of cardinalities of companions sets since these sets need not be disjoint. In order to overcome these difficulties, we introduce for each companion $u \in I_{\mathrm{c}}$ the *frequency* $\mathrm{freq}(u) := |\{J \in \mathcal{I} \mid J \ni u\}|$ of $u$ within the family $\mathcal{I}$ and a weight $w(u) := \frac{1}{\mathrm{freq}(u)}$. This weight is extended to covered leaves $l$ by setting $w(l) := w(I_{\mathrm{p}}(l) \cup I_{\mathrm{a}}(l))$. The weight $w(l)$ may be regarded as the *contribution* of the companions of $l$ to the total number $|I_{\mathrm{c}}|$ of companions. Indeed, we can express $|I_{\mathrm{c}}|$ as the sum $w(C) = \sum_{l \in C} w(l)$ of weighted covered leaves.

In the sequel we will prove a series of lemmas which bound the values $\mathrm{freq}(\cdot)$ and $w(\cdot)$ and form the main ingredients of our final result. Let $l$ be a covered leaf. If $P(l)$ has length 2 we call $l$ a *short leaf*. If $P(l)$ has length at least 3 we call $l$ a *long leaf*.

**Lemma 4** *For any* (*path or auxiliary*) *companion* $u$, *we have* $\text{freq}(u) \leq 2$.

*Proof* Consider first the case $u$ is a path companion of some covered leaf $l$. Since the sets $I_{\text{p}}(\cdot)$ are pairwisely disjoint $u$ has no further occurrences as a path companion in $\mathcal{I}$. So assume that $u$ is an auxiliary companion for covered leaves $l', l''$. We claim that $l' = l''$: It follows from Lemma 3(i) that $l' \neq l$ and $l'' \neq l$. Let us assume further that $u$ is the *upper* path companion of $l$. Then $u$ is $T^*$-adjacent to $l$ and hence $s(l') = l = s(l'')$ by Lemma 3(ii). Thus $l' = l''$ by Lemma 2. If $u$ is the *lower* path companion of $l$ then $u$ is $T^*$-adjacent to $s(l)$ and hence $l' = s(l) = l''$.

Now consider the case that $u$ is not a path companion. Moreover, let us assume that $u$ is an auxiliary companion of some covered leaf $l$. Then $u$ is $T$-adjacent to some path companion of $l$. Since $u$ has degree $d_T(u) = 2$ and the path companion sets $I_{\text{p}}(\cdot)$ are pairwisely disjoint we deduce that there are at most two such covered leaves. □

The following lemma is an immediate consequence:

**Lemma 5** *For any long leaf* $l$, *we have* $w(l) \geq 1$.

**Lemma 6** *For any short leaf* $l$ *with path companion* $u$, *we have* $\text{freq}(u) = 1$ *and* $w(l) \geq \frac{3}{2}$.

*Proof* Since $u$ is $G$-adjacent to two $T$-leaves $u$ cannot be an auxiliary companion and therefore $\text{freq}(u) = 1$.

We claim that $l$ has at least one auxiliary companion. Note that $u$ is in particular the upper path companion of $l$ and therefore any of the three cases in Definition 2 may apply to $l$. We assume first that $P(l)$ is a subpath of $T$. Since $T$ is not a path $u$ must be a branching. This implies $u = b(l)$. Let $x$ be the father of $l$ in $T^*$. Then $l$ is not $T$-adjacent to $x$ as $l$ is a $T$-leaf with $T$-neighbor $u \neq x$. Hence $u^{\to x}$ is an auxiliary companion of $l$ and therefore $w(l) \geq \frac{3}{2}$.

Now consider the case that $P(l)$ is not a subpath of $T$. Then $u$ is $T$-independent from some $l' \in \{l, s(l)\}$. But this implies that $u^{\to l'}$ is an auxiliary companion of $l$ and therefore $w(l) \geq \frac{3}{2}$. □

Let us denote by $C_{=\alpha}$ and $C_{\geq\alpha}$ the sets of covered leaves $l$ with $w(l) = \alpha$ and $w(l) \geq \alpha$, respectively. Lemma 4 shows that the weights of covered leaves are always multiples of $\frac{1}{2}$. Since we would like to have $w(C) \geq \frac{3}{2}|C|$, leaves of weight $w(l) = 1$ could be problematic. But fortunately we can show that those light covered leaves are compensated by heavy ones:

**Lemma 7** *We have* $|C_{=1}| \leq |C_{\geq 2}|$.

*Proof* Let $l$ be a covered leaf with weight $w(l) = 1$. According to Lemma 6 it must be a long leaf. Since $l$ has already two path companions with weight at least $\frac{1}{2}$ it cannot have an auxiliary companion of $l$. Consider the upper path companion $u$ of $l$. If $u$ was not $T$-adjacent to $l$ then $l$ would have an auxiliary companion due to Definition 2(i). We conclude that $l$ is $T$-adjacent to $u$. Moreover, $w(u) = \frac{1}{2}$ by premise. This

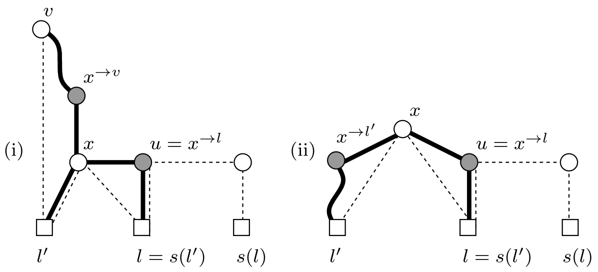

**Fig. 4** The illustration shows the two cases for an upper path companion $u$ of a long leaf where $w(u) = \frac{1}{2}$. *Thick* and *dashed straight lines* mark the edges of $T$ and $T^*$, respectively. *Wiggly lines* depict paths in $T$

implies that $u$ is auxiliary companion of some covered leaf $l' \neq l$ and thus $l = s(l')$ by Lemma 3(ii). Since $u$ is $T$-adjacent to leaf $l$ we can rule out that $u$ complies with Definition 2(iii). Let $x$ be the upper path companion of $l'$. Then either $u = x^{\to l'}$ or $u = x^{\to s(l')}$ according to Definition 2(i) and (ii). Since $x \neq l$ only the latter case may occur which implies that $l'$ is a short leaf.

We will now show that $w(l') \geq 2$ which completes the proof since $l'$ is uniquely determined by $l$ through the relation $s(l') = l$. According to Lemma 6 leaf $l'$ has the unique path companion $x$ of weight $w(x) = 1$. Since $l'$ has also the auxiliary companion $u$ with weight $w(u) \geq \frac{1}{2}$ it suffices to identify a second auxiliary companion for $l'$. To this end assume first that $l'$ is $T$-adjacent to $x$ (confer Fig. 4(i)). Then $x$ must be a branching in $T$, for otherwise, $s(l)$ would be disconnected from $T$. Let $v$ be the $T^*$-father of $l'$. Then $(l', v) \notin E(T)$. Hence $x^{\to v} \neq u$ is an additional auxiliary companion of $l'$. Let us finally assume that $l'$ is not $T$-adjacent to $x$ (confer Fig. 4(ii)). Then $x^{\to l'} \neq u$ is an additional auxiliary companion of $l'$ and therefore $w(l') \geq 2$. □

From Lemmas 4–7 we infer immediately:

$$\begin{aligned}
\left|I(T)\right| &\overset{\text{Lm. 4}}{\geq} w(C_{=1}) + w(C_{=\frac{3}{2}}) + w(C_{\geq 2}) \\
&\geq |C_{=1}| + \frac{3}{2}|C_{=\frac{3}{2}}| + 2|C_{\geq 2}| \\
&= \left(|C_{=1}| + \frac{1}{2}|C_{\geq 2}|\right) + \frac{3}{2}|C_{=\frac{3}{2}}| + \frac{3}{2}|C_{\geq 2}| \\
&\overset{\text{Lm. 7}}{\geq} \frac{3}{2}|C_{=1}| + \frac{3}{2}|C_{=\frac{3}{2}}| + \frac{3}{2}|C_{\geq 2}| \\
&= \frac{3}{2}|C|.
\end{aligned}$$

And finally by inequality (1):

**Theorem 1** LOSTLIGHT *is a* $\frac{5}{3}$*-approximation algorithm for the* MAXIMUM INTERNAL SPANNING TREE *problem with running time* $O(|V|^3)$.

## 4 The Weighted Case

In this section we investigate the MAXIMUM WEIGHTED INTERNAL SPANNING TREE problem (MAXWIST). Salamon [9] developed a local search algorithm, called

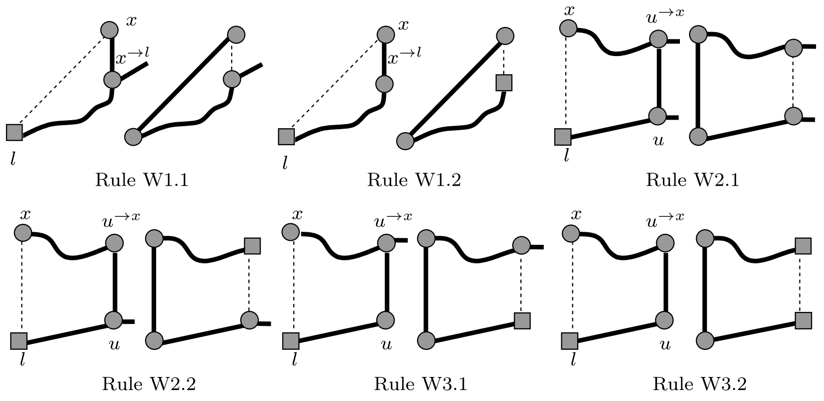

**Fig. 5** The rules of WLOSTADVANCED. *Thick* and *dashed lines mark* $T$-edges and non-tree edges, respectively. *Wiggly lines* depict paths of $T$

WLOST, which runs in polynomial time and gives a $(2 \cdot \Delta(G) - 3)$-approximation for MAXWIST. We will present the first constant-factor approximation algorithm for this problem. Our result is based on a new pseudo-polynomial local search algorithm WLOSTADVANCED which is a 3-approximation algorithm for MAXWIST. This result can then be extended to an efficient $(3 + \epsilon)$-approximation scheme.

Our algorithm WLOSTADVANCED maintains a spanning tree $T$. Let $l$ be a $T$-leaf, $u$ its unique $T$-neighbor and $(l, x)$ a non-tree edge for some node $x$. We introduce the following six rules (confer Fig. 5):

Rule W1.1 If $x^{\to l}$ is a branching then add edge $(l, x)$ and remove $(x, x^{\to l})$.
Rule W1.1 If $x^{\to l}$ has $T$-degree 2 and $c(x^{\to l}) < c(l)$ then add $(l, x)$ and remove $(x, x^{\to l})$.
Rule W2.1 If $u$ and $u^{\to x}$ are branchings then add $(l, x)$ and remove $(u, u^{\to x})$.
Rule W2.2 If $u$ is a branching, $u^{\to x}$ has $T$-degree 2 and $c(u^{\to x}) < c(l)$ then add $(l, x)$ and remove $(u, u^{\to x})$.
Rule W3.1 If $u^{\to x}$ is a branching, $u$ has $T$-degree 2 and $c(u) < c(l)$ then add $(l, x)$ and remove $(u, u^{\to x})$.
Rule W3.2 If $u$ and $u^{\to x}$ have $T$-degree 2 and $c(u) + c(u^{\to x}) < c(l)$ then add $(l, x)$ and remove $(u, u^{\to x})$.

Again we start with an arbitrary spanning tree of the input graph and apply Rules W1.1–W3.2 until we reach a local optimum. Similar to LOSTLIGHT all rules of WLOSTADVANCED improve the objective function $c(I(T))$. Moreover, all rules can be tested and performed in $O(n^2)$. Multiplying all node weights with the smallest common denominator we may assume that they are integral. Hence we obtain a pseudo-polynomial running time $O(n^2 \cdot c(G))$.

In what follows we show that any local optimum computed by WLOSTADVANCED is a 3-approximation. To this end let $T$ be a local and $T^*$ be a global optimum. Furthermore, consider the set $S := L(T) - L(T^*)$ of so called *superfluous leaves*. The obvious relation $I(T^*) \subseteq I(T) \cup S$ then yields

$$\frac{c(I(T^*))}{c(I(T))} \leq 1 + \frac{c(S)}{c(I(T))}. \tag{2}$$

Our goal is to show that the weight of superfluous leaves does not become too large in comparison to the weight of all internal nodes of $T$. More formally we show that $c(S) \le 2 \cdot c(I(T))$ which yields the desired factor of 3. As in the unweighted case this is achieved by furnishing each superfluous leaf $l$ with a suitable set $I(l)$ of *companions* which are internal nodes of $T$.

Let us choose an arbitrary internal node $r$ of $T$ and consider $T^*$ as $r$-rooted. Now let $l$ be an arbitrary superfluous leaf. Since $l$ is an internal node of $T^*$ it has at least one child in $T^*$. We pick an arbitrary child and denote it by $s(l)$.

**Definition 3** (Companions) Let $l$ be a superfluous leaf. Then the set of *companions of $l$* contains the following nodes:

(i) Node $s(l)^{\to l}$ if $l$ is not $T$-adjacent to $s(l)$.
(ii) Node $s(l)$ if $s(l)$ has degree $d_T(s(l)) = 2$ and is $T$-adjacent to $l$.
(iii) Node $u := s(l)^{\to v}$ if $v$ is the father of $l$ in $T^*$, $l$ is $T$-adjacent to $s(l)$ and $u$ has degree $d_T(u) = 2$.

**Lemma 8** *Any companion $u$ of a leaf $l$ has degree $d_T(u) = 2$ and is either equal or $T$-adjacent to $s(l)$.*

*Proof* The fact that $u$ has degree $d_T(u) = 2$ follows from Rule W1.1 in Case (i) and is directly stated in Definition 3 in the two other cases. The other claim also immediate from the definition. □

The following two lemmas help bound from below the total weight of companions.

**Lemma 9** *No node is companion of more than two superfluous leaves.*

*Proof* Let $u$ be a companion and assume that it is companion of at least three superfluous leaves. Due to Lemma 8 we can conclude the existence of three different $T$-leaves $l, l', l''$ such that $u$ is the companion of these leaves, is equal to $s(l)$, and has exactly two $T$-neighbors $s(l')$ and $s(l'')$. Obviously $u$ is a companion of $l$ according to Definition 3(ii) and is thus $T$-adjacent to $l$. Therefore, we may assume w.l.o.g. that $s(l')$ is the unique $T$-neighbor of $u$ which does *not* equal $l$. Since $u$ has degree 2 in $T$ we have $u \ne s(l')^{\to l'}$. Hence $u$ is a companion of $l'$ according to Definition 3(iii) which implies that $l'$ is $T$-adjacent to $s(l')$. Node $s(l')$ must be a branching in $T$, for otherwise, $l''$ would be disconnected from $T$. Let $v$ be the $T^*$-father of $l'$. It follows from $u = s(l')^{\to v}$ that either $v = u$ or $v = l$. Since in both cases Rule W1.1 is applicable this is a contradiction. □

**Lemma 10** *The weight of a superfluous leaf never exceeds that of its companions.*

*Proof* Let $l$ be a superfluous leaf. We distinguish the cases of Definition 3:

If $l$ is not $T$-adjacent to $s(l)$ then $s(l)^{\to l}$ is a companion. If the weight of this node was smaller than that of $l$ then Rule W1.2 would be applicable.

Now let $l$ be $T$-adjacent to $s(l)$ and $v$ be the $T^*$-father of $l$. Assume that we add $(v, l)$ to $T$ and remove $(s(l), s(l)^{\to v})$ from it. In doing so, a node of $T$ becomes a newly created leaf if and only if it had degree two in $T$ and was incident with the edge removed. Consequently those new leaves are exactly the companions of $l$. It is easy to observe that one of the Rules W2.1–W3.2 would be applicable if the weight of the companions was smaller than that of $l$. □

Above two lemmas immediately imply that $c(S) \leq 2 \cdot c(I(T))$. Hence every local optimum of WLOSTADVANCED is a 3-approximation for MAXWIST. Combining this with recent results [7] we are now able to prove the following approximability result.

**Theorem 2** MAXWIST *can be approximated within a factor of* $3 + \epsilon$ *for any* $\epsilon > 0$ *in polynomial time.*

*Proof* Orlin et al. [7] prove the following fundamental result (confer Theorem 4.3) on local search: If there is a neighborhood $N$ for a combinatorial optimization problem $\Pi$ such that every local optimum with respect to $N$ is an $\alpha$-approximation and $N$ is efficiently searchable then $\Pi$ is efficiently approximable within $\alpha + \epsilon$ for any $\epsilon > 0$. There are several restrictions imposed on $\Pi$ and $N$: Each problem instance of $\Pi$ must be a triple $(H, F, c)$ where $H$ is a set of *ground elements*, $F \subseteq 2^H$ a set of *feasible solutions* and $c\colon H \to \mathbb{Q}^+$ a *cost function*. The *objective function* is given by $c(S)$ for any feasible solution $S \in F$. Moreover, it is required that if $(H, F, c)$ is a problem instance for $\Pi$ then so is $(H, F, \lambda \cdot c)$ for any $\lambda \in \mathbb{Q}^+$. Finally the neighborhood $N$ must not depend on the function $c$ but only on $H$ and $F$.

In fact MAXWIST can be stated as a combinatorial optimization problem in the above sense: The ground elements are the nodes and edges of $G$. A feasible solution (spanning tree) is represented by its edges and internal nodes. The function $c$ is already defined in MAXWIST in form of the node weights and can be extended to all ground elements by setting $c(e) = 0$ for any edge $e$. In order to make the neighborhood independent from $c$ we may simply disregard the conditions on the node weights in Rules W1.1–W3.2 which leaves the neighborhood efficiently searchable. Moreover, any local optimum to this neighborhood is a 3-approximation as shown above. Hence we may apply the theorem of Orlin et al. which shows the claim. □

## 5 Tight Worst Case Instances

For the unweighted case consider graph $G_k$ shown in Fig. 6 where $k$ is a positive integer. The graph is constructed as follows. Introduce $k$ node sets, called components, $V_1, \ldots, V_k$ where $V_i = \{v_{i1}, \ldots, v_{i6}\}$ for $i = 1, \ldots, k$. We add three nodes $u_1, u_2$ and $u_3$ completing the node set of $G_k$. Note that $G_k$ has $6k + 3$ nodes. For each component $V_i$, $i = 1, \ldots, k$ we introduce the edges $(v_{i1}, v_{i2})$, $(v_{i3}v_{i2})$, $(v_{i4}v_{i2})$, $(v_{i6}v_{i2})$, $(v_{i3}v_{i5})$, $(v_{i4}v_{i5})$, $(v_{i6}, v_{i5})$ and $(v_{i1}, u_2)$. For each $i = 1, \ldots, k - 1$ we introduce the edges $(v_{i6}, v_{i+1,6})$ and $(v_{i2}, v_{i+1,5})$. Finally, we add the edges $(u_1, u_2)$, $(u_2, u_3)$ and $(u_3, v_{15})$. This completes the description of $G_k$.

Let $T_k$ and $T'_k$ be the two spanning trees of $G_k$ shown in Fig. 6 where thick edges represent the respective tree edges (non-tree edges of $G_k$ are dashed). It is not hard

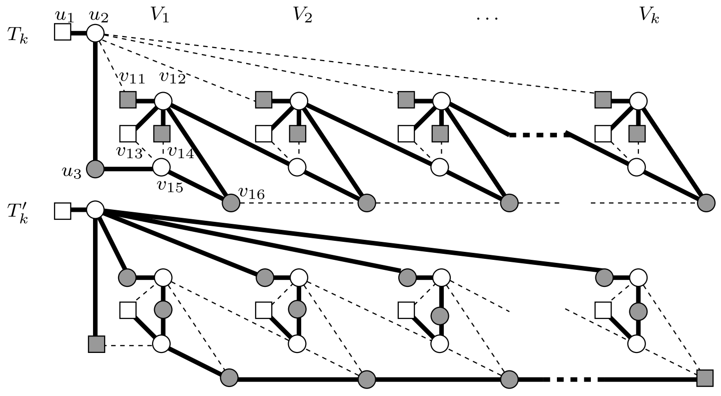

**Fig. 6** Two spanning trees of the same graph $G_k$. *Thick edges* are tree edges while *dashed edges* are non-tree edges. The spanning tree $T_k$ is locally optimal with respect to LOSTLIGHT and WLOSTADVANCED. The tree $T'_k$ achieves a deviation close to the upper bounds. In the weighted case the *grey nodes* are heavy in comparison to the white ones

to verify that $T_k$ is a DFS-tree which is locally optimal with respect to LOSTLIGHT since none of Rules 1–3.2 is applicable. $T_k$ has $3k + 2$ internal nodes. On the other hand the spanning tree $T'_k$ has $5k$ internal nodes. Increasing $k$ we come arbitrarily close to factor $\frac{5}{3}$. It is worth noting that $T_k$ is even locally optimal for Salamon's algorithm LOST. Thus that algorithm does not perform better in the worst case although it involves far more rules than LOSTLIGHT does.

For the weighted case consider the same graph where each component contains three grey nodes weighted with a large number $\Omega$. White nodes are unit weighted. The spanning tree $T_k$ is locally optimal with respect to WLOSTADVANCED (and also with respect to Salamon's algorithm WLOST). The approximation factor converges to 3 with increasing $\Omega$ and $k$. We conclude that our analyses of the algorithms LOSTLIGHT and WLOSTADVANCED are best possible.

**Acknowledgements** We thank Hartmut Noltemeier and Hans-Christoph Wirth for helpful discussions and the anonymous reviewers for helpful suggestions to improve the paper.

## References

1. Binkele-Raible, D., Fernau, H., Gaspers, S., Liedloff, M.: Exact and parameterized algorithms for max internal spanning tree. Algorithmica **65**(1), 95–128 (2013)
2. Chimani, M., Spoerhase, J.: Approximating spanning trees with few branches. In: Proc. 10th International Workshop on Approximation and Online Algorithms (WAOA'12), pp. 30–41 (2012)
3. Flandrin, E., Kaiser, T., Kuzel, R., Li, H., Ryjácek, Z.: Neighborhood unions and extremal spanning trees. Discrete Math. **308**(12), 2343–2350 (2008)
4. Fomin, F.V., Gaspers, S., Saurabh, S., Thomassé, S.: A linear vertex kernel for maximum internal spanning tree. J. Comput. Syst. Sci. **79**(1), 1–6 (2013)
5. Knauer, M., Spoerhase, J.: Better approximation algorithms for the maximum internal spanning tree problem. In: Proc. 11th Algorithms and Data Structures Symposium (WADS'09), pp. 459–470 (2009)
6. Lu, H.I., Ravi, R.: The power of local optimization: approximation algorithms for maximum-leaf spanning tree. In: Proc. 30th Annual Allerton Conference on Communication, Control and Computing, pp. 533–542 (1992)

7. Orlin, J.B., Punnen, A.P., Schulz, A.S.: Approximate local search in combinatorial optimization. SIAM J. Comput. **33**(5), 1201–1214 (2004)
8. Prieto, E., Sloper, C.: Reducing to independent set structure—the case of $k$-internal spanning tree. Nord. J. Comput. **12**(3), 308–318 (2005)
9. Salamon, G.: Approximating the maximum internal spanning tree problem. Theor. Comput. Sci. **410**(50), 5273–5284 (2009)
10. Salamon, G.: A survey on algorithms for the maximum internal spanning tree and related problems. Electron. Notes Discrete Math. **36**, 1209–1216 (2010)
11. Salamon, G., Wiener, G.: On finding spanning trees with few leaves. Inf. Process. Lett. **105**, 164–169 (2008)