Theory Comput Syst
DOI 10.1007/s00224-014-9556-6

Approximating Spanning Trees with Few Branches

Markus Chimani · Joachim Spoerhase

Received: 9 January 2013 / Accepted: 12 May 2014

Abstract Given an undirected, connected graph, the aim of the *minimum branch-node spanning tree* problem is to find a spanning tree with the minimum number of nodes of degree larger than 2. The problem is motivated by network design problems where junctions are significantly more expensive than simple end- or through-nodes, and are thus to be avoided. Unfortunately, it is NP-hard to recognize instances that admit an objective value of zero, rendering the search for guaranteed approximation ratios futile. We suggest to investigate a *complementary* formulation, called *maximum path-node spanning tree*, where the goal is to find a spanning tree that maximizes the number of nodes with degree at most two. While the optimal solutions (and the practical applications) of both formulations coincide, our formulation proves more suitable for approximation. In fact, it admits a trivial $1/2$-approximation algorithm. Our main contribution is a local search algorithm that guarantees a ratio of $6/11$, as well as showing that the problem is APX-hard, i.e., it does not allow a PTAS.

Keywords Theory of computation · Design and analysis of algorithms · Approximation algorithms analysis · Routing and network design problems

This research started from an open problem posed at Dagstuhl Seminar 11191. The authors are grateful for Dagstuhl's support.

Markus Chimani was funded by a Carl-Zeiss-Foundation juniorprofessorship

M. Chimani
Institute of Computer Science, Friedrich-Schiller-University, Jena, Germany
e-mail: markus.chimani@uni-jena.de

J. Spoerhase (✉)
Institute of Computer Science, University of Würzburg, Würzburg, Germany
e-mail: joachim.spoerhase@uni-wuerzburg.de

1 Introduction

Let G be an undirected, connected graph. Finding a spanning tree T of G with the smallest number of *branch nodes* (nodes of degree at least 3) is known as the *minimum branch-node spanning tree* problem (MBST). It was introduced by Gargano et al. [8], where it is pointed out that MBST can also be seen as a natural optimization version of the Hamiltonian path problem since any spanning tree without branch nodes is a Hamiltonian path and vice versa.

Practical applications of MBST can be found in the design of optical networks. Gargano et al. [8] consider *light trees* as a means to realize all-optical multicast (sending information from one source to multiple sinks). Classical wavelength division multiplexing technology (WDM) supports only unicast connections (i.e., light paths). In a WDM network, multicast can therefore only be achieved by establishing multiple unicast connections leading to a decreased performance. Light trees rely on a new technology—light-splitting switches—capable of replicating light signals by splitting light. These sophisticated and thus expensive switches have to be placed at exactly the branch nodes of the light tree, which naturally leads to MBST.

Gargano et al. [8] give bounds on the minimum number of branch nodes depending on density conditions and on other graph parameters such as connectivity, independence number, and length of longest paths. In later works [6, 7], sufficient density conditions for the existence of *spanning spiders*—spanning trees with at most one branch node—are considered.

Salamon [17] gives a different practical motivation for MBST, which is also based on optical networks. Moreover, he provides logarithmic upper bounds on the required number of branch nodes depending on the total number of nodes and the density of the input graph. Cerulli et al. [4] develop ILP formulations for MBST which allow them to solve small instances to optimality. For larger instances they propose three different heuristic approaches. Silva et al. [19] propose heuristic algorithms for MBST based on an iterative refinement idea. For further works on MBST we refer to [2, 3, 15].

In this paper, we are interested in algorithms for MBST with provable quality guarantees. Unfortunately, the above problem formulation is not suitable for analyzing approximation algorithms, due to its close relation to the NP-hard Hamiltonian path problem. It is already NP-hard to decide whether a given instance allows a "perfect" solution, i.e., a solution without any branch node. The main problem arises from the observation that such a solution, a Hamiltonian path, corresponds to an objective value $OPT = 0$. These facts not only show that approximating MBST within a bounded ratio is NP-hard [8], but that if $OPT = 0$ then the ratio of *every* suboptimal solution to the optimum one is unbounded and therefore equally bad. It would, however, be more reasonable to assign spanning trees with fewer branch nodes a better approximative performance than ones with more branch nodes.

We consider the *complementary* formulation of MBST whose objective function is the number of *path nodes*. Put formally, we aim at finding a spanning tree of the input graph that *maximizes* the number of nodes with degree at most 2. We denote such low-degree nodes as *path nodes*, as they are exactly the nodes that arise in paths, and consequently call this reformulation *maximum path-node spanning tree* (MPST). Although the original and our formulation of the problem lead to the same optimum solutions (and have the same practical applications), ours is more appropriate for analyzing approximation algorithms. In fact, every spanning tree is a $1/2$-approximation for MPST since at least half the nodes have degree 1 or 2 [17].

Let N be a function that maps each tree to a certain subset $N(T)$ of its nodes. Consider the optimization problem Π that aims at finding a spanning tree T for a given graph G so that $|N(T)|$ is minimized. Now consider the problem Π' that asks for a spanning tree T of G that *maximizes* $|V \setminus N(T)|$. We say that Π' is complementary to Π and vice versa. Note that complementary problem pairs lead to the same optimum solutions but may behave differently with respect to their approximability. MBST and MPST are in fact complementary to each other according to this definition.

Note that other complementary problem pairs have also been considered in the literature. For example, Lu an Ravi [12] show that there is no constant-factor approximation algorithm for the problem of finding a spanning tree with a minimum number of leaves. Later, the complementary problem of maximizing the number of internal nodes has been suggested in [14] and lead to a series of improved constant-factor approximation algorithms [16, 18] where $\frac{5}{3}$ is the currently best approximation ratio known [11]. Another complementary problem pair that have received significant attention in the algorithmic literature are the *maximum leaf spanning tree* and *minimum connected dominating set* problems where the currently best factors are 2 [20] and $O(\log n)$ [9], respectively. Finally, the well as the *full-degree spanning tree* problem with best possible approximation factor $\sqrt{n}$ [1] and its complementary problem with the currently best factor 3 [10] have been investigated.

There are several local search approximation algorithms for spanning tree problems known in the literature [10–12]. One of the probably most influential works in this context is the additive 1-approximation of the maximum degree spanning tree problem by Fürer and Raghavachari [5].

Our Contribution. We propose an approximation algorithm for the maximum path-node spanning tree problem. The algorithm is based on local search and has a ratio of $6/11$. This beats the (trivial) approximation factor of $1/2$, which is often a critical bound for combinatorial optimization problems (e. g., vertex cover, k-center, maximum leaf spanning tree, full-degree spanning tree, etc.). We also show that constant approximation ratios are the best we can hope for, by showing that the problem is APX-hard.

The paper is organized as follows. In Section 2 we describe our local search approximation algorithm. The analysis of the performance guarantee of the algorithm constitutes the main body of this work and is presented in Section 3. We make some observations concerning the tightness of our analysis in Section 4, and show the APX-hardness in Section 5.

We also argue that there is no *asymptotic* approximation algorithm for the minimization formulation MBST with a ratio of the form $\alpha(n)\mathrm{OPT} + o(n)$ unless

P = NP. This is an even stronger justification of our formulation as a maximization problem.

2 Algorithm

As pointed out above every spanning tree of the input graph is already a $1/2$-approximation since at least one half of the nodes in a tree have degree 1 or 2. The worst case of ratio $1/2$ occurs when the considered spanning tree consists of only leaves and cubic nodes (nodes with degree 3) while the optimum is a Hamiltonian path. (Note that a star with a single high-degree node is only slightly worse than a Hamiltonian path.) We will improve this ratio following the idea that we can obtain a ratio better than $1/2$ if we can ensure that a certain fraction of the nodes that are non-cubic (which means that their degree is different from 3) in the optimum are also non-cubic in our solution.

In what follows, G is the input graph and T is a spanning tree of G. We denote the edge set of a graph H by $E(H)$. If H is a subgraph of G and $E' \subseteq E(G)$, then we denote by $H + E'$ the subgraph obtained from H by adding E' to the edge set of H. Similarly, we define $H - E''$ as the graph obtained by removing edges $E'' \subseteq E(H)$ from H.

A *feasible k-flip* replaces k edges of a spanning tree T with the same number of edges in $E(G) \setminus E(T)$ such that the resulting graph T' is a spanning tree of G. A k-flip is *improving* if it either increases the number of path nodes (degree 1 or 2), or if the number of path nodes remains constant while the number of cubic nodes (degree 3) decreases. For simplicity, a $\bar{k}$-flip denotes any ℓ-flip with $1 \leq \ell \leq k$.

Algorithm Starting with an arbitrary spanning tree, perform improving $\bar{2}$-flips as long as there exist any. ◇

The main part of this paper is devoted to show that this algorithms guarantees a solution with at least 6/11 of the optimum solution's path nodes. Before we do so, we show:

Lemma 1 *The above algorithm runs in polynomial time. In particular, it only requires $O(|V(G)|)$ flips.*

Proof It is clear that checking for improving $\bar{2}$-flips can be done in polynomial time (e.g., by evaluating all $O(|E(G)|^4)$ possible $\bar{2}$-flips). Furthermore, the definition of improving flips gives rise to a lexicographical objective function of two parts: the most significant part is n_P, the number of path nodes where more is better; to compare solutions with equal n_P, we consider the number of cubic nodes n_C where less is better. Any improving flip either increases n_P (while possibly also increasing n_C) or leaves n_P unchanged and decreases n_C. Since both measures are non-negative and bounded by $|V(G)|$ from above, a trivial upper bound of $O(|V(G)|^2)$ iterations follows. We can improve on this by realizing that a flip increasing n_P cannot increase n_C arbitrarily: Since any $\bar{2}$-flip touches at most $8 = O(1)$ nodes, the sum of all increases of n_C is bounded by $O(|V(G)|)$, and hence our algorithm performs at most $O(|V(G)|)$ flips overall. □

3 Analysis of Approximation

In the following, T always denotes a locally optimal spanning tree for G obtained by our algorithm, and we partition the nodes into subsets based on their degree in T: L denotes the set of leaves in T (T-leaves), F the set of degree-2-nodes in T (*forward nodes*[1]), $P = L \cup F$ the set of path nodes, C the set of degree-3 (*cubic*) nodes, and H the set of nodes with degree ≥ 4 in T (*high-degree nodes*). Generally, we may use graph theoretic terms with preceding graph specification to specify whether we consider the whole graph G or only T. E.g., we use the term T*-neighbors* to denote neighbors of a node in T. Let T^* be a spanning tree in G with a maximum number of path nodes, and $C' \subseteq C$ be the set of cubic nodes in T that are non-cubic in T^*. We observe that the set $C' \cup H$ contains all nodes (if any) that are path nodes in T^* but not in T.

3.1 Overview

The nodes $C \cup H$ are the branch nodes in T. Observe that $|L| = 2 + \sum_{v \in C \cup H}(\deg(v) - 2)$ where $\deg(v)$ denotes the degree of a node v in T. Intuitively, this can be seen by comparing T to a simple spanning path. The latter has precisely 2 leaves. For every branch node v in T, the number of leaves increases by $\deg(v) - 2$. Hence, we can think of each branch node v as contributing $\deg(v) - 2$ leaves to the tree. See [3] for a formal proof. Rearranging the above equation yields

$$|L| = 2 + |C \cup H| + \sum_{v \in H}(\deg(v) - 3) \geq |C' \cup H| + \sum_{v \in H}(\deg(v) - 3)\,.$$

We interpret this inequality so that each branch node in $C' \cup H$ is guaranteed to contribute one *regular* leaf to T and that each node in H contributes $\deg(v) - 3$ *supplementary* leaves to T. More precisely, we associate (in a fixed but arbitrary manner) each branch node v in $C' \cup H$ with one leaf, which we call *regular leaf for v*, and each node $v \in H$ with $\deg(v) - 3$ many further leaves, which we call *supplementary leaves for v*. In doing so, we ensure that no leaf is associated to more than one node and that no leaf is simultaneously a regular and a supplementary leaf. Let L_r be the set of regular leaves and L_s be the set of supplementary leaves.

Assume we can show that the number $|L_s \cup F|$ of supplementary leaves or forward nodes is at least $1/5 \cdot |C' \cup H|$. Together with the $|L_r| = |C' \cup H|$ regular leaves,

[1]The term *forward node* is motivated by the technical background of WDM networks, and, e.g., consistent with the definition of maximum forward-node spanning trees [17].

this implies the existence of at least $6/5 \cdot |C' \cup H|$ path nodes in T. Since the nodes in $C \setminus C'$ are cubic in T^* by definition of C' we have that $\text{OPT} \leq |P| + |C' \cup H|$. Hence the performance guarantee of our algorithm is bounded by

$$\frac{|P|}{\text{OPT}} \geq \frac{|P|}{|P| + |C' \cup H|} \geq \frac{|P|}{|P| + 5/6|P|} = \frac{6}{11}. \tag{1}$$

It remains how to show that $|L_s \cup F| \geq 1/5 \cdot |C' \cup H|$. We consider the following coin transfer scheme. Initially, each node in $C' \cup H$ receives one *coin*. In four stages, described in detail below, each node in $C' \cup H$ transfers its coin to a forward node or a supplementary leaf such that none of these nodes receives more than five coins. Hence it must be the case that $|L_s \cup F| \geq 1/5|C' \cup H|$ which implies the claimed performance ratio of $6/11$ as argued above.

Our transfer scheme will ensure that after Stages I–III the following three properties hold.

(P1) Only nodes in $F \cup H$ hold coins,
(P2) each node in F holds at most five coins, and
(P3) each node $v \in H$ holds at most $\deg(v) + 1$ coins.

In Stage IV we will then transfer the coins from nodes in H to supplementary leaves such that no supplementary leaf receives more than five coins.

3.2 Transfer Scheme – Stage I

For each $u \in C'$ consider its T-neighbors. If any T-neighbor lies in $F \cup H$ then u transfers its coin to this neighbor (breaking ties arbitrarily). It is clear that any node in $v \in F \cup H$ receives at most $\deg(v)$ coins in Stage I.

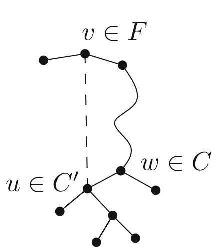
3.3 Transfer Scheme – Stage II

Let $u \in C'$ be a node that still owns its coin after Stage I. This implies that all T-neighbors of u are in $L \cup C$. We can show the following.

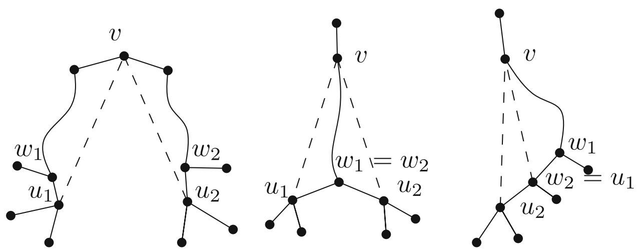
Lemma 2 *Assume $u \in C'$ has only T-neighbors in $L \cup C$ and there exists an edge $(u, v) \in E(G) \setminus E(T)$. Then, $v \in F$.*

Proof See Fig. 1a for an illustration. Assume $v \notin F$. Consider the 1-flip where we add (u, v) to T and remove the unique other edge (u, w) incident to u in T that lies on the so-established cycle. By this operation, $\deg(u)$ does not change, $\deg(v)$ is increased by one, and $\deg(w)$ is decreased by one. Clearly, w was not in L before and hence was a cubic node that becomes a forward node. Since $v \notin F$, v does not become a cubic node but it remains in P or H after the 1-flip. Overall, our flip would increase the number of path nodes, which contradicts the local optimality of T. □

In Stage II we process all nodes u that satisfy the condition of Lemma 2. For each such node we transfer the coin of u to a node $v \in F$ that is G-adjacent to u. According to Lemma 2 such a node always exists.



(a) Lemma 2.



(b) The three cases in the proof of Lemma 3.

Fig. 1 Illustration for Lemmata 2 and 3. Solid edges are edges in T whereas dashed edges are edges in $E(G) - E(T)$

Lemma 3 *No two nodes that are processed in Stage II transfer their coins to the same node.*

Proof See Fig. 1b for an illustration. Assume there are two distinct cubic nodes u_1, u_2 processed in Stage II which assign their coin to some node $v \in F$. Consider adding the two edges (u_1, v) and (u_2, v) to T. The introduction of each of these edges (without introducing the other one) establishes a cycle: let (u_1, w_1) and (u_2, w_2) be the unique other incident edges to u_1 and u_2, respectively, that lie on the respective cycle. Now, consider the 2-flip removing (u_1, w_1) and (u_2, w_2) and instead inserting (u_1, v) and (u_2, v). The remaining structure is clearly still a tree, and we can investigate how the node degrees change. The nodes w_1 and w_2 are cubic nodes, as they cannot be leaves while lying on a cycle, and their neighboring u_1, u_2 were not able to transfer a coin to them in Stage I. Obviously, $\deg(v)$ is increased by 2 and becomes a degree-4 node. By case distinction we show that this 2-flip would always either increase the number of path nodes, or—when this number does not change—decrease the number of cubic nodes; both contradicts the local optimality of T. See Fig. 1b for an illustration of these cases.

- $w_1 \neq w_2$, $w_2 \neq u_1$ and $w_1 \neq u_2$: The nodes u_1, u_2 remain cubic, but the cubic nodes w_1 and w_2 become path nodes.
- $w_1 = w_2$: The nodes u_1, u_2 remain cubic, but the cubic node w_1 becomes a leaf.
- $w_2 = u_1$ or $w_1 = u_2$: Assume, w.l.o.g. $u_1 = w_2$. Then, u_2 remains cubic, but the cubic nodes u_1 and w_1 become path nodes. □

Applying this lemma iteratively, we obtain:

Corollary 1 *Any forward node gains at most one coin in Stage II.*

3.4 Transfer Scheme – Stage III

A node $u \in C'$ is called *unprocessed* if it has not been processed in Stage I or II. This implies that u is T-adjacent only to nodes in $L \cup C$ and moreover, there is no edge $e \in E(G) \setminus E(T)$ incident on u. Since u lies in C' there must be an edge

$e = (u, v) \in E(T) \setminus E(T^*)$, for, otherwise u would be cubic in T^*, too. If there are multiple (at most 2) such edges, we pick one of these as e arbitrarily. We call edge e *critical*. Note that v itself may also be unprocessed, and e therefore critical due to both.

Let V_u, V_v be the node sets of the two connected components obtained by removing a critical edge (u, v) from T (let $u \in V_u$). Since T^* is connected there must be an edge $e^* = (x, y) \in E(T^*)$ with $x \in V_u$ and $y \in V_v$. Of course $x \neq u$ since u has not been processed in Stage II. Note that $T + e^* - e$ is a tree, i.e., the corresponding 1-flip is feasible.

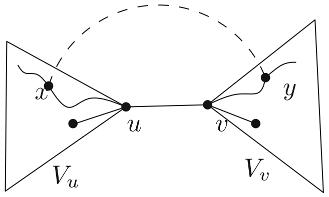
Lemma 4 *At least one of the nodes x, y is a forward node in T.*

Proof See Fig. 2a for an illustration. Assume both x, y are not in F, and consider a 1-flip removing (u, v) and inserting (x, y). If $v = y$, $\deg(v)$ does not change, the cubic node u becomes a path node, and $\deg(x)$ increases by one. Since $x \notin F$ remains in P or in H after the flip, the flip would be improving.

If $v \neq y$, then v cannot have been a T-leaf. Consequently, the cubic nodes u and v become path nodes; the non-path-nodes x, y increase their degree by one and remain in P or in H after the flip. Again, the flip would be improving, which is a contradiction to T's local optimality. □

In Stage III we process all yet unprocessed nodes in C'. Let u be one of these nodes. The idea is that there is at least one node $y \in F$ associated with u according to Lemma 4. We transfer the coin of u to this node. If this is done in an arbitrary manner we cannot rule out that one forward node gains a large number of coins in Stage III. In what follows we develop a refined transfer scheme that ensures that each forward node in T gains *at most two* coins in Stage III.

First, we transfer all coins located at unprocessed nodes to their respective critical edges. This gives at most two coins per critical edge. Then we identify for each critical edge e a *replacement edge* $r(e) \in E(T^*) \setminus E(T)$ such that the 1-flip replacing e with $r(e)$ leads to a tree. According to Lemma 4 one of the endpoints of $r(e)$ is a forward node in T. We will construct the mapping $r(\cdot)$ so that the number of forward nodes incident on replacement edges is greater than or equal to the number of critical



(a) A critical edge and its replacement edge.

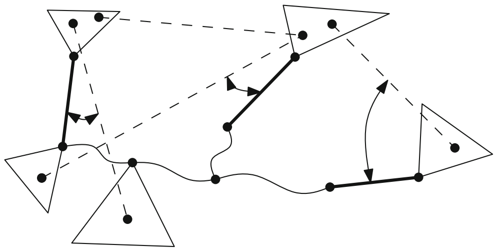

(b) Injective mapping provided by Lemma 5. Critical edges are thick and potential replacement edges are dashed. The arrows show a possible mapping.

Fig. 2 Illustrations for Lemmata 4, 5, and 7

edges. This allows us to transfer the coins from the critical edges to forward nodes incident to their respective replacement edges such that each forward node receives at most two coins.

It remains to show the existence of such a mapping $r(\cdot)$. We first show (see Fig. 2b for an illustration).

Lemma 5 *There exists an injective mapping $r(\cdot)$ from critical edges to replacement edges, that is, $T - e + r(e)$ is a tree and $r(e) \neq r(e')$ for distinct critical edges e, e'.*

Proof Let $e = (u, v)$ be a critical edge and let R_e be the set of edges in $E(T^*) \setminus E(T)$ that cross the cut (V_u, V_v). The edges in R_e are exactly the set of edges e' such that $T - e + e'$ is a tree. Now consider an arbitrary subset E' of critical edges and let $\{V_1, \dots, V_{|E'|+1}\}$ be the connected components of the forest $T - E'$. Since T^* is connected there are at least $|E'|$ edges in T^* connecting distinct components $V_i \neq V_j$. Let E'' be the set of such edges. Note that each edge in E'' lies in R_e for some $e \in E'$ since nodes in different components $V_i \neq V_j$ are separated in T by some critical edge in E'. Hence $|\bigcup_{e \in E'} R_e| \geq |E''| \geq |E'|$. By Hall's marriage theorem there is an injective mapping r as required above. □

In what follows we show that the set of endpoints of replacement edges contains at least as many forward nodes as there are critical edges, whenever we have an injective mapping $r(\cdot)$. We know by Lemma 4 that each replacement edge has a forward node in T as one of its endpoints. We face, however, the problem that two replacement edges might be adjacent leading to multiple countings of forward nodes.

The following lemma shows that two replacement edges can only be adjacent at a forward node if they do not allow a feasible 2-flip.

Lemma 6 *Let e, e' be distinct critical edges. Assume that the 2-flip replacing e, e' with $r(e), r(e')$ leads to a tree. Then $r(e)$ and $r(e')$ are not incident on a common forward node.*

Proof See Fig. 3 for an illustration. Let $e = (u, v)$ and $e' = (u', v')$ be distinct critical edges with $u \neq u'$ as unprocessed nodes and let $r(e) = (x, y)$ and $r(e') = (x', y')$ be their replacement edges. By way of contradiction assume that both replacement edges are incident on forward node $x = x'$.

Now consider the 2-flip replacing e, e' with $r(e), r(e')$. Due to our assumption this 2-flip leads to a tree T'. We show below that this is an improving 2-flip, contradicting the local optimality of T.

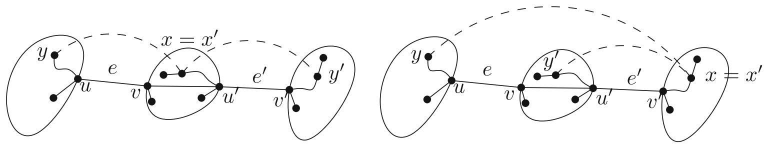


Fig. 3 Two possible configurations in the proof of Lemma 6

To this end let $D = \{u, u', v, v'\}$ be the set of nodes incident on the edges e, e' and let $I = \{x, y, y'\}$ be the set of nodes incident on the edge $r(e), r(e')$. Note that $x \notin D$ since x is a forward node and D contains only leaves and cubic nodes. Furthermore, the 2-flip changes the degree of x from 2 to 4. We distinguish two cases and show that in both cases the flip would be improving.

- D contains at least three cubic nodes: The removal of e and e' transforms at least three cubic nodes in D into path nodes. By the addition of $r(e)$ and $r(e')$, at most three path nodes are destroyed; yet, at most two of these latter nodes (namely y, y') can become cubic.
- D contains less then three cubic nodes: By assumption, u and u' are distinct unprocessed and therefore cubic nodes. Hence, neither v nor v' are cubic. As argued above, u, u' are only T-adjacent to cubic nodes and leaves; therefore v, v' are both T-leaves. Their degree ensures that v and v' must also be distinct. Hence $y = v$ and $y' = v'$ since otherwise one of these nodes would be disconnected by the 2-flip. The removal of e, e' creates two forward nodes (namely u, u'). The addition of $r(e), r(e')$ destroys only x's path node property, as the nodes v, v' remain leaves after the flip.

□

The following two lemmas show that two critical edges can only share a common forward node in T if all of their endpoints are forward nodes. In some sense this compensates the negative effect of "multiple counting".

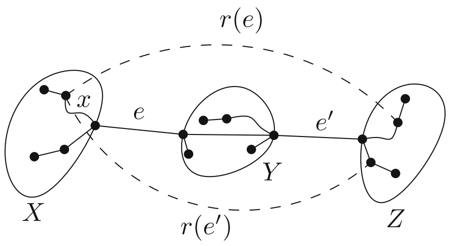
Lemma 7 *Let e be a critical edge and let e^* be an edge in T^* that is not adjacent to e. If the 1-flip replacing e with e^* is feasible then both endpoints of e^* are forward nodes in T.*

Proof See Fig. 2a for an illustration. Let $e = (u, v)$ and let $e^* = (x, y)$. Since e and e^* are not adjacent and the 1-flip replacing e with e^* is feasible the T-path connecting x and y contains u, v as interior nodes. Hence u and v are not leaves in T. Assume w.l.o.g. that u is unprocessed and hence cubic. Then v must also be cubic since u is only adjacent to leaves or cubic nodes. Now, if one of the nodes x, y were not a forward node then the above 1-flip would be improving contradicting the local optimality of T. □

Lemma 8 *Let e, e' be distinct critical edges. Assume that $r(e)$ and $r(e')$ are incident on the same forward node. Then all the three endpoints of $r(e)$ and $r(e')$ are forward nodes.*

Proof See Fig. 4a for an illustration. According to Lemma 6 the 2-flip replacing e, e' with $r(e), r(e')$ in T does not lead to a tree.

If we delete e, e' from T we obtain three connected components X, Y, Z. W.l.o.g. assume that e connects X, Y and e' connects Y, Z. Since the 1-flips replacing e and e' with $r(e)$ and $r(e')$, respectively, are feasible but the above 2-flip is *not* feasible, we conclude that both $r(e)$ and $r(e')$ must connect X and Z.



(a) Proof of Lemma 8.

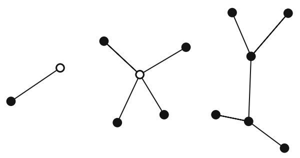

(b) Forest $\mathscr{F}$ of replacement edges. Black nodes are T-forward nodes and white ones are not T-forward.

Fig. 4 Illustrations for Lemma 8 and of the forest of replacement edges

Let x be the forward node shared by $r(e)$ and $r(e')$. Assume w.l.o.g. that x lies in X. Since x is a forward node it is not incident on the critical edge e. Since the other endpoints of $r(e)$ and $r(e')$ both lie in Z but e connects X, Y none of the edges $r(e)$ or $r(e')$ can be adjacent to e.

Now we observe that the 1-flip replacing e with $r(e)$ and the 1-flip replacing e (instead of e') with $r(e')$ are both feasible and satisfy the requirements of Lemma 7. Hence all three endpoints of $r(e)$ and $r(e')$ are forward nodes. □

Now consider the forest $\mathcal{F}$ consisting of all replacement edges. See Fig. 4b for an illustration. Recall that each replacement edge is adjacent to at least one forward node and that if two edges share a forward node then all their endpoints are forward nodes. Hence, each connected component of $\mathcal{F}$ is either a star centered at a non-forward node or a subtree with only forward nodes. Therefore, there are at least as many forward nodes in T as there are replacement edges, which, in turn, equals the number of critical edges. As any critical edge is incident to at most two unprocessed nodes, each unprocessed node can transfer its coin to a forward node so that no forward node gains more than two coins in Stage III.

Corollary 2 *No forward node in T gains more than two coins in Stage III.*

Observation 1 *After Stage III, all coins from C' are transferred to some nodes of $F \cup H$.*

3.5 Transfer Scheme – Stage IV

We now consider the result of Stages I–III and summarize our findings. Observation 1 above established property (P1).

In Stage I we transferred some of the coins located at nodes in C' to nodes in $F \cup H$. Each node in $F \cup H$ gains at most deg(v) additional coins in Stage I. In particular, each node v in H owns at most deg(v) + 1 coins after Stage I and will not receive further coins in the subsequent stages. This establishes property (P3).

In Stages II and III, coins are only transferred from nodes in C' to nodes in F. Since each forward node gains at most 2 coins in Stage I and, according to Corollaries 1 and 2, gains at most three further coins in Stages II and III we conclude that each forward node owns at most 5 coins after the execution of all three stages. This establishes property (P1). To summarize, properties (P1)–(P3) hold after the execution of the first three stages.

Observe that by Property (P1), leaves currently hold no coins. In Stage IV we finally transfer the coins from nodes in H to their corresponding supplementary leaves. To this end, each node $v \in H$ distributes its coins evenly among its supplementary leaves where leaves may receive a fractional amount of coins.

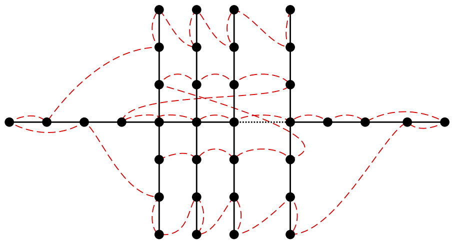
Property (P3) guarantees that after Stage III, each $v \in H$ holds at most $\deg(v) + 1$ coins. Hence, none of the $\deg(v) - 3$ supplementary leaves of v receives more than $(\deg(v)+1)/(\deg(v)-3) \leq 5$ coins. Hence, after Stage IV only nodes in $L_s \cup F$ hold coins and none of these nodes holds more than five coins. As argued in Section 3.1, this implies our main result.

Theorem 1 *Our algorithm approximates the problem of finding a spanning tree with the maximum number of path nodes within a ratio of* $6/11$.

4 Tightness of Approximation

It remains an open problem, whether the ratio proved above is tight. The following two examples show certain structural properties which may be interesting in the future hunt for an answer to this question.

Figure 5a shows an instance where the optimal solution is $|V(G)|$, i.e., the graph allows a Hamiltonian path (dashed red lines) and hence requires no branch nodes at


(a) The optimal solution is a Hamilton path. In general, we consider this graph to have k many vertical paths.

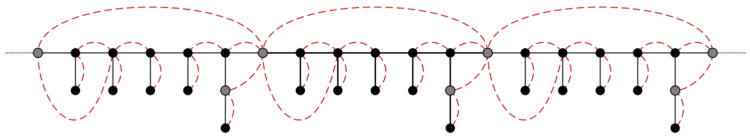

(b) The optimal solution has high-degree nodes. In general, we will repeat the central pattern k times; the figure shows 3 repetitions.

Fig. 5 Bad cases. The solid black edges form a locally optimal solution found by the algorithm, while the dashed red edges form the optimum solution. The graph itself consists of exactly the union of these two edge sets (disregarding multiplicity of edges)

all. Yet, our algorithm may terminate with a tree (solid black lines) containing multiple branch nodes. Observe the repetitive pattern (vertical structures) that may be repeated, say, k times. The depicted solution then has k branch nodes (of degree 4, rather than the degree 3 nodes presumably required for a tight example). Since the overall graph has $7k + 8$ nodes, this example would only suggest a ratio of (asymptotically) at least $6/7$.

Figure 5b shows an instance (in fact, a repetitive pattern) which requires branch nodes. Each pattern has 12 nodes, the figure shows three repetitions. A final instance would consist of k repetitions, with some constant size "caps" on the left and right end of the graph structure. The optimum solution requires 2 branch nodes and hence allows 10 path nodes per pattern, whereas the algorithm may terminate with only 7 path nodes (5 branch nodes) per pattern. Hence, this example would result in a ratio of (asymptotically) $7/10$.

5 Complexity of Approximation

It is natural to try to obtain better and better approximation ratios, simply by considering k-flips for larger values of k. Assuming that such an increase really results in better approximation ratios, the question would be if this strategy can be the starting point of a polynomial approximation scheme. We answer this question in the negative, in that we show that MPST is APX-hard, i.e., it is very unlikely that MPST can be approximated arbitrarily close in polynomial time. In order words, constant approximation ratios are the best we can reasonably hope for.

Theorem 2 *MPST is APX-hard.*

Proof As a starting point we consider the $(1, 2)$-TSP, that is, the traveling salesman problem on n nodes where each edge (in a complete graph) has either weight 1 or 2. This problem is known to be APX-hard [13]. The hardness proof given in [13] reduces 3SAT (with a bounded number of occurences of each literal) to $(1, 2)$-TSP by mapping the 3SAT instance into a graph G_1 with maximum degree $\Delta = 6$. It is stated that a more careful construction would yield the result for $\Delta = 4$; in the following, $\Delta = 6$ will suffice. To obtain a $(1, 2)$-TSP instance G, the edges of G_1 are assigned weight 1, and the graph is augmented to a complete graph by adding weight-2 edges between non-adjacent nodes.

The construction of G_1 is such that positive 3SAT instances are mapped to Hamiltonian graphs, i.e., graphs that contain a tour visiting every vertex exactly once. Hence, the TSP instance admits a TSP tour consisting only of weight-1 edges. On the other hand, negative 3SAT instances are mapped to graphs G in which every TSP tour uses at least $\alpha \cdot n$ weight-2 edges, for some constant $\alpha > 0$.

We use the same reduction as [13] to obtain the graph G_1. If we reduce from a positive 3SAT instance we know that G_1 is Hamiltonian. It hence also contains a Hamiltonian path, that is, a spanning tree consisting of n path nodes. Hence $\mathrm{OPT} = n$ whenever we reduce from a positive 3SAT instance.

Now assume that we reduce from a negative 3SAT instance. Let G_1 be the resulting graph and T be a spanning tree of G_1 with k branch nodes. We want to deduce a solution to the TSP problem from T. First, we iteratively transform T into T' by removing an arbitrary edge incident to a branch node, until all nodes have degree at most 2. Since we remove at most $k(\Delta - 2)$ edges, T' is a spanning forest of G_1 that consists of at most $c := 1 + k(\Delta - 2)$ many paths. Consider the complete graph G that is produced by the reduction of [13] and that contains G_1 and hence T' as spanning subgraphs. We can connect the paths of T' to a simple cycle using at most c many edges of weight at most 2. As remarked above, each TSP tour of G must have at least $\alpha \cdot n$ many weight-2 edges since we reduced from a negative 3SAT instance. Therefore, we have that $c = 1 + k(\Delta - 2) \geq \alpha n$. Assuming that $n \geq 2/\alpha$ (which is a constant) we obtain

$$k \geq \frac{\alpha \cdot n - 1}{\Delta - 2} \geq \frac{\alpha}{2(\Delta - 2)} n \ . \tag{2}$$

Since this inequality holds for all spanning trees we have that OPT $\leq n - k \leq \left(1 - \frac{\alpha}{2(\Delta-2)}\right) n$ whenever we reduce from a negative 3SAT instance.

Using a standard argument we show that there can be no approximation algorithm for MPST with performance better than $\beta := 1 - \frac{\alpha}{2(\Delta-2)} < 1$ unless P $=$ NP. Assume to the contrary that there is an approximation algorithm $\mathcal{A}$ with ratio $\beta' > \beta$. We could decide 3SAT as follows. Take a 3SAT formula F and apply the reduction to obtain a graph G_1. Formula F is satisfiable if and only if the tree obtained by $\mathcal{A}$ has at least $\beta' n$ path nodes. Clearly, if F is satisfiable then OPT $= n$ and therefore $\mathcal{A}$ outputs a solution with at least $\beta' n > \beta n$ path nodes. If F is unsatisfiable, OPT $\leq \beta n < \beta' n$. Therefore our algorithm cannot output a spanning tree with at least $\beta' n$ path nodes. □

Our hardness proof also has an important implication for the minimization version MBST. As we pointed out in the introduction, it is NP-hard to distinguish between instances that have spanning trees with no branch nodes (Hamiltonian paths) and instances without such spanning trees. This implies that there can be no approximation algorithm of finite ratio for MBST. This argument does not rule out, however, that there are algorithms with an asymptotic performance guarantee of the form $\gamma(n)$OPT $+ o(n)$. In fact one may argue that such an algorithm would better respect the cost of MBST than our reformulation as a maximization problem. Using our above reduction, we can show that such an algorithm can not exist. This is an even stronger justification of our problem formulation.

Corollary 3 *There is no approximation algorithm with a ratio* $\gamma(n)$OPT $+ o(n)$ *for MBST unless* P $=$ NP.

Proof Assume there is an algorithm $\mathcal{A}$ with a performance $\gamma(n)$OPT $+ \delta(n)$ where $\delta(n) \in o(n)$. We could decide 3SAT (with a bounded number of occurences of each literal) as follows. Take the 3SAT formula F and apply the reduction of Theorem 2 to obtain a graph G_1. Formula F is satisfiable if and only if the tree found by $\mathcal{A}$ has at most $\delta(n)$ branch nodes: If F is satisfiable then G_1 is Hamiltonian and therefore

OPT $= 0$. Hence $\mathcal{A}$ outputs a solution with at most $\delta(n)$ branch nodes. On the other hand, if F is unsatisfiable every spanning tree of G_1 has at least $\frac{\alpha}{2(\Delta-2)}n \in \Omega(n)$ many branch nodes by inequality (2). Assuming that n is larger than some suitable constant, our algorithm must output a solution with strictly more than $\delta(n) \in o(n)$ branch nodes. $\square$

6 Conclusions

We proposed the *maximum path-node spanning tree* problem as a complementary formulation to the established but approximation-wise impractical *minimum branch-node spanning tree*. Both formulations have the very same practical applications. We showed a conceptually and implementation-wise simple algorithm to tackle this NP-hard problem. The paper's core lies in the proof that this algorithm in fact guarantees an approximation ratio of (at least) 6/11, beating the critical barrier of ratio 1/2. It remains open whether this bound is tight or if an even better ratio can be shown for this algorithm. However, we have shown that no PTAS can exist for the problem. It would also be interesting to investigate whether we can do better with k-flips for a constant $k > 2$.

References

1. Bhatia, R., Khuller, S., Pless, R., Sussmann, Y.J.: The full degree spanning tree problem. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), pp. 864–865 (1999)
2. Carrabs, F., Cerulli, R., Gaudioso, M., Gentili, M.: Lower and upper bounds for the spanning tree with minimum branch vertices. Comp. Opt. Appl. **56**(2), 405–438 (2013)
3. Cerrone, C., Cerulli, R., Raiconi, A.: Relations, models and a memetic approach for three degree-dependent spanning tree problems. Eur. J. Oper. Res. **232**(3), 442–453 (2014)
4. Cerulli, R., Gentili, M., Iossa, A.: Bounded-degree spanning tree problems: models and new algorithms. Comput. Optim. Appl. **42**(3), 353–370 (2009). doi:10.1007/s10589-007-9120-2
5. Fürer, M., Raghavachari, B.: Approximating the minimum degree spanning tree to within one from the optimal degree. In: Proceedings of the Third Annual ACM/SIGACT-SIAM symposium on Discrete Algorithms (SODA'92), pp. 317–324 (1992)
6. Gargano, L., Hammar, M.: There are spanning spiders in dense graphs (and we know how to find them. In: Proceedings 30th International Colloquium on Automata, Languages and Programming (ICALP'03), Lecture Notes in Computer Science, vol. 2719, pp. 802–816 (2003)
7. Gargano, L., Hammar, M., Hell, P., Stacho, L., Vaccaro, U.: Spanning spiders and light-splitting switches. Discret. Math. **285**(1-3), 83–95 (2004). doi:10.1016/j.disc.2004.04.005
8. Gargano, L., Hell, P., Stacho, L., Vaccaro, U.: Spanning trees with bounded number of branch vertices. In: Proceedings 29th Intenational Colloquium on Automata, Languages and Programming (ICALP'02), Lecture Notes in Computer Science, vol. 2380, pp. 355–365 (2002)
9. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica **20**(4), 374–387 (1998)
10. Khuller, S., Bhatia, R., Pless, R.: On local search and placement of meters in networks. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'00), pp. 319–328 (2000)
11. Knauer, M., Spoerhase, J.: Better approximation algorithms for the maximum internal spanning tree problem. In: Proceedings of the 11th Algorithms and Data Structures Symposium (WADS'09), Lecture Notes in Computer Science, vol. 5664, pp. 459–470 (2009)

12. Lu, H.I., Ravi, R.: The power of local optimization: approximation algorithms for maximum-leaf spanning tree. In: Proceedings of the Thirtieth Annual Allerton Conference on Communication, Control and computing, pp. 533–542 (1992)
13. Papadimitriou, C.H., Yannakakis, M.: The traveling salesman problem with distances one and two. Math. Oper. Res. **18**(1), 1–11 (1993)
14. Prieto, E., Sloper, C.: Reducing to independent set structure – the case of k-internal spanning tree. Nord. J. Comput. **12**(3), 308–318 (2005)
15. Rossi, A., Singh, A., Sundar, S.: Cutting-plane-based algorithms for two branch vertices related spanning tree problems. Optim. Eng., 1–33 (2013)
16. Salamon, G.: Approximating the maximum internal spanning tree problem. Theor. Comput. Sci. **410**(50), 5273–5284 (2009)
17. Salamon, G.: Degree-based Spanning tree Optimization. Ph.D. Thesis, Department of Computer Science and Information Theory. Budapest University of Technology and Economics, Hungary (2010)
18. Salamon, G., Wiener, G.: On finding spanning trees with few leaves. Inf. Process. Lett. **105**, 164–169 (2008)
19. Silva, D.M., Silva, R.M.A., Mateus, G.R., Gonçalves, J.F., Resende, M.G.C., Festa, P.: An iterative refinement algorithm for the minimum branch vertices problem. In: Proc. 10th International Symposium on Experimental algorithms (SEA'11), Lecture notes in computer science, vol. 6630, pp. 421–433 (2011)
20. Solis-Oba, R.: 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In: Proceedings of the 6th Annual European Symposium on Slgorithms (ESA'98), Lecture Notes in Computer Science, vol. 1461, pp. 441–452 (1998)