SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2021, Vol. 64 152103:1–152103:8
https://doi.org/10.1007/s11432-019-2630-2

Incremental algorithms for the maximum internal spanning tree problem

Xianbin ZHU[1], Wenjun LI[3], Yongjie YANG[1,2] & Jianxin WANG[1*]

[1]*School of Computer Science and Engineering, Central South University, Changsha* 410083, *China;*
[2]*Chair of Economic Theory, Saarland University, Saarbrücken* 66123, *Germany;*
[3]*Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha* 410114, *China*

Received 20 January 2019/Revised 4 June 2019/Accepted 28 July 2019/Published online 6 April 2021

Abstract The maximum internal spanning tree (MIST) problem is utilized to determine a spanning tree in a graph G, with the maximum number of possible internal vertices. The incremental maximum internal spanning tree (IMIST) problem is the incremental version of MIST whose feasible solutions are edge-sequences $e_1, e_2, \ldots, e_{n-1}$ such that the first k edges form trees for all $k \in [n-1]$. A solution's quality is measured using $\max_{k\in[n-1]} \frac{\mathrm{opt}(G,k)}{|\mathrm{In}(T_k)|}$ with lower being better. Here, $\mathrm{opt}(G,k)$ denotes the number of internal vertices in a tree with k edges in G, which has the largest possible number of internal vertices, and $|\mathrm{In}(T_k)|$ is the number of internal vertices in the tree comprising the solution's first k edges. We first obtained an IMIST algorithm with a competitive ratio of 2, followed by a 12/7-competitive algorithm based on an approximation algorithm for MIST.

Keywords maximum internal spanning tree, incremental problem, approximation algorithm, competitive ratio

Citation Zhu X B, Li W J, Yang Y J, et al. Incremental algorithms for the maximum internal spanning tree problem. Sci China Inf Sci, 2021, 64(5): 152103, https://doi.org/10.1007/s11432-019-2630-2

1 Introduction

The aim of the maximum internal spanning tree problem (MIST) is to determine a spanning tree with the largest possible number of internal vertices in a connected undirected graph G. It is trivial to observe that MIST is NP-hard because it is essentially a variant of the classic Hamiltonian path problem, which is NP-hard [1]. Because of its applications to the design of communication networks and water supply systems [2], MIST has been explored in the fields of parameterized computation and approximation algorithms. Parameterized computation, a clever method of dealing with NP-hard problems [3–5] has been extensively researched and applied in multiple fields [6–11]. The parameterized version of MIST is the k-internal spanning tree (k-IST) problem. Prieto et al. [12] proposed an $O(k^3)$ kernel for MIST based on the relationship between the k-IST and k-vertex cover problems. This result was subsequently improved to $O(k^2)$ in [13] by applying a parameterized computation technique known as crown decompositions. Fomin et al. [14] achieved a further breakthrough, thus reducing the kernel size for MIST to $3k$. Based on a deeper local search, Li et al. [15] then yielded a kernel of size $2k$.

Approaching the problem from an approximation algorithms perspective, Prieto et al. [12] proposed an MIST algorithm with an approximation ratio of 2. After multiple improvements [15–17], the current best approximation ratio is $\frac{17}{13}$, achieved by an algorithm proposed by Chen et al. [2]. This algorithm has an execution time of $O(|V|^2|E|^2)$, where $|E|$ is the number of edges and $|V|$ is the number of vertices in the graph. However, quicker algorithms with slightly worse approximation ratio have been founded, e.g., Salamon et al. [18] derived a 2-approximate algorithm that runs in linear time, while Knauer et al. [17] developed a $\frac{5}{3}$-approximate algorithm that runs in $O(n^2)$ time, where n is the number of vertices. Finally, we should report that MIST is APX-hard [16].

* Corresponding author (email: jxwang@mail.csu.edu.cn)

In this study, we examine the incremental version of MIST, which is known as the incremental maximum internal spanning tree (IMIST) problem. In general, the incremental variants of combinatorial problems focus on how to expand a small local solution to a complete solution to the whole problem instance via a sequence of intermediate local solutions [19]. Importantly, the sequence's quality is measured in terms of the worst ratio between intermediate and optimal solutions of similar sizes. Incremental problems are relevant in multiple real-world applications such as when budget limitations only allow a small expansion of the current local solution at each step. After the initial work by Mettu et al. [20], the incremental versions of combinatorial problems have been investigated by multiple researchers [20–24]. In the case of IMIST, the aim is to obtain a tree sequence $T_1, T_2, \ldots, T_{n-1}$ in a given graph G, where tree T_k has k edges and is obtained from tree T_{k-1} by adding a single edge. This is equivalent to determining an edge-sequence $e_1, e_2, \ldots, e_{n-1}$ such that T_k is formed from the first k edges in the sequence. Moreover, IMIST considers the quality of all trees T_k, and not only the last one T_{n-1}. An optimal maximum internal k-edge spanning tree (MI-k-EST) of a graph G is a tree in G with k edges which has the largest possible number of internal vertices. In particular, if we define the number of internal vertices in an optimal MI-k-EST by $\mathrm{opt}(G,k)$, the competitive ratio of the sequence $T_1, \ldots, T_{n-1}$ is $\max_{k\in[n-1]} \frac{\mathrm{opt}(G,k)}{|\mathrm{In}(T_k)|}$, where $|\mathrm{In}(T_k)|$ is the number of internal vertices in the tree T_k. Here, we want this ratio to be as small as possible.

To our knowledge, there has been no previous published work on IMIST. Our primary contributions are a simple 2-competitive algorithm and a refined $\frac{12}{7}$-competitive algorithm for IMIST. The key to the second algorithm is utilizing a local search strategy to adjust the structure of the tree obtained using an approximation algorithm for MIST.

2 Preliminaries

Here, we only considered unweighted, undirected graphs without loops or parallel edges. We denote a graph by $G = (V, E)$, where E and V denote the edge set and vertex set, respectively. $E(G)$ and $V(G)$ denote the edge set and vertex set of G, respectively. Furthermore, we use $N_G(u)$ to denote the set of all vertices adjacent to a given vertex u in G, i.e., $N_G(u) = \{v \in V(G) : \{u, v\} \in E(G)\}$. If two vertices u and v are connected via an edge, we call them adjacent. The degree of a vertex u in G is the number of vertices in $N_G(u)$. For a vertex subset $U \subseteq V$, let $G[U]$ denote the subgraph induced by U, i.e., the graph with edge set $\{\{u, v\} \in E : u, v \in U\}$, and vertex set U. Next, $G \setminus U$ denotes the modified graph created by removing all the vertices in U from G, i.e., $G \setminus U = G[V \setminus U]$. Similarly, for a set of edges $F \subseteq E$, $G[F]$ denotes the subgraph induced by F, i.e., the graph with edge set F whose vertex set contains all the endpoints of edges in F. An independent set (IS) in a graph G is a set of vertices whose induced subgraph includes no edges. Finally, a star is a complete bipartite graph whose vertices can be divided into a set containing only one vertex and an IS.

Conventionally, we denote the set $\{i, i+1, \ldots, j\}$, where i and j are two positive integers such that $i \leqslant j$, by $[i, j]$. If $i = j$, $[i, j] = \{i\}$. For simplicity, we write $[1, i]$ as $[i]$.

We utilized the vertex sequence $(v_1, v_2, \ldots, v_t)$ to denote a path P in G, where $v_i \in V(G)$ for all $i \in [t]$ and $v_i \neq v_{i+1}$, and $\{v_i, v_{i+1}\} \in E(G)$ for all $i \in [t-1]$. The path is simple if all the vertices in the path are different from each other, and all paths we consider in this study are simple. We call the first and last vertices of P (i.e., v_1 and v_t) the starting and ending vertices, respectively, and we call these the endpoints of P. The length of P, denoted by $\mathrm{len}(P)$, is equal to the number of vertices in P minus one, i.e., $t-1$.

We denote the set of edges in the path by $E(P)$, i.e., $E(P) = \{\{v_j, v_{j+1}\} : j \in [t-1]\}$. A path $(v_1, \ldots, v_t)$ is maximal if $(N_G(v_1) \cup N_G(v_t)) \subseteq \{v_1, \ldots, v_t\}$. For an edge $\{u, v\}$ and a path $P = (v_1, v_2, \ldots, v_t)$ such that $u \in \{v_1, v_t\}$ and $v \notin \{v_1, \ldots, v_t\}$, we define (v, P) to be the path $(v, v_1, \ldots, v_t)$ if $v_1 = u$; otherwise, $(v, v_t, v_{t-1}, \ldots, v_1)$. We determined that an undirected graph G is connected if any two vertices in $V(G)$ can be linked via a path. A cycle is a vertex sequence $(v_1, v_2, \ldots, v_t)$ such that $(v_1, v_2, \ldots, v_t)$ is a path and v_1 and v_t are connected via an edge. A tree is a connected graph without cycles. An internal vertex in a tree is a tree vertex whose degree is at least two. Tree vertices that are not internal are called leaves of the tree. For a tree T, $\mathrm{In}(T)$ and $\mathrm{Le}(T)$ denote the sets of internal vertices and leaves, respectively.

A k-edge spanning tree of G is a subgraph of G that comprises exactly k edges. For a graph G with n edges, we call an $(n-1)$-edge spanning tree simply a spanning tree. For all $k \in [n-1]$, $\mathrm{opt}(G, k)$ denotes the largest number of internal vertices present in any of the k-edge spanning trees of G, i.e.,

$$\mathrm{opt}(G, k) = \max\{|\mathrm{In}(T)| : T \text{ is a } k\text{-edge spanning tree in } G\}.$$

Next, we formally define the problem considered in this paper.

Problem 1 (IMIST)**.**

Instance. A connected graph $G = (V, E)$.

Goal. A sequence $e_1, e_2, \ldots, e_{n-1}$ of $n-1$ edges in G, where $n = |V|$ such that:

(1) For all $k \in [n-1]$, the first k edges in the sequence induce a tree denoted by T_k, i.e., $T_k = G[\{e_1, \ldots, e_k\}]$;

(2) Among all the sequences of $n-1$ edges that satisfy the first condition, it minimizes the competitive ratio, defined as follows:

$$\max_{k\in[n-1\}} \left\{ \frac{\mathrm{opt}(G,k)}{|\mathrm{In}(T_k)|} \right\}.$$

For an IMIST instance, we call an edge sequence that satisfies the first of the abovementioned conditions a feasible solution for this instance. Hence, using IMIST, we aim to determine a feasible solution with the lowest possible competitive ratio. For a given real number r, an r-competitive algorithm for IMIST is an algorithm that outputs a feasible solution with a competitive ratio of at most r for all connected graphs G.

3 Simple 2-competitive algorithm

In this section, we present a simple 2-competitive algorithm, which iteratively computes k-edge spanning trees for all $k \in [n-1]$, starting from a tree comprising an arbitrary edge, for IMIST. It then expands this edge to create a spanning tree by individually adding edges obtained from maximal paths. In particular, let T_1 be a tree comprising an arbitrary edge and assume that we have determined a tree T_k, where $k < n$. Then, if there is a maximal path of length $\ell \geqslant 1$ in $G \setminus V(T_k)$ whose starting vertex v has a neighbor u in T_k, we add the edge $\{u, v\}$ to T_k, individually followed by the other edges on the maximal path, to construct the subsequent trees $T_{k+1}, T_{k+2}, \ldots, T_{k+\ell}$. If no such maximal path exists, the vertices not in T_k form an IS. In that case, we add certain edges between vertices in T_k and vertices that are not in T_k are used to construct a spanning tree in G. Note that the Algorithm 1 presents a formal description of the algorithm.

Algorithm 1 Simple 2-competitive algorithm for IMIST

Require: A connected graph $G = (V, E)$.
Ensure: A feasible solution S for G with competitive ratio 2.
1: **if** G only contains one vertex **then**
2: Return an empty sequence;
3: **end if**
4: Let T be a tree comprising an arbitrary edge in E;
5: Use S to maintain the feasible solution and set the only edge in T to be the first edge in S;
6: **while** $|V(T)| < n$ **do**
7: **if** there exists a maximal path P of length at least one in $G \setminus V(T)$ whose starting vertex v, has a neighbor u in T **then**
8: Let $P' = (u, P)$;
9: Add all edges in P' to T;
10: Append edges in P' one-by-one into S, from the first to the last along the path;
11: **else** {in this case $V \setminus V(T)$ is an IS}
12: **for** all $v \in V \setminus V(T)$ **do**
13: Add an edge $\{u, v\}$ connecting v and some vertex $u \in V(T)$ to T;
14: Append $\{u, v\}$ into S;
15: **end for**
16: **end if**
17: **end while**
18: Return S.

Our primary result concerning this algorithm can be summarized as follows.

Theorem 1. Algorithm 1 is a 2-competitive algorithm for IMIST and can be executed in $O(n^2)$ time.

Before proving the abovementioned theorem, let us examine certain properties of the spanning tree constructed in Algorithm 1. For ease of exposition, let T_k be the tree with k edges constructed using Algorithm 1 for all $k \in [n-1]$, and let a be the vertex in T_1 that is not in the first maximal path P considered at line 7. Then, we have Lemma 1.

Lemma 1. For all $k \in [2, n-1]$, $\mathrm{Le}(T_k) \setminus \{a\}$ is an IS in G.

Proof. Assume there exist two adjacent vertices u and v such that $u, v \in \mathrm{Le}(T_k) \setminus \{a\}$. Without loss of generality, assume that u is in T_i but not T_{i-1} for certain i $(1 < i < k)$, and that v is in T_j but not T_{j-1} for certain j $(i < j \leqslant k)$. This indicates that immediately after creating T_i, u has at least one adjacent vertex, i.e., v, which is not in T_i. Because of lines 7–9, T_{i+1} is obtained from T_i by adding an edge between u and some vertex u' not in T_i. However, because u is adjacent to certain vertex in T_{i-1} (which contains at least one vertex, say a), u cannot be a leaf in T_{i+1}, let alone in T_k, a contradiction.

Based on Lemma 1, we can derive a relationship between the number of internal vertices in T_{n-1} and that in an optimal maximum internal spanning tree. In the following, for all $k \in [n-1]$, let T^k_{opt} be a k-edge spanning tree in G with the largest possible number of internal vertices. Recall that $\mathrm{opt}(G, k)$ denotes the number of internal vertices in an optimal k-edge spanning tree; therefore, $\mathrm{opt}(G, k) = |\mathrm{In}(T^k_{\mathrm{opt}})|$. Moreover, let A be the set of vertices in $\mathrm{Le}(T_{n-1}) \setminus \{a\}$ that are internal vertices in T^{n-1}_{opt}. This leads us to Lemma 2.

Lemma 2. The inequality $|\mathrm{In}(T_{n-1})| \geqslant |A| + 1$ holds.

Proof. By Lemma 1, A is an IS in G. Let w be a vertex in A. Because w is an internal vertex in T^{n-1}_{opt}, it has at least two neighbors in T^{n-1}_{opt}. Let u and v be any two such neighbors. Because $\mathrm{Le}(T_{n-1}) \setminus \{a\}$ is an IS in G and $A \subseteq \mathrm{Le}(T_{n-1}) \setminus \{a\}$, we have that u and v belong to $\mathrm{In}(T_{n-1}) \cup \{a\}$. Now, we developed a bipartite graph based on the vertex partition $(\mathrm{In}(T_{n-1}), A)$, and added edges between $u \in \mathrm{In}(T_{n-1})$ and $v \in A$ when u is adjacent to v in T^{n-1}_{opt}. Because T^k_{opt} is a tree, the bipartite graph is acyclic. Based on the above discussion, all vertices in A have degree at least two in this bipartite graph, from which it follows that $|\mathrm{In}(T_{n-1})| \geqslant |A| + 1$.

Based on Lemma 2, we can now determine a more interesting result: Algorithm 1 is a 2-approximate algorithm for the MIST problem.

Lemma 3. T_{n-1} is a 2-approximate maximum internal spanning tree in G.

Proof. We have

$$\frac{\left|\mathrm{In}(T^{n-1}_{\mathrm{opt}})\right|}{|\mathrm{In}(T_{n-1})|} \leqslant \frac{|\mathrm{In}(T_{n-1})| + |A| + 1}{|\mathrm{In}(T_{n-1})|} = 1 + \frac{|A| + 1}{|\mathrm{In}(T_{n-1})|} \leqslant 2.$$

The integer 1 on the right-hand side of the first inequality is caused by the vertex a. More precisely, $\mathrm{In}(T^{n-1}_{\mathrm{opt}})$ includes A, some of the vertices in $\mathrm{In}(T_{n-1})$, and possibly the vertex a. The last inequality is because of Lemma 2.

Now, we are ready to obtain the following proof of Theorem 1.

Proof of Theorem 1. Let $P'_1, P'_2, \ldots, P'_t$ be the sequence of paths considered at line 8 in Algorithm 1, and let j be the combined length of the paths. Hence, after the last execution of lines 8 and 9, we obtain the tree T_{j+1}. For all $k \in [2, n-1]$, let $\alpha(k) \in [t]$ be the integer such that the edge present in T_k but absent from T_{k-1} is from path $P_{\alpha(k)}$. Because all the paths in the above sequence have length at least two, we have

$$\alpha(k) \leqslant k/2. \tag{1}$$

Now, we bound the competitive ratios for the T_k. Indeed, the T_1 constructed by the algorithm is optimal. For all $k \in [2, j+1]$, we have

$$\frac{\left|\mathrm{In}(T^k_{\mathrm{opt}})\right|}{|\mathrm{In}(T_k)|} \leqslant \frac{k-1}{|\mathrm{In}(T_k)|} = \frac{k-1}{k-\alpha(k)} \leqslant 2.$$

Here, the last inequality is caused by inequality (1).

Next, we analyzed the competitive ratios of the T_k for $(j+1 < k \leqslant n-1)$. Note that these are obtained from the **else** branch. In particular, T_k is obtained from T_{k-1} by adding a leaf vertex adjacent to an internal vertex in T_{k-1}. Thus, we have

$$\frac{\left|\mathrm{In}(T^k_{\mathrm{opt}})\right|}{|\mathrm{In}(T_k)|} = \frac{\left|\mathrm{In}(T^k_{\mathrm{opt}})\right|}{|\mathrm{In}(T_{n-1})|} \leqslant \frac{\left|\mathrm{In}(T^{n-1}_{\mathrm{opt}})\right|}{|\mathrm{In}(T_{n-1})|} \leqslant 2.$$

Here, the last inequality is because of Lemma 3.

If we use an appropriate data structure to store the graphs involved and maintain a list of the degree-1 and isolated vertices in $G \setminus V(T)$ with neighbors in $V(T)$, this algorithm can be executed in $O(n^2)$ time.

4 Refined competitive algorithm

In this section, we design an improved algorithm with a competitive ratio of at most $\frac{12}{7}$. This involves two primary steps. First, we determine an approximate maximum internal spanning tree in G with certain useful properties. This tree is the final spanning tree in the desired feasible solution. Currently, the best approximation algorithm for MIST in terms of the approximation ratio is reported by Chen et al [2], with an approximation factor of $\frac{17}{13}$ and an execution time of $O(n^2m^2)$. However, a $\frac{5}{3}$-approximate algorithm with a substantially lower execution time ($O(n^3)$) has been reported by Knauer et al. [17]. Because both algorithms offer the same competitive ratio ($\frac{12}{7}$) but the latter is more efficient, our algorithm uses the $\frac{5}{3}$-approximate algorithm. After using this algorithm to calculate an approximate tree, the second step is aimed to order the tree's edges such that a feasible solution can be constructed. In the following, we present the exact algorithm and analyze its competitive ratio.

Let T be an r-approximate maximum internal spanning tree. First, we present two reduction rules that transform T into an r'-approximate maximum internal spanning tree whose leaves induce a subgraph with at most one edge and where r' is not larger than r.

Rule 1. If there is an edge $\{u, v\} \in E(G)$ between two leaves $v, u \in \mathrm{Le}(T)$ such that, on the unique path between u and v in T, there is a vertex w of degree at least 3 in T. We first removed an arbitrary edge that is incident to w and on the unique path, then added the edge between u and v to T.

Indeed, each application of Rule 1 increases the number of internal vertices by at least one, indicating that it can be used at most $n-3$ times. An important consequence of exhaustively applying this reduction rule is that the leaves of the resulting tree are almost an IS. This leads us to Lemma 4.

Lemma 4. Let T' be the tree obtained from T by exhaustively applying Rule 1. Then, T' is an r'-approximate maximum internal spanning tree where $r' \leqslant r$ and the subgraph induced by the leaves of T' contains at most one edge.

Proof. The lemma's first claim is clearly straightforward because each application of Rule 1 increases the number of internal vertices by at least one. This just leaves the second claim, which we show via proof by contradiction. Assume that there are two edges $\{u, v\} \in E(G)$ and $\{x, y\} \in E(G)$ such that $u, v, x, y \in \mathrm{Le}(T')$ are leaves in T' and $\{u, v\} \neq \{x, y\}$. If we add these two edges to T', it creates two cycles, C_1 and C_2. Observe that $C_1 \cup C_2$ contains a vertex of degree at least three in T'. In fact, if all the vertices in C_1 and C_2 are of degree two, the two cycles must be disjoint, contradicting the fact that T' is a tree. However, if C_1 or C_2 includes a vertex with degree at least three, we can apply Rule 1, which is a contradiction.

Now, we will introduce a second reduction rule. A leaf-path of a tree is a path where one of the endpoints is a leaf and all the vertices on the path have degree 2 in the tree. Here, we assume that Rule 1 cannot applied to T.

Rule 2. If $P_1 = (u, v)$ and $P_2 = (x, y)$ are two maximal leaf-paths in T, where u and x are leaves in T and $\{u, y\} \in E(G)$, we first remove the edge incident to y that is on the unique path between u and y in T, and then add the edge $\{u, y\}$ to T.

Each application of Rule 2 increases the number of internal vertices by one; hence it can be used at most $n - 3$ times. Algorithm 2 provides a formal description of our second IMIST algorithm, and our main result is presented in Theorem 2.

Algorithm 2 Refined algorithm for the IMIST problem

Require: A connected graph $G = (V, E)$.
Ensure: A feasible solution S of G.
1: Determine a $\frac{5}{3}$-approximate maximum internal spanning tree T of G;
2: Set S to be an empty sequence;
3: Exhaustively apply Rules 1 and 2 to T and only apply Rule 2 when Rule 1 is not applicable;
4: Determine a longest path P in T;
5: Individually add the edges in P to S such that the edges in S form a tree after each addition;
6: Let $T' := P$;
7: **while** $|V(T')| < n$ **do**
8: Determined a longest path P in $T \setminus V(T')$;
9: Let u be the endpoint of P that has a neighbor v in T';
10: Let $P' = (v, P)$;
11: Add the edges in P' to T';
12: Individually append the edges in P' to S, from first to last;
13: **end while**
14: Return S.

Theorem 2. Algorithm 2 is a $\frac{12}{7}$-competitive algorithm for the IMIST problem and can be executed in $O(n^3)$ time.

Proof. In Algorithm 2, let P_1 be the longest path considered at line 4; moreover, $P_2, P_3, \ldots, P_t$ be the sequence of paths defined at line 10 in the order of definition. Here, we assume that P_1 has length at least three because otherwise the given graph is a star, indicating that an optimal solution can be determined in polynomial time. Let μ be the integer such that all paths $P_1, \ldots, P_\mu$ have lengths of at least three and all the other paths have lengths of at most two. Moreover, for all $k \in [n-1]$, T_k denotes the tree formed by the first k edges in the returned solution, and we define the integer $\alpha(k)$ such that T_k contains certain (or all) edges on $P_{\alpha(k)}$ but none of the edges on $P_{\alpha(k)+1}$. In particular, T_k contains all the edges on $P_1, P_2, \ldots, P_{\alpha(k)-1}$; some (or all) of the edges on $P_{\alpha(k)}$; and none of the edges on any other path. For ease of exposition, for all $i \in [t]$, let $\mathrm{len}(P_{\leqslant i})$ denote the combined length of the paths $P_1, P_2, \ldots, P_i$, i.e.,

$$\mathrm{len}(P_{\leqslant i}) = \sum_{j=1}^{i} \mathrm{len}(P_j).$$

In the following, we demonstrate that, for all $k \in [n-1]$, the number of internal vertices in T_k is at least 7/12 times that in an optimal k-edge spanning tree.

First, for $k \in [\mathrm{len}(P_1)]$, the competitive ratio of T_k is one because all T_k are paths.

Second, we consider the case where $k \in [\mathrm{len}(P_1)+1, \mathrm{len}(P_{\leqslant\mu})]$. Because all the paths $P_1, \ldots, P_{\alpha(k)}$ are of length at least three, we can immediately obtain $\alpha(k) \leqslant k/3$, from which it follows that

$$\frac{\left|\mathrm{In}(T^k_{\mathrm{opt}})\right|}{|\mathrm{In}(T_k)|} \leqslant \frac{k-1}{k-\alpha(k)} \leqslant \frac{k-1}{k-k/3} < 1.5.$$

Third, we consider the case where $k > \mathrm{len}(P_{\leqslant\mu})$. Let $\tau > \mu$ be the integer such that $P_1, P_2, \ldots, P_\tau$ are all of length at least two, whereas $P_{\tau+1}, \ldots, P_t$ are all of length one. Observe that, after the algorithm defines path P_τ, it only adds leaf vertices to the tree. Thus, when $k > \mathrm{len}(P_{\leqslant\tau})$, we have $\mathrm{In}(T_k) = \mathrm{In}(T_{n-1})$. Moreover, for all $k \in [n-1]$, we have $\left|\mathrm{In}(T^k_{\mathrm{opt}})\right| \leqslant \left|\mathrm{In}(T^{n-1}_{\mathrm{opt}})\right|$; consequently, in this case, the competitive ratio of T_k is bounded by

$$\frac{\left|\mathrm{In}(T^{n-1}_{\mathrm{opt}})\right|}{|\mathrm{In}(T_{n-1})|} \leqslant \frac{5}{3}.$$

T_{n-1} is obtained from a $\frac{5}{3}$-approximate maximum internal spanning tree by exhaustively applying Rules 1 and 2. Using Lemma 4, T_{n-1} is a $\frac{5}{3}$-approximate maximum internal spanning tree, so the above inequality holds.

Thus, only the case where $\mathrm{len}(P_{\leqslant\mu}) < k \leqslant \mathrm{len}(P_\tau)$ needs to be considered. In general, Algorithm 2 can be divided into three phases. The first phase adds paths of length at least three (corresponding to $P_1, \ldots, P_\mu$), while the second phase adds paths of length two (corresponding to $P_{\mu+1}, \ldots, P_\tau$). Then, the final phase adds paths of length one (corresponding to $P_{\tau+1}, \ldots, P_t$) where each path has a leaf of T_{n-1} as an endpoint. Let Z and X denote the sets of internal vertices and leaves, respectively, for the first phase, i.e., the internal vertices and leaves in the tree $T_{\mathrm{len}(P_{\leqslant\mu})}$. Similarly, let $B = \{P_{\mu+1}, \ldots, P_\tau\}$ be the set of paths in the second phase. Below, the leaf-vertex of a path P in B is the vertex on P that is a leaf in T_{n-1}. Let D represent the leaves added during the final phase, and let $\zeta = |Z|$, $\chi = |X|$, $\beta = |B|$, and $\delta = |D|$.

Let T^k_{opt} be an optimal maximum internal spanning tree in G and let Z' and X' be the sets of its internal vertices that are in Z and X, respectively. Moreover, let B' be the set of paths in B whose middle (internal) and leaf vertices are internal vertices of T^k_{opt}. Then, denote the remainder of the internal vertices in T^k_{opt} by D'. Note that $D' \subseteq B \cup D$. Let $\zeta' = |Z'|$, $\chi' = |X'|$, $\beta' = |B'|$, and $\delta' = |D'|$. Moreover, let β^* be the number of middle vertices in the paths $P_{\mu+1}, \ldots, P_{\alpha(k)}$ that are internal vertices in T_k. Thus, T_k has $\zeta + \beta^*$ internal vertices. Clearly, we have that $\zeta' \leqslant \zeta$, $\chi' \leqslant \chi$, $\beta' \leqslant \beta$, and $\beta^* \leqslant \beta$. Similar to the proof of Lemma 2, we have the following claim.

Claim 1. The inequality $\zeta \geqslant \chi' + \beta' + \delta' + 1$ holds.

Proof of Claim 1. To confirm this claim, we construct an auxiliary bipartite graph G', based on the vertex partition $(Z, X' \cup D' \cup R)$, where Z, X', and D' are as defined above and R comprises one vertex c_P for each path $P \in B'$. Then, we create edges between vertices in Z and vertices in $X' \cup D'$ if and only if they are adjacent in T^k_{opt}. Moreover, we create edges between vertices in $v \in Z$ and vertices $c_P \in R$ if

the middle or leaf vertex of P is adjacent to v in T^k_{opt}. First, observe that the bipartite graph must be acyclic. Now, we demonstrate that all vertices in $X' \cup D' \cup R$ have at least two neighbors in Z in the bipartite graph G'. As all vertices $v \in X' \cup D'$ are internal vertices in T^k_{opt}, v has at least two neighbors, u and w, in T^k_{opt}. Both u and w must be in Z, since we could otherwise apply Rule 1 or Rule 2, thus contradicting the fact that the tree has been exhaustively reduced by them. For all vertices c_P in R, let v and v' be the middle and leaf vertices, respectively, of a path P in B'. As these are both internal vertices in T^k_{opt}, they have a total of at least three neighbors in T^k_{opt}. Furthermore, these neighbors must be in Z, since we could apply otherwise Rule 1 or Rule 2. In summary, all vertices in $X' \cup D' \cup R$ have at least two neighbors in G' and G' is acyclic. This directly indicates that $\zeta \geqslant \chi' + \beta' + \delta' + 1$, completing the proof of the claim.

Furthermore, we have the following claim.

Claim 2. The inequality $\zeta' + \chi' + 2\beta' + \delta' < \frac{3}{2}\zeta + 2\beta^*$ holds.

Proof of Claim 2. First, recall that T_k and T^k_{opt} have $k+1$ vertices and that $\zeta' + \chi' + 2\beta' + \delta'$ is a lower bound on the number of internal vertices in T^k_{opt}. Thus, we have

$$k + 1 \geqslant \zeta' + \chi' + 2\beta' + \delta + 2. \tag{2}$$

The last integer 2 is because all trees with at least two vertices have at least two leaves. However, T_k can have at most $\zeta + \chi + 2\beta^*$ vertices, i.e.,

$$k + 1 \leqslant \zeta + \chi + 2\beta^*. \tag{3}$$

Moreover, because all paths $P_1, \ldots, P_\mu$ have length at least three, we have

$$\chi \leqslant \frac{1}{2}\zeta + 1. \tag{4}$$

Combining (2)–(4) yields

$$\zeta' + \chi' + 2\beta' + \delta' \leqslant \frac{3}{2}\zeta + 2\beta^*. \tag{5}$$

This completes the proof of Claim 2.

Now, we analyze the competitive ratio of T_k, which we denote by r. First, we have

$$r = \frac{\left|\text{In}(T^k_{\text{opt}})\right|}{|\text{In}(T_k)|} \leqslant \frac{\zeta' + \chi' + 2\beta' + \delta'}{\zeta + \beta^*}. \tag{6}$$

By Claim 1 and inequality (6), the competitive ratio is bounded by

$$r \leqslant \frac{2\zeta + \beta'}{\zeta + \beta^*} \leqslant \frac{3\zeta}{\zeta + \beta^*}. \tag{7}$$

Furthermore, by Claim 2 and inequality (6), we have

$$r \leqslant \frac{1.5\zeta + 2\beta^*}{\zeta + \beta^*} = 2 - \frac{\zeta}{2(\zeta + \beta^*)}. \tag{8}$$

We now proceed by handling the following two cases.

- Case: $0 \leqslant \beta^* \leqslant \frac{3}{4}\zeta$. Here, the bound in (8) gives us

$$r \leqslant 2 - \frac{\zeta}{2(\zeta + \frac{3}{4}\zeta)} = \frac{12}{7}.$$

- Case: $\beta^* \geqslant \frac{3}{4}\zeta$. Here, the bound in (7) gives us

$$r \leqslant \frac{3\zeta}{\zeta + \frac{3}{4}\zeta} = \frac{12}{7}.$$

Moreover, the algorithm can be implemented in $O(n^3)$ time. First, we need $O(n^3)$ time to determine a $\frac{5}{3}$-approximate maximum internal spanning tree in G [17]. Second, Rules 1 and 2 can be applied a total of at most $n-3$ times, and each application requires at most $O(n^2)$ time. Finally, the while loop can be implemented in $O(n^3)$ time by maintaining a list of all the unique paths between pairs of vertices in the tree T, ordered from longest to shortest.

5 Conclusion

In this paper, we have presented two local-search-based approximation algorithms for the incremental version of the MIST problem. One is a simple 2-competitive algorithm, while the other is a 12/7-competitive algorithm based on an approximation algorithm for the MIST problem. To our knowledge, few previous studies have used local search to solve incremental problems. Because local search is a useful algorithm design approach, we believe that incremental variants of classical problems, such as incremental maximum leaf nodes spanning tree (MLNST), incremental k-hop spanning tree (KHST), and other relevant problems, could be explored in this manner.

The MLNST problem requires us to determine a spanning tree with the largest possible number of leaf nodes in G. Although this is clearly just the dual problem of MIST, it may be very hard to design algorithms to solve the incremental version of this problem with low ratios. To solve incremental MLNST, we would be required to determine as many leaf nodes as possible at each step. However, devising an algorithm to achieve this by connecting components with many leaf nodes is challenging, because we do not know the optimal order. However, it may be easier to prove a lower bound on incremental MLNST for the same reason. The aim of the KHST problem is to determine a minimum weighted rooted spanning tree with diameter at most k. A previously presented method for dealing with the incremental k-MST problem [21] cannot be applied to incremental KHST because it cannot guarantee that the tree's diameter will be no larger than k. However, it may be straightforward to solve incremental KHST when $k = 2$.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61672536, 61502054, 61702557, 61420106009, 61872048, 61872450, 61828205), Hunan Provincial Science and Technology Program (Grant No. 2018WK4001), Natural Science Foundation of Hunan Province (Grant No. 2017JJ3333), Scientific Research Fund of Hunan Provincial Education Department (Grant No. 17C0047), China Postdoctoral Science Foundation (Grant No. 2017M612584), and Postdoctoral Science Foundation of Central South University.

References

1 Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: WH Freeman & Co, 1979
2 Chen Z Z, Harada Y, Guo F, et al. An approximation algorithm for maximum internal spanning tree. J Comb Optim, 2018, 35: 955–979
3 Cygan M, Fomin F V, Kowalik Ł, et al. Parameterized Algorithms. Berlin: Springer, 2015
4 Downey R G, Fellows M R. Parameterized Complexity. Berlin: Springer, 1999
5 Fomin F V, Golovach P A, Simonov K. Parameterized k-clustering: the distance matters! 2019. ArXiv:1902.08559
6 Li W J, Liu H Y, Wang J X, et al. An improved linear kernel for complementary maximal strip recovery: simpler and smaller. Theory Comput Sci, 2019, 786: 55–66
7 Shi F, Chen J E, Feng Q L, et al. A parameterized algorithm for the maximum agreement forest problem on multiple rooted multifurcating trees. J Comput Syst Sci, 2018, 97: 28–44
8 Guo L K, Shen H, Zhu W X. Efficient approximation algorithms for multi-antennae largest weight data retrieval. IEEE Trans Mobile Comput, 2017, 16: 3320–3333
9 Feng Q L, Hu J X, Huang N, et al. Improved PTAS for the constrained k-means problem. J Comb Optim, 2019, 37: 1091–1110
10 Feng Q L, Zhu S M, Wang J X. An improved kernel for max-bisection above tight lower bound. Theory Comput Sci, 2018. doi: 10.1016/j.tcs.2018.06.027
11 Fomin F V, Lokshtanov D, Saurabh S, et al. Kernelization: Theory of Parameterized Preprocessing. Cambridge: Cambridge University Press, 2019
12 Prieto E, Sloper C. Either/or: using vertex cover structure in designing FPT-algorithms — the case of k-internal spanning tree. In: Proceedings of the 8th International Workshop on Algorithms and Data Structures (WADS), Ottawa, 2003. 474–483
13 Prieto E, Sloper C. Reducing to independent set structure: the case of k-internal spanning tree. Nord J Comput, 2005, 12: 308–318
14 Fomin F V, Gaspers S, Saurabh S, et al. A linear vertex kernel for maximum internal spanning tree. J Comput Syst Sci, 2013, 79: 1–6
15 Li W J, Cao Y X, Chen J E, et al. Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree. Inf Comput, 2017, 252: 187–200
16 Li X F, Zhu D M. Approximating the maximum internal spanning tree problem via a maximum path-cycle cover. In: Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC), Jeonju, 2014. 467–478
17 Knauer M, Spoerhase J. Better approximation algorithms for the maximum internal spanning tree problem. Algorithmica, 2015, 71: 797–811
18 Salamon G, Wiener G. On finding spanning trees with few leaves. Inf Process Lett, 2008, 105: 164–169
19 Sharp A M. Incremental algorithms: solving problems in a changing world. Dissertation for Ph.D. Degree. Ithaca: Cornell University, 2007
20 Mettu R R, Plaxton C G. The online median problem. SIAM J Comput, 2003, 32: 816–832
21 Lin G L, Nagarajan C, Rajaraman R, et al. A general approach for incremental approximation and hierarchical clustering. SIAM J Comput, 2010, 39: 3633–3669
22 Bernstein A, Disser Y, Groß M. General bounds for incremental maximization. In: Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP), Warsaw, 2017
23 Blum A, Chalasani P, Coppersmith D, et al. The minimum latency problem. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC), Montreal, 1994. 163–171
24 Codenotti B, de Marco G, Leoncini M, et al. Approximation algorithms for a hierarchically structured bin packing problem. Inf Process Lett, 2004, 89: 215–221