Journal of Combinatorial Optimization (2022) 44:2085–2103
https://doi.org/10.1007/s10878-020-00579-z

Exact algorithms for finding constrained minimum spanning trees

Pei Yao[1] · **Longkun Guo**[1,2,3]

Published online: 5 May 2020

Abstract
For a given undirected graph with each edge associated with a weight and a length, the constrained minimum spanning tree (CMST) problem aims to compute a minimum weight spanning tree with total length bounded by a given fixed integer $L \in \mathbb{Z}^+$. In the paper, we first present an exact algorithm with a runtime $O(mn^2)$ for CMST when the edge length is restricted to 0 and 1 based on combining the local search method and our developed bicameral edge replacement approach. Then we extend the algorithm to solve a more general case when the edge length is restricted to 0, 1 and 2 via iteratively improving a feasible solution of CMST towards an optimum solution. At last, numerical experiments are carried out to validate the practical performance of the proposed algorithms by comparing with previous algorithms as baselines.

Keywords Constrained minimum spanning tree · Bicameral edge replacement · Local search

1 Introduction

Broadcasting has become a fundamental method for public information dissemination in nowadays networks because of its advantages in high throughput, energy saving,

Dedicated to Professor Minyi Yue on the Occasion of His 100th Birthday.

✉ Longkun Guo
longkun.guo@gmail.com

Pei Yao
Pei.Yao@foxmail.com

[1] College of mathematics and Computer Science, Fuzhou University, Fuzhou 350116, P.R. China

[2] Shandong Key Laboratory of Computer Networks, School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, P.R. China

[3] Shandong Computer Science Center, (National Supercomputer Center in Jinan), Jinan 250353, P.R. China

efficiency, etc. Most data broadcasting applications require to minimize the occupied resources while guarantee customer experience simultaneously, which is typically to minimize the waiting time of the clients between proposing a request and receiving the data. In the context, a link has a length as the delay of data transmission over the link, and a weight as its occupied resource. Then the constrained minimum spanning tree problem (CMST) arises, which is to compute a tree spanning all the nodes in the network, such that the total edge weight of the tree is minimized and the total length is bounded by a given threshold. Formally, we have the following definition:

Definition 1 (*The Constrained Minimum Spanning Tree problem, CMST*) Given an undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_0^+$, a length function $l : E \to \mathbb{Z}_0^+$, and a length bound $L \in \mathbb{Z}_0^+$, the CMST problem aims to find a spanning tree T with its weight sum minimized and length sum bounded by L, i.e. to minimize $\sum_{e \in T} w(e)$ subject to $\sum_{e \in T} l(e) \leq L$.

1.1 Related works

To the best of our knowledge, the CMST problem was first addressed by Aggarwal et al. (1982) decades ago and was shown weakly $\mathcal{NP}$-hard by reducing from the knapsack problem. Later, Marathe and Ravi (1995) gave a $(2,\ 2)$-approximation algorithm for the CMST problem based on Hassin's approximation approach for computing constrained shortest path. Ravi and Goemans (1996) presented a *polynomial-time approximation scheme* (PTAS) based on their proposed approximation algorithm via Lagrangean relaxation, where PTAS is a polynomial-time algorithm which, for any fixed parameter $\epsilon > 0$, produces a solution with its objective bounded by $1 + \epsilon$ times of the optimum. Although achieving the best possible solution quality, PTAS is always considered of some theoretical value rather than practical applications, because it has the drawback with its time complexity to be similar to $O(kn^{1/\epsilon})$, which in the case for CMST is $O(n^{O(\frac{1}{\epsilon})}(m \log^2 n + n \log^3 n))$ in paper (Ravi and Goemans 1996). In contrast, Hong et al. (2004) proposed a bifactor FPTAS with a runtime $O(mn^5\tau(\lfloor (n-1)/\epsilon \rfloor, \lfloor (n-1)/\delta \rfloor))$ based on tree matrix. As the long standing best theoretical result, Hassin and Levin (2004) proposed an *efficient polynomial time approximate scheme* (EPTAS) with a runtime $O\left(O\left(1/\epsilon\right)^{O(1/\epsilon)}\left(n^4\right)\right)$ via 3-matroid intersection. Superior to PTAS and EPTAS, a *fully polynomial-time approximation scheme* (FPTAS) is also an approximation algorithm which produces a solution within $1+\epsilon$ times of the optimum, yet with the time complexity polynomial to $\frac{1}{\epsilon}$, (i.e., polynomial in both the problem size and $\frac{1}{\epsilon}$); it is thus practically useful and considered a seed leading to many effective and efficient algorithms. However, whether CMST admits an FPTAS remains open for more than 20 years.

The famous minimum spanning tree (MST) problem, a special case of CMST when all edges are with length 0 that the length constraint is actually nullified, is one of the most important optimization problems among the others. Many elegant algorithms were developed for the MST problem, including the two well-known algorithms due to Kruskal (1956) and Prim (1957). Rather than the length constraint, other constraints such as bounding the degrees of the nodes or the delay from the root to every other nodes in the tree were also studied. For the for-

mer, Narula and Ho (1980) proposed the degree constrained minimum spanning tree problem (DCMST) in paper, which was later shown $\mathcal{NP}$-hard via a reduction from the Hamiltonian path problem by Boldon et al. (1996). Then, an approximation algorithm was developed by Lau and Singh (2007) with an additive factor of 1, which achieved the best factor possible under the assumption $\mathcal{P} \neq \mathcal{NP}$. Later, Gao and Jia (2016) proposed a genetic algorithm to solve three uncertain edge value models (uncertain expected value DCMST model, uncertain α-DCMST model and uncertain most chance DCMST model) for the DCMST problem. For the delay constrain, Bicalho et al. (2016) studied the *minimum shallow-light spanning tree* (SLST) problem and gave an exact algorithm based on the branch-and-cut method with exponential runtime. It was shown that even when the delay constraint is only 2 and every edge is with delay 1, the SLST problem remains $\mathcal{NP}$-Hard in paper (Dahl 1998). For a even more special case when all edge costs are within $\{1, 2\}$, Alfandari and Paschos (1999) presented a 1.25-approximation algorithm.

1.2 Our results

In this paper, we present two exact algorithms for CMST when its edge length is within $\{0, 1\}$ and $\{0, 1, 2\}$, respectively. The two algorithms are based on combining local search and our proposed bicameral edge replacement method. The main results of our paper can be summarized as follows:

1. Two exact algorithms are presented for special cases of CMST that the length of edges is in a given set of integers, with time complexity $O(mn^2)$ and $O(m^2n^2)$, respectively.
2. Numerical results demonstrate the practical performance gain of our algorithm. Compared to two baselines including the integer linear programming for the special CMST and the traditional algorithm for computing minimum spanning tree without length constraint, we show that our algorithms achieve the best solution quality among all the tested algorithms. In addition, they also have better runtime in practical than any other algorithms, while retaining the same solution quality.

1.3 Organization

The rest of this paper is organized as follows. Section 2 gives the definition of bicameral edge replacement as well as some related propositions; Sect. 3 proposes an algorithm to solve CMST when $l(e) \in \{0, 1\}$ via the proposed bicameral edge replacement; Sect. 4 shows that the algorithm can be extended to optimally solve the case when $l(e) \in \{0, 1, 2\}$ Sect. 5 gives numerical experiments to demonstrate the practical performance and runtime of the proposed algorithms; At last, Sect. 6 concludes the paper.

2 Bicameral edge replacement

For a pair of edges e and $e', e \in T, e' \in G \backslash T$, we say $(e',\ e)$ is a *tree edge replacement (TER)* iff $T \backslash \{e\} \cup \{e'\}$ remains a tree. For notation briefness, we denote $l(e',\ e) = l(e') - l(e)$ and $w(e',\ e) = w(e') - w(e)$. Then obviously, $\frac{l(e',\ e)}{w(e',\ e)}$ is the exchanging rate between length and weight, when using $(e',\ e)$ to improve the length of T.

For the constrained minimum spanning tree problem (CMST), the crucial question remains how to decide which one TER would benefit better. Intuitively, $r(e',\ e) = \frac{l(e',\ e)}{w(e',\ e)}$ is the exchanging rate with respect to length and weight while swapping e and e' for T. Note that when $l(e',\ e) < 0$ and $w(e',\ e) > 0$, we want to minimize $r(e',\ e)$ as it means maximizing the rate of length decrement over weight increment; identically, when $l(e',\ e) > 0$ and $w(e',\ e) < 0$, we want to maximize $r(e',\ e)$. For briefness, we say a TER $(e',\ e)$ is positive if $l(e',\ e) < 0$ and $w(e',\ e) > 0$; and the TER is negative, if $l(e',\ e) > 0$ and $w(e',\ e) < 0$. It remains to choose between the best negative and positive TERs i.e. choose between the positive TER with minimum $r(e',\ e)$ and the negative TER with maximum $r(e',\ e)$. For the task, inspired by the bicameral cycle cancellation in Guo et al. (2015), we introduce bicameral edge replacement formally as below:

Definition 2 (*Bicameral Edge Replacement, BER*) Let T be a spanning tree with $l(T) > L + 1$ and $(e',\ e)$ be a TER wrt T. Let $\triangle L = L - l(T)$ and $\triangle W = W_{OPT} - w(T)$, where W_{OPT} is the weight of an optimal solution to CMST. Then we have three types of bicameral edge replacements as in the following:

1. Let $(e',\ e)$ be a TER. If $l(e',\ e) < 0$ and $w(e',\ e) \leq 0$ or $l(e',\ e) \leq 0$ and $w(e',\ e) < 0$, then it is a type-I bicameral edge replacement (BER-I);
2. Let $\left(e'_1,\ e_1\right)$ be a TER with $l(e'_1,\ e_1) < 0$, $w(e'_1,\ e_1) > 0$ and $\frac{l(e'_1,\ e_1)}{w(e'_1,\ e_1)} \leq \frac{\triangle L}{\triangle W}$, such that

$$r\left(e'_1,\ e_1\right) = \min\left\{ \frac{l(e',\ e)}{w(e',\ e)} \middle| (e',\ e)\ is\ a\ positive\ TER \right\};$$

Similarly, let $\left(e'_2,\ e_2\right)$ be a TER with $l(e'_2,\ e_2) > 0$, $w(e'_2,\ e_2) < 0$ and $\frac{l(e'_1,\ e_1)}{w(e'_1,\ e_1)} \geq \frac{\triangle L}{\triangle W}$, such that

$$r\left(e'_2,\ e_2\right) = \min\left\{ \frac{l(e',\ e)}{w(e',\ e)} \middle| (e',\ e)\ is\ a\ negative\ TER \right\}.$$

Then If $r\left(e'_1,\ e_1\right) \geq r\left(e'_2,\ e_2\right)$, $\left(e'_1,\ e_1\right)$ is a type-II bicameral edge replacement (BER-II); Otherwise, $\left(e'_2,\ e_2\right)$ is a type-III bicameral edge replacement (BER-III).

Let T be the current spanning tree and T^* be an optimal solution to CMST. The key observation inspiring our algorithm is that the edges of $T \backslash T^*$ can pair with the edges of $T^* \backslash T$ to form a set of disjoint edge pairs, each of which is a TER. So first of all, we will show the edges of $T \backslash T^*$ can pair with the edges of $T^* \backslash T$ to form a set of

disjoint TERs. Actually, we have a more general property over the relationship of the two sets of different edges of any two distinct spanning trees, as stated below:

Lemma 1 *Let T, T' be two distinct spanning trees in graph that $E_1 = E(T)\backslash E(T')$ and $E_2 = E(T')\backslash E(T)$. Let $H = (U, V, E)$ be a bipartite graph whose vertex sets are $U = E_1$ and $V = E_2$. Let $E(H) = \{(e', e)|e \in E_1, e' \in E_2, T\backslash\{e\} \cup \{e'\}$ is a tree$\}$. Then there exists a perfect matching in H.*

Proof Firstly and obviously, $|E_1| = |E_2|$ holds since T and T' are both spanning trees. Suppose the lemma is not true, then there exists a maximal matching $\mathcal{P}'$ with $|\mathcal{P}'| < |E_1| = |E_2|$. Assume that $E'_1 \subseteq E_1$, $E'_2 \subseteq E_2$ be the two sets of edges that do not appear in $\mathcal{P}'$. Obviously, for any $p \in E'_1$, $T' \cup \{p\}$ contains a cycle O. Note that $O \cap E'_2 = \varnothing$, since otherwise suppose $q \in O\backslash O \cap E'_2$, then $T' \cup \{p\}\backslash\{q\}$ remains a tree, and hence $\mathcal{P}' \cup \{(p, q)\}$ remains a matching, which contradicts with the maximality of $\mathcal{P}'$. So $O \subseteq T' \cup \{p\}\backslash E'_2$ holds. Since $T \cap T' = T'\backslash E'_2 = T\backslash E'_1$, we have $O\backslash\{p\} \subseteq T \cap T' \subseteq T$. Then since $p \in E'_1 \subseteq T$ also holds, $O \subseteq T$ is true. Thus, T contains a cycle, contradicting with the fact that T is a tree. □

Following the above lemma, we immediately have the following corollary:

Corollary 1 *There exists a perfect matching $\mathcal{P}$ between the edges of $T\backslash T^*$ and $T^*\backslash T$, such that: (1) $|\mathcal{P}| = |T\backslash T^*| = |T^*\backslash T|$; (2) $x \cap y = \emptyset$ for any distinct $x \neq y \in \mathcal{P}$; (3) For any $(e', e) \in \mathcal{P}$, $e \in T\backslash T^*$ and $e' \in T^*\backslash T$ both hold, and $T\backslash\{e\} \cup \{e'\}$ is a tree.*

3 An optimal algorithm for CMST with edges of length 0 and 1

In this section, we present an algorithm with a polynomial runtime for the special case of the constrained minimum spanning tree problem (CMST) problem that the length of the edge is either 0 or 1. The key idea is similar to the local search method: initially compute a minimum weight spanning tree T without considering the length constraint, then repeatedly improve the length sum of T towards an optimal solution of CMST.

3.1 A naive algorithm

A naive idea to improve the length sum of T by directly employing local search can be simply as: repeat swapping an edge inside T with another edge outside T (with comparative smaller length), such that the length of T decreases until the length constraint is satisfied. The edge swap simply processes as: (1) Compute a tree edge replacement (e', e), such that $l(e', e) < 0$ and $\frac{l(e', e)}{w(e', e)}$ attains minimum; (2) Set $T := T\backslash\{e\} \cup \{e'\}$.

The traditional local search method might generate poor solutions because it only swaps edges to decrease the length of T, but never allows increasing the length of T. So different to traditional local search method, our algorithm allows necessary increment over the length of T, and in general acts like a crafty merchant who sells or buys items depending on whichever produces better profit. That is, the algorithm

decides to decrease or increase the length (and the weight of T will increase or decrease accordingly) depending on whichever produces better "profit".

3.2 Our exact algorithm

Different with the naive algorithm, the key idea of our algorithm is to use the definition of bicameral edge replacement (BER) as in Definition 2 based on local search. Firstly, compute a minimum weight spanning tree T; Then, classify all tree edge replacements (TERs) of T, the type-I of bicameral edge replacement (BER-I) is chosen firstly since it can decrease the length (weight) without any increase the weight (length), otherwise find an optimal TER (e', e) which is the maximum $\frac{l(e',e)}{w(e',e)}$ between $\min\left\{\frac{l(e',e)}{w(e',e)}|(e', e)\, is,\ BER-II\right\}$ and $\max\left\{\frac{l(e',e)}{w(e',e)}|(e', e)\, is\ BER-III\right\}$; Finally, set $T := T\backslash\{e\} \cup \{e'\}$, such that via repeatedly choosing process against T, the length of T can be decreased until the length constraint is satisfied. BER-I is preferred since it decreases the length without any weight gain. However, there exists no BER-I for every tree T, since every T is a minimum weight tree under the length bound $l(T)$. The detail algorithm is described as in Algorithm 1.

Algorithm 1 An algorithm for CMST.

Input: An undirected graph $G = (V, E)$, a weight function $w : E \to Z_0^+$, a length function $l : E \to \{0, 1\}$, and a length bound $L \in \mathbb{R}^+$;
Output: An exactly solution to CMST.
1: Calculate a minimum weight spanning tree T, without considering edge length;
/*T can be computed by Prim's algorithm Prim (1957). */
2: **If** $l(T) = \sum_{e\in T} l(e) \le L$ **then** return T;
/* Otherwise $l(T) > L$, decrease $l(T)$ as in the following.*/
3: **While** $l(T) > L$ **do**
4: Set $r = 0,\ R := \emptyset$;
5: **For** each TER (e, e') **do**
6: **If** $l(e', e) < 0,\ w(e', e) > 0$ **then** set $R_1 := R_1 \cup (e', e)$;
7: **If** $l(e', e) > 0,\ w(e', e) < 0$ **then** set $R_2 := R_2 \cup (e', e)$;
/* R_2 is an empty-set for $l(e) \in \{0, 1\}$. */
8: **Endfor**
9: Set $r_1(e', e) = \min\left\{\frac{l(e',e)}{w(e',e)}|(e', e) \in R_1\right\}, r_2(e', e) = \max\left\{\frac{l(e',e)}{w(e',e)}|(e', e) \in R_2\right\}$
10: **If** $r_1(e', e) < r_2(e', e)$ **then** set $R := arg\max r_2(e', e)$; /* That is BER-III.*/
11: **Else** set $R := arg\min r_1(e', e)$, ;/* BER-II is obtained.*/
12: Set $T := T \cup R_1$;
13: **Endwhile**
14: Return T.

The runtime and the correctness of Algorithm 1 can be stated as in the following:

Theorem 1 *Algorithm 1 produces an optimal solution to special CMST with* $l(e) \in \{0, 1\}$, $\forall e \in G$, *or determines the instance of special CMST is infeasible after* $O\left(mn^2\right)$ *iterations.*

Algorithm 1 length decreases 1 in each iteration. That is the algorithm iterates at most $l(T_0)$ times where T_0 is minimum weight spanning tree and $l(T_0) \leq n$, each of which takes $O(mn)$ time to find a bicameral edge replacement. Hence, the running time of Algorithm 1 is $O\left(mn^2\right)$. The proof of the above theorem will be given in the next section, as there are some propositions that need to be given first.

3.3 Proof of Theorem 1

To prove the correctness of both Algorithm 1 as in Theorem 1, we first show that Algorithm 1 terminates in finite steps. Assume that the algorithm terminates in f iterations. Without loss of generality, assume that $f > 1$. Then we first give the following lemma and show that our algorithm will either improve *the average rate* or keep the average rate unchanged but decrease the length of the tree. Therefore, the algorithm will terminate once there exists no further decrement of the average rate and the length of the tree. Before presenting the proof, we will first prove the proposition as follows:

Proposition 1 *Let* $\{a_1, \ldots, a_n\}$ *and* $\{b_1, \ldots, b_n\}$ *be two sets of numbers. Then when* a_i *and* b_i *are with the same sign, there must exist an integer* $i^* \in [1, n]$ *and an integer* $j^* \in [1, n]$ *such that* $\frac{a_{i^*}}{b_{i^*}} \leq \frac{\sum_i a_i}{\sum_i b_i} \leq \frac{a_{j^*}}{b_{j^*}}$.

Proof We shall only show that $\frac{a_{i^*}}{b_{i^*}} \leq \frac{\sum_i a_i}{\sum_i b_i}$ when $a_i, b_i > 0$ as the second inequality and $a_i, b_i < 0$ are similar. When $n = 1$ the proposition is obviously true. When $n = 2$, w.l.o.g. assume that $\frac{a_1}{b_1} \leq \frac{a_2}{b_2}$, then we have $a_2b_1 - a_1b_2 \geq 0$ and hence $\frac{a_1+a_2}{b_1+b_2} - \frac{a_1}{b_1} = \frac{a_2b_1-a_1b_2}{b_1(b_1+b_2)} \geq 0$. That is, $\frac{a_1+a_2}{b_1+b_2} \geq \frac{a_1}{b_1} \geq \min\left\{\frac{a_1}{b_1}, \frac{a_2}{b_2}\right\}$ is true.

By induction hypothesis, the proposition holds when $n = l - 1$, i.e. there exists an integer $i^* \in [1, l-1]$ such that $\frac{a_{i^*}}{b_{i^*}} \leq \frac{\sum_{i=1}^{l-1} a_i}{\sum_i b_i}$.

When $n = l$, by hypothesis, we have

$$\min\left\{\frac{\sum_{i=1}^{l-1} a_i}{\sum_{i=1}^{l-1} b_i}, \frac{a_l}{b_l}\right\} \leq \frac{\sum_{i=1}^{l} a_i}{\sum_{i=1}^{l} b_i}.$$

Again by hypothesis, there must exist $i^* \in [1, l-1]$, such that

$$\frac{a_{i^*}}{b_{i^*}} \leq \frac{\sum_{i=1}^{l-1} a_i}{\sum_{i=1}^{l-1} b_i}.$$

Then combining the two inequalities above, we have

$$\min\left\{\frac{a_{i^*}}{b_{i^*}}, \frac{a_l}{b_l}\right\} \leq \min\left\{\frac{\sum_{i=1}^{l-1} a_i}{\sum_{i=1}^{l-1} b_i}, \frac{a_l}{b_l}\right\} \leq \frac{\sum_{i=1}^{l} a_i}{\sum_{i=1}^{l} b_i}.$$

□

Lemma 2 *Let T_i be obtained spanning tree in ith iteration, and L_i and W_i be the length and weight of the current solution, respectively. Let (e_i', e_i) be chosen TER in ith step. Let $\triangle L_i = L - L_i$, $\triangle W = W_{OPT} - W_i$ and $r_i = \frac{\triangle L_i}{\triangle W_i}$. Then for the ith iteration, $i < f$, at least one of the following two cases holds:*

1. *$r_{i+1} = r_i$, and $\triangle L_{i+1} < \triangle L_i$;*
2. *$r_{i+1} > r_i$.*

Proof Assume that $\left(e_i', e_i\right)$ is the edge replacement in the ith iteration. If $\left(e_i', e_i\right)$ is BER-I, then either $l\left(e_i', e_i\right) < 0$ and $-W_{OPT} \leq w\left(e_i', e_i\right) \leq 0$ or $l\left(e_i', e_i\right) \leq 0$ and $-W_{OPT} \leq w\left(e_i', e_i\right) < 0$ hold. That is, Clause 2 always holds for BER-I. So we need only to prove the lemma holds for BER-II and BER-III, as the two cases in the following:

1. $\left(e_i', e_i\right)$ is BER-II

   According the Definition 2, we can obtain $\frac{l(e_i', e_i)}{w(e_i', e_i)} \leq r_i$. Then since $w\left(e_i', e_i\right) > 0$, we have:

   $$l\left(e_i', e_i\right) \leq r_i \cdot w\left(e_i', e_i\right). \tag{1}$$

   For the ratio r_i, we have the following equation after one edge replacement:

   $$r_{i+1} = \frac{\triangle L_{i+1}}{\triangle W_{i+1}} = \frac{\triangle L_i - l\left(e_i', e_i\right)}{\triangle W_i - w\left(e_i', e_i\right)}. \tag{2}$$

   Then, combining Inequality 1 with Inequality 2 immediately yields the following inequality:

   $$r_{i+1} \geq \frac{\triangle L_i - r_i \cdot w\left(e_i', e_i\right)}{\triangle W_i - w\left(e_i', e_i\right)} = r_i \cdot \frac{\triangle W_i - w\left(e_i', e_i\right)}{\triangle W_i - w\left(e_i', e_i\right)} = r_i. \tag{3}$$

   Apparently, when the first equality in Inequality 3 holds, Clause 1 is true; Otherwise, Clause 2 is true.
2. $\left(e_i', e_i\right)$ is BER-III

   This proof is similar to the first case. Firstly, according to Definition 2, $\frac{l(e_i', e_i)}{w(e_i', e_i)} \geq r_i$ holds and $w\left(e_i', e_i\right) < 0$. Hence,

   $$l\left(e_i', e_i\right) \leq r_i \cdot w\left(e_i', e_i\right). \tag{4}$$

   Then since $r_{i+1} = \frac{\triangle L_{i+1}}{\triangle W_{i+1}} = \frac{\triangle L_i - l(e_i', e_i)}{\triangle W_i - w(e_i', e_i)}$, from Inequality 4 we have the following inequality:

   $$r_{i+1} \geq \frac{\triangle L_i - r_i \cdot w\left(e_i', e_i\right)}{\triangle W_i - w\left(e_i', e_i\right)} = r_i \cdot \frac{\triangle W_i - w\left(e_i', e_i\right)}{\triangle W_i - w\left(e_i', e_i\right)} = r_i. \tag{5}$$

   Similar to the first case, either Clause 1 or Clause 2 holds. □

According to the above lemma, Algorithm 1 will terminate in finite steps. It remains to show the optimality of the output of Algorithm 1. For the task, we first propose the following property:

Theorem 2 *In each iteration, Algorithm 1 obtains a spanning tree T_i, for which there exists no other spanning tree $T \neq T_i$ such that $l(T) = l(T_i)$ and $w(T_i) > w(T)$.*

Proof Assume that the theorem holds before the ith step (exclusive). Then we shall first show that the length of the tree never increases until the ith step (inclusive). Otherwise, suppose $j < i$ is the first iteration that the length increases. Apparently, $j \geq 2$ since T_0 is the tree with minimum weight. Then we have $l(T_j) = l(T_{j-2})$ since each edge is with length either 0 or 1. By the induction hypothesis, T_{j-2} is with the minimum weight under the length $l(T_{j-2})$, so we have $w(T_j) \geq w(T_{j-2})$. This results in $r_j = \frac{\triangle L_j}{\triangle W_j} \leq \frac{\triangle L_{j-2}}{\triangle W_{j-2}} = r_{j-2}$ and $\triangle L_j = \triangle L_{j-2}$, which contradict with Lemma 2.

Let (e_i', e_i) be the TER in the ith iteration. Suppose there exists a spanning tree T' such that $l(T') = l(T_i)$, $w(T') < w(T_i)$. We shall show T_i will not be chosen as a BER for T_{i-1} in the algorithm and hence a contradiction.

Following Lemma 1, assume that there exists a disjoint TER set M such that $T_{i-1} \cup M = T'$ and $|M| \geq 2$. Then by the definition of T', we have $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} < \frac{l(e_i', e_i)}{w(e_i', e_i)}$. Note that in M there may exist both $l(e', e) < 0$ and $l(e', e) > 0$ by the assumption $l(T') = l(T_i)$. So M can be divided into two disjoint sets M_1 and M_2, which are the two sets of TERs respectively with $l(e', e) < 0$ and $l(e', e) > 0$. Then we need only to discuss the following two cases by comparing M_1 and M_2:

1. $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} \geq \frac{\sum_{M_1} l(e', e)}{\sum_{M_1} w(e', e)}$

   By Proposition 1, there exists (e', e) such that $\frac{l(e', e)}{w(e', e)} \leq \frac{\sum_{M_1} l(e', e)}{\sum_{M_1} w(e', e)} < \frac{l(e_i', e_i)}{w(e_i', e_i)}$. This contradicts the minimality selection rule of Algorithm 1, as it should be (e', e) be selected instead of (e_i', e_i).

2. $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} < \frac{\sum_{M_1} l(e', e)}{\sum_{M_1} w(e', e)}$

   We shall first show $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} < \frac{\sum_{M_2} l(e', e)}{\sum_{M_2} w(e', e)}$. Because the length of the tree must decrease, we have

$$\sum_M l(e', e) < 0. \tag{6}$$

   Then by combining the assumption $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} < \frac{\sum_{M_1} l(e', e)}{\sum_{M_1} w(e', e)}$ with Inequality 6, we have $\sum_{M_2} l(e', e) \sum_{M_1} w(e', e) - \sum_{M_2} w(e', e) \sum_{M_1} l(e', e) < 0$, i.e $\frac{\sum_{M_1} l(e', e)}{\sum_{M_1} w(e', e)} < \frac{\sum_{M_2} l(e', e)}{\sum_{M_2} w(e', e)}$. Hence, inequality $\frac{\sum_M l(e', e)}{\sum_M w(e', e)} < \frac{\sum_{M_2} l(e', e)}{\sum_{M_2} w(e', e)}$ can be obtained. However, there exists (e', e) with $\frac{l(e', e)}{w(e', e)} > \frac{l(e_i', e_i)}{w(e_i', e_i)}$ and $l(e', e) > 0$, $w(e', e) < 0$ by Proposition 1. Similar to case 1, this contradicts with the minimality selection rule. □

Combining Lemma 2 and Theorem 2, we immediately have the correctness of Theorem 1. Moreover, from the proof of Theorem 1, we have the following corollary:

Corollary 2 *There exist no BER-III in any iteration of Algorithm 1.*

4 Extension to CMST with $l(e) \in \{0, 1, 2\}$

In this section, we propose an algorithm for the constrained minimum spanning tree problem (CMST) when $l(e) \in \{0, 1, 2\}$ by generalizing Algorithm 1. Note that different with the case of $l(e) \in \{0, 1\}$, there can exist type-III bicameral edge replacements (BER-III), since Corollary 2 no longer holds.

4.1 Exact algorithm for $l(e) \in \{0, 1, 2\}$

Our algorithm employs a similar idea of Algorithm 1 to solve the case $l(e) \in \{0, 1, 2\}$. However, the algorithm encounters a different case in which the length of T is larger than L. In that case, we still have $l(T) \leq L + 1$. Our algorithm will improve the tree by using the best tree edge replacement (TER) between the two best TERs that is respectively in the two sets: (1) Find a TER with minimum $\frac{l(e', e)}{w(e', e)}$ for all $l(e', e) = -1$; (2) Find a pair TERs (e', e) and (e'_1, e_1) with minimum $\frac{l(e', e)+l(e'_1, e_1)}{w(e', e)+w(e'_1, e_1)}$ for all $l(e', e) = -2$ where $l(e'_1, e_1) = 1$. Finally, by applying the best TER to the current tree which violates the length constraint, we obtain a minimum weight spanning tree satisfying the length constraint. The detailed algorithm is stated in Algorithm 2.

Figure 1 compares the results of three methods: our algorithm, the mixed normalized spanning tree and Prim's algorithm. In the example, we set $l(e) \in \{0, 1, 2\}$ and the length constraint $L = 3$. Figure 2 illustrates an execution of Algorithm 2. The length of dashed edge is 0, the length of normal edge is 1 and the dotted edge is 2 in Fig. 1a.

With the given complete graph in Fig. 1a and the length constraint $L = 3$, our algorithm will first produce a minimum weight spanning tree T_0 by Prim's algorithm as in Fig. 2a. In Fig. 2b, the new tree is generated by replacing the dashed edge of Fig. 2a with dotted edge. In Fig. 2c, we show the second iteration of our algorithm by replacing the dashed edge with the dotted edge in the tree of the lower part of the graph. In Fig. 2d, we construct three new trees, where tree (I) is generated via decreases length 1 of the spanning tree T_2 by replacing the dashed edge with the dotted edge, tree (II) is obtained via replacing the dash-dotted edge by the dotted edge with length decreased by 2, and tree (III) is produced by replacing the dash edge in tree (II) with dotted edge and the length is increased by 1. Figure 2e shows the optimal tree produced by our algorithm.

For the time complexity and correctness of Algorithm 2, we have:

Theorem 3 *Algorithm 2 runs in time* $O\left(m^2n^2\right)$ *and produces an optimal solution to CMST when the length is only included in* $\{0, 1, 2\}$ *for all edges, or determines the instance of CMST is infeasible.*

Algorithm 2 An algorithm for CMST when $l(e) \in \{0,\ 1,\ 2\}$

Input: An undirected graph $G = (V,\ E)$, a weight function $w:\ E \to Z_0^+$, a length function $l:\ E \to \{0,\ 1,\ 2\}$, and a length bound $L \in \mathbb{R}^+$;
Output: An exact solution to CMST when $l(e) \in \{0,\ 1,\ 2\}$.
1: Calculate a minimum weight spanning tree T, without considering edge length;
/*T can be computed by Prim's algorithm Prim (1957). */
2: **If** $l(T) = \sum_{e \in T} l(e) \le L$ **then** return T;
/* Otherwise $l(T) > L$, decrease $l(T)$ as in the following.*/
3: **While** $l(T) > L$ **do**
4: Set $r = 0,\ R := \emptyset$;
5: **For** each TER $(e,\ e')$ **do**
6: **If** $l(e',\ e) < 0,\ w(e',\ e) > 0$ **then** set $R_1 := R_1 \cup (e',\ e)$;
7: **If** $l(e',\ e) > 0,\ w(e',\ e) < 0$ **then** set $R_2 := R_2 \cup (e',\ e)$;
8: **Endfor**
9: Set $r_1(e',\ e) = \min\left\{\frac{l(e',e)}{w(e',e)} | (e',\ e) \in R_1\right\}, r_2(e',\ e) = \max\left\{\frac{l(e',e)}{w(e',e)} | (e',\ e) \in R_2\right\}$
10: **If** $r_1(e',\ e) < r_2(e',\ e)$ **then** set $R := arg\max r_2(e',\ e)$; /* That is BER-III.*/
11: **Else** set $R := arg\min r_1(e',\ e)$; /* BER-II is obtained.*/
12: **If** $l(T) = L + 1$ and $l(R) < -1$ **then**
13: Compute-min-tree $(T,\ L)$, break;
14: **Else**
15: Set $T := T \cup R$;
16: **If** $l(T) = L$ **then** break;
17: **Endif**
18: **Endwhile**
19: Return T.

Algorithm 3 Compute-min-tree $(T,\ R)$

Input: A spanning tree T with $l(T) > L$.
Output: A spanning tree T with $l(T) \le L$.
1: Set $\mathcal{S}_1 := \emptyset$ and $\mathcal{S}_2 := \emptyset$, where $\mathcal{S}_1$ is a set of TERs of length -1;
2: **For** each TER $(e',\ e)$ of T **do**
3: **If** $l(e',\ e) = -1$ **then** set $\mathcal{S}_1 := \mathcal{S}_1 \cup (e',\ e)$;
4: **If** $l(e',\ e) = -2$ **then** /*To compute an optimal pair of TERs. */
5: Set $T_i := T \backslash \{e\} \cup \{e'\}$ and $\mathcal{S}_2 := \emptyset$, where $\mathcal{S}_2$ will be the set of pairs of TERs with a total length -1, i.e. two TERs respectively with length -2 and 1;
5: **For** each TER $(e'_1,\ e_1)$ of T_i **do**
6: **If** $l(e'_1,\ e_1) = 1$ **then** set $\mathcal{S}_2 := \mathcal{S}_2 \cup \{(e',\ e),\ (e'_1,\ e_1)\}$;
7: **Endfor**
9: **Endif**
10: **Endfor**
11: Set the best TER of length -1 as $R_1 := arg\min\left\{\frac{l(e',e)}{w(e',e)} \middle| (e',\ e) \in \mathcal{S}_1\right\}$, and the best TER pair of length -2 as $R_2 := arg\min\left\{\frac{l(e',e)+l(e'_1,e_1)}{w(e',e)+w(e'_1,e_1)} \middle| (e',\ e),\ (e'_1,\ e_1) \in \mathcal{S}_2\right\}$;
12: Return $T := T \cup R$, where $R = arg\min\{w(R_1),\ w(R_2)\}$.

4.2 Proof of Theorem 3

The key idea of the proof is to show that each calculated tree T_i attains minimum weight under the length bound $l(T_i)$ excepting the last iteration.

Lemma 3 *Algorithm 2 produces optimal solutions in each iteration before $l(T) > L$.*

Proof This proof is similar to Theorem 2 but with more sophisticated details.

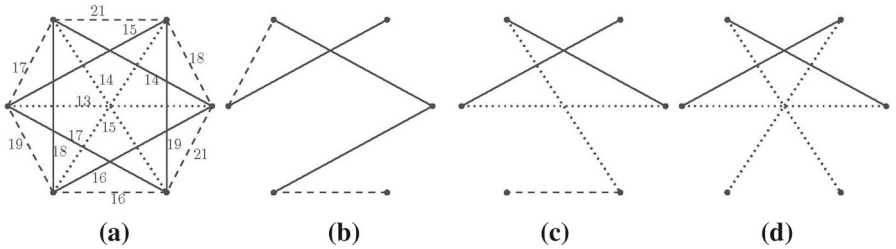


Fig. 1 Comparison of the results output by Algorithm 2, the mixed normalized spanning tree algorithm, and Prim's algorithm. **a** A random complete graph with six vertices and a length constraint $L = 3$, where the length of dashed edges is 0, the length of solid edges is 1 and the length of dotted edges is 2; **b** A spanning tree T_{OPT} produced by our exact algorithm with $l(T_{OPT}) = 3$ and $w(T_{OPT}) = 78$; **c** A spanning tree produced by the mixed normalized spanning tree algorithm with length 6 and weight 72; **d** A tree produced by Prim's algorithm with length and weight respectively being $l(T_0) = 8$ and $w(T_0) = 71$

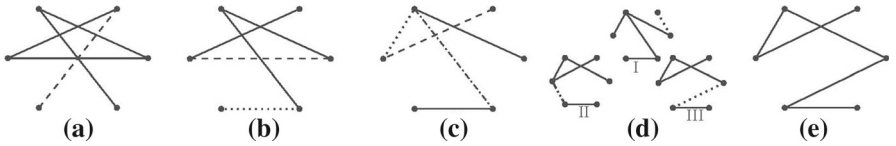


Fig. 2 Example of executing our algorithm on Fig. 1a with the length constraint $L = 3$: **a** the minimum weight spanning tree T_0 with $l(T_0) = 8$ and $w(T_0) = 71$; **b** T_1 that is obtained from T_0 by an iteration of improvement as Algorithm 2, with $l(T_1) = 6$ and $w(T_1) = 72$; **c** T_2 that is produced by improving T_1 by a second iteration with $l(T_2) = 4 = L + 1$ and $w(T_2) = 76$; **d** The set of TERs where tree (I) is the spanning tree generated from the spanning tree T_2 by decreasing length 1 and increasing weight 3, tree (II) is obtained by decreasing length 2 and increasing weight 5, tree (III) is produced by increasing the length of (II) by 1 and decreasing its weight by 3; **e** The optimal spanning tree generated by our algorithm with $l(T_{OPT}) = 3$ and $w(T_{OPT}) = 78$

Firstly, assume that the theorem holds before the ith-step (exclusive). Let (e_i', e_i) be the TER in the ith step and T_i be a spanning tree that is produced by Algorithm 2. Suppose there exists a spanning tree T' such that $l(T') = l(T_i)$, $w(T') < w(T_i)$. We shall show that T_i can not be selected in the algorithm and hence a contradiction.

Following Lemma 1, we can assume that there exists a disjoint TERs set M such that $T_{i-1} \cup M = T'$ and $|M| \geq 2$. Then by the assumption of T', we have $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} < \frac{l(e_i', e_i)}{w(e_i', e_i)}$ when $l(e_i', e_i) < 0$, otherwise $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} > \frac{l(e_i', e_i)}{w(e_i', e_i)}$. Note that in M there may exist both BER-II and BER-III. So we divide M into two disjoint sets M_1 and M_2, which are respectively the set of BER-II and BER-III. When $l(e_i', e_i) < 0$, we can know that there no exist T' that satisfied $l(T') = l(T_i)$, $w(T') < w(T_i)$ by Theorem 2. Hence, we need only to discuss the case $l(e_i', e_i) > 0$. Similar to Theorem 2, we need only to consider the following two cases that compare M_2 and M:

1. $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} \leq \frac{\sum_{M_2} l(e',e)}{\sum_{M_2} w(e',e)}$

   There exists a BER-III (e', e) such that $\frac{l(e_i', e_i)}{w(e_i', e_i)} < \frac{\sum_{M_2} l(e',e)}{\sum_{M_2} w(e',e)} \leq \frac{l(e',e)}{w(e',e)}$ by Proposition 1. This contradicts the maximality selection rule of Algorithm 2 about BER-III, that is (e', e) should be selected instead of (e_i', e_i).

2. $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} > \frac{\sum_{M_2} l(e',e)}{\sum_{M_2} w(e',e)}$

We firstly show $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} > \frac{\sum_{M_1} l(e',e)}{\sum_{M_1} w(e',e)}$ holds. Combining the assumption $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} - \frac{\sum_{M_2} l(e',e)}{\sum_{M_2} w(e',e)} = \frac{\sum_1 l(e',e)\sum_{M_2} w(e',e) - \sum_1 w(e',e)\sum_{M_2} l(e',e)}{\sum_M w(e',e)\sum_{M_1} w(e',e)} > 0$ with $\sum_M l(e',\ e) > 0, \sum_2 l(e',\ e)\sum_{M_1} w(e',\ e) - \sum_2 w(e',\ e)\sum_{M_1} l(e',\ e) < 0$ can be obtained, i.e $\frac{\sum_{M_1} l(e',e)}{\sum_{M_1} w(e',e)} < \frac{\sum_{M_2} l(e',e)}{\sum_{M_2} w(e',e)}$. Hence, the inequality $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} > \frac{\sum_{M_1} l(e',e)}{\sum_{M_1} w(e',e)}$ holds. Then similar to case 1, there exists a BER-II $(e',\ e)$ with $\frac{l(e',e)}{w(e',e)} > \frac{l(e'_i,e_i)}{w(e'_i,e_i)}$ by Proposition 1. According to the selection rule of Algorithm 2 between BER-II and BER-III, it would be $(e',\ e)$ to be selected instead of $(e'_i,\ e_i)$, contradicting with the assumption that $(e'_i,\ e_i)$ is selected.

□

It remains to show the case for the last iteration of Algorithm 2, which immediately results in the optimality of the algorithm as stated below:

Theorem 4 *The spanning tree T^* produced by Algorithm 2 is optimal, i.e. there does not exist any spanning tree that satisfies $l(T) \le L$ and $w(T) < w(T^*)$.*

Proof Assume that the theorem is not true, then there exists a spanning tree T with $l(T) \le L$ and $w(T) < w(T^*)$. Suppose T^* is obtained from T_i by applying one or more TERs. Then $l(T_i) = L + 1$ holds. If $l(T_i) = L + 2$ and $l(T^*) = L$, then let $(e'_i,\ e_i)$ be a TER applied on T_i. Then there must exist a TER set M between T_i and T such that $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} < \frac{l(e'_i,e_i)}{w(e'_i,e_i)}$. Hence, there must exist a TER $(e',\ e)$ such that $\frac{l(e',e)}{w(e',e)} < \frac{l(e'_i,e_i)}{w(e'_i,e_i)}$ if $(e',\ e)$ is BER-II otherwise $\frac{l(e',e)}{w(e',e)} > \frac{l(e'_i,e_i)}{w(e'_i,e_i)}$ by Theorem 3. That contradiction the selection rule of Algorithm 2. And $l(T) < L - 2$ can not exist, since the decrease of length is produced by the increase of weight. Hence, through Algorithm 2, we know that there are two cases for the length of T^* and T. Hence, we need only to discuss the following cases:

1. $l(T) = L$

T^* is obtained by applying one or more TERs on T_i but that is no more than three TERs. Let S be a TERs set between T_i and T^*. Then there exists a disjoint TERs set M between T_i and T by Lemma 1. Through the assumption, we have $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} < \frac{\sum_S l(e',e)}{\sum_S w(e',e)}$. Since $l(T) - l(T_i) = -1$ and M is a disjoint TERs set, then these elements in M must be combined some subset M_i such that the sum length of these subset is -1 where $i \ge 1$, the number of subset satisfies $|M_i| \le 2$. Hence, we have $\frac{\sum_i \sum_{M_i} l(e',e)}{\sum_i \sum_{M_i} w(e',e)} < \frac{\sum_{M_1} l(e',e)}{\sum_{M_1} w(e',e)}$ and M can be divided two parts, let $\mathcal{M}_1$ be a set of these subsets with length -1 and $\mathcal{M}_2$ is a set that all element length is 1. Then through similar to Theorem 2, there exists a subset M_i such that $\frac{\sum_{M_i} l(e',e)}{\sum_{M_i} w(e',e)} < \frac{\sum_S l(e',e)}{\sum_S w(e',e)}$ and $\sum_{M_i} l(e',\ e) = -1$ or $\frac{\sum_{M_i} l(e',e)}{\sum_{M_i} w(e',e)} > \frac{\sum_S l(e',e)}{\sum_S w(e',e)}$ and

$\sum_{M_i} l(e',\ e) = 1$. For the former, it contradicts the selection rule of Algorithm 2. For the later, the element of length in M_i must be $\{-1,\ 2\}$ then there must exist a element of M_i such that its ratio is better than $\frac{\sum_S l(e',e)}{\sum_S w(e',e)}$. That is contradiction the operation of Algorithm 2.

2. $l(T) = L - 1$
   By Lemma 1, there must exist a disjoint TERs set M such that T_i can be changed to T by applying M. Then assume that T_i uses a TERs set S to obtain T^*. Then, $\frac{\sum_M l(e',e)}{\sum_M w(e',e)} < \frac{\sum_S l(e',e)}{\sum_S w(e',e)}$ holds. And M can be divided some disjoint subsets and the length of these subset is no more than 2. Since $\sum_M l(e',\ e) = -2 < 0$, then the proof is similar to Case 1.

□

It remains to show the time complexity of Algorithm 2. The algorithm decreases the length of the tree by 1 or 2 in each iteration when the length of the tree is bounded by L. That is, the algorithm terminates after at most $l(T_0)$ iterations, where $l(T_0) < 2n$ is the weight of a minimum spanning tree. Each of the iterations takes $O(mn)$ time to compute a TER. When $l(T) = L + 1$, the algorithm takes $O(m^2n^2)$ to find a TER to decrease the length of T by 1. Therefore, the running time of Algorithm 1 is $O\left(m^2n^2\right)$.

5 Numerical experiments

In this section, we shall evaluate the practical performance of Algorithm 1 and Algorithm 2 (the exact algorithm, denoted by EA) by comparing its runtime and solution quality with two other baselines: the integer liner programs of the constrained minimum spanning tree (CMST) problem (denoted by ILP), the PRIM algorithm that is a traditional algorithm for computing minimum spanning tree without length constraint (denoted by MST) (Prim 1957). The testing instances for the experiments are composed by a set of randomly generated graphs. In experiments, the number of the edges is denoted by m, and the length constrained is denoted by L. For better comparison, all the algorithms are implemented in Java, on a PC with Intel Core i7 processor and 16GB memory. The ILP for CMST we use in the paper is similar to the ILP for minimum spanning tree as in Korte et al. (2002).

The runtime of EA is evaluated by simulation experiments. In these experiments, we randomly generate a complete graph via randomly producing edges between vertices. The edge weight is integer and uniformly distributed in $[1,\ 1000]$, while the length is chosen from $\{0,\ 1\}$ or $\{0,\ 1,\ 2\}$. These experimental results are reported in Table 1 and Table 2, where the length in Table 1 is restricted in $\{0,\ 1\}$ and Table 2 is in $\{0,\ 1,\ 2\}$.

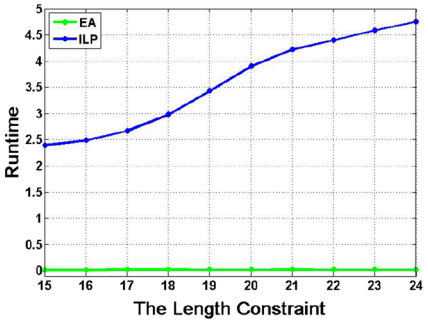
Among the three algorithms in Table 1, EA and ILP both produce solutions with the best quality. This agrees with the theoretical analysis that EA and ILP both produce optimal solutions. In contrast, MST always produces solutions with the minimum weight (even smaller than our EA in some of cases), but the solution might violate the length constraint, where it has an average length sum of $1.2L$. As with compromising solution quality, MST compares favorably in the practical runtime against EA and ILP, while ILP has the worst runtime which is rapidly rising as the number of edge

Table 1 Comparison of our algorithm with three baselines when $l(e) \in \{0, 1\}$

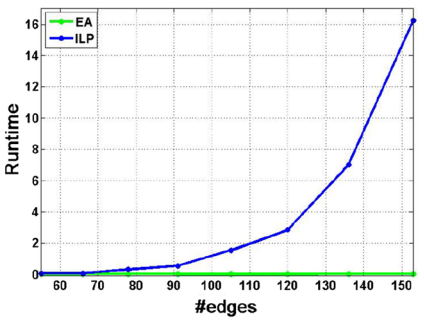
Random	Size		ILP			Our EA			MST		
	m	L	Weight	Length	Time(s)	Weight	Length	Time(s)	Weight	Length	Time(s)
Ran_01	45	11	1938	11	0.038	1938	11	0.007	1844	13	<1ms
Ran_02	55	12	737	12	0.070	737	12	0.015	536	15	0
Ran_03	78	14	1705	14	0.40	1705	14	0.017	1452	17	0
Ran_04	91	16	1496	16	0.85	1496	16	0.019	1243	20	0
Ran_05	136	20	1398	20	12.13	1398	20	0.035	1018	24	0
Ran_06	153	21	1266	21	15.33	1266	21	0.033	1096	25	0
Ran_07	171	22	1109	22	35.50	1227	22	0.036	932	27	0
Ran_08	190	22	1109	22	90.08	1109	22	0.046	1009	26	0
Ran_09	4950	123	–	–	–	1468	123	0.316	1224	147	0
Ran_10	16110	221	–	–	–	1322	221	0.892	1046	267	0
Ran_11	19900	249	–	–	–	1451	249	1.58	1157	299	0
Ran_12	44850	375	–	–	–	1154	375	3.95	894	451	0

Table 2 Comparison of our algorithm with three baselines when $l(e) \in \{0, 1, 2\}$

Random	Size		ILP			Our EA			MST		
	m	L	Weight	Length	Time(s)	Weight	Length	Time(s)	Weight	Length	Time(s)
Ran_01	45	11	1062	11	0.020	1062	11	0.015	897	14	0
Ran_02	55	12	1019	12	0.044	1019	12	0.016	795	15	0
Ran_03	78	15	1714	15	0.25	1714	15	0.037	1337	19	0
Ran_04	105	17	1571	17	1.31	1571	17	0.040	1330	21	0
Ran_05	120	20	1475	20	2.287	1475	20	0.040	1276	25	0
Ran_06	136	20	2506	20	6.88	2506	20	0.060	1109	25	0
Ran_07	171	23	1651	23	47.27	1651	23	0.053	1244	28	0
Ran_08	4950	43	–	–	–	1569	43	0.422	1215	87	0
Ran_09	11175	74	–	–	–	1488	74	0.959	1232	148	0
Ran_10	19900	100	–	–	–	1380	100	1.799	1012	200	0
Ran_11	31125	128	–	–	–	1314	128	2.929	957	256	0
Ran_12	44858	139	–	–	–	1429	139	4.199	1064	278	0

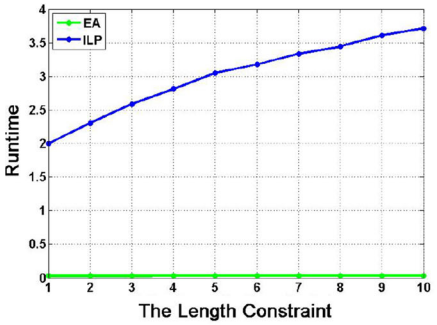


(a) Runtime of EA and ILP wrt length constraint

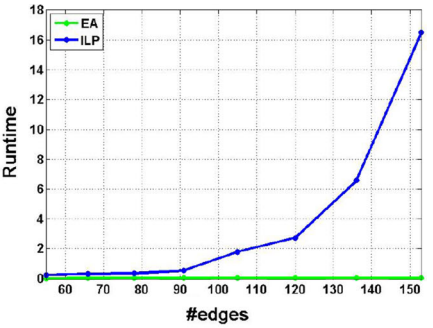


(b) Runtime of EA and ILP wrt #edges

Fig. 3 Runtime comparison for $l(e) \in \{0,\ 1\}$



(a) Runtime of EA and ILP wrt length constraint



(b) Runtime of EA and ILP wrt #edges

Fig. 4 Runtime comparison for $l(e) \in \{0,\ 1,\ 2\}$

increases. When the number of edge is not less than 190 (i.e. a complete graph with 20 vertices), the runtime of ILP rises so high that it could not be used to solve the CMST problem, but our EA can obtain an optimal solution with less than 1 seconds time when the number of edge is no more than 16110 (i.e. a complete graph with 180 vertices). Since there has two variables m and L in Table 1 and we could not find which is more important for the change of runtime. Hence, we use the Matlab software to draw figures to show the influences of the length constraint and the number of edges for the runtime of algorithms. Figure 3a demonstrates the influence of length constraint for $l(e) \in \{0,\ 1\}$ for the number of edges being fixed at 120 (i.e. a complete graph with 16 vertices). It is shown that the runtime of ILP grows when the difference between length constraint and the length of minimum weight spanning tree increases. Figure 3b demonstrates the change of runtime when the number of edges increases from 55 to 153 where the graph is a complete graph and the number of vertices increases from 11 to 18 with one at each time. In this figure, we can see the runtime of ILP grows much faster than EA when the number of edge increases, indicating the large runtime gap between ILP and EA.

The experimental results for $l(e) \in \{0,\ 1,\ 2\}$ are as illustrated in Table 2, where ILP and EA produce identical solutions, while the length of the spanning tree computed

by MST is almost $1.2L$ that is larger than the length constraint. The runtime of our EA is always less than ILP that cannot solve CMST in practice when the number of edges is larger than 4950 (i.e a complete graph with 100 vertices). In contrast, the runtime of our EA is less than one second for the instances. We use Fig. 4 to better evaluate the influence of the length constraint and the number of edges. The number of edges is fixed at 120 (i.e. a complete graph with 16 vertices) in Fig. 4a and the length constraint is fixed at 17 in Fig. 4b. As illustrated in Figs. 3 and 4, the runtime of EA grows much slower for $l(e) \in \{0, 1\}$ than for $l(e) \in \{0, 1, 2\}$ provided that the two algorithms are against the same increasing rate of either length constraint or edge number.

6 Conclusion

In this paper, we have devised two exact algorithms for the constrained minimum spanning tree (CMST) problem when the edge length is within $l(e) \in \{0, 1\}$ and $l(e) \in \{0, 1, 2\}$, respectively. The two algorithms respectively run in time $O(mn^2)$ and $O(m^2n^2)$, where m and n are respectively the number of edges and the number of vertices. The key idea of the algorithm is based on an enhanced local search method combining with our proposed bicameral edge replacement approach. Then we carried numerical experiments to evaluate the performance of our algorithms by comparing with other baselines. Our algorithm has the potential to be extended to solve CMST with $l(e) \in \{0, 1, \ldots, n\}$.

Acknowledgements The research is supported by National Science Foundation of China (No. 61772005), Innovative Team of Youth and Creative Science and Technology Program of Shandong Province (2020KJN008) and Natural Science Foundation of Fujian Province (No. 2017J01753).

References

Aggarwal V, Aneja YP, Nair KPK (1982) Minimal spanning tree subject to a side constraint. Comput Oper Res 9(4):287–296

Alfandari L, Paschos VT (1999) Approximating minimum spanning tree of depth 2. Int Trans Oper Res 6(6):607–622

Bicalho LH, Cunha ASD, Lucena A (2016) Branch-and-cut-and-price algorithms for the degree constrained minimum spanning tree problem. Comput Optim Appl 63(3):1–38

Boldon B, Deo N, Kumar N (1996) Minimum-weight degree-constrained spanning tree problem: Heuristics and implementation on an simd parallel machine. Parallel Comput 22(3):369–382

Dahl G (1998) The 2-hop spanning tree problem. Oper Res Lett 23(1):21–26

Gao X, Jia L (2016) Degree-constrained minimum spanning tree problem with uncertain edge weights. Appl Soft Comput 56:580–588

Guo L, Liao K, Shen H, Li P (2015) Brief announcement: Efficient approximation algorithms for computing k disjoint restricted shortest paths. In: Proceedings of the 27th ACM on symposium on parallelism in algorithms and architectures, SPAA 2015, Portland, OR, USA, June 13–15, 2015, pp 62–64

Hassin R, Levin A (2004) An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. SIAM J Comput 33(2):261–268

Hong SP, Chung SJ, Park BH (2004) A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem. Oper Res Lett 32(3):233–239

Korte B, Vygen J, Korte B, Vygen J (2002) Combinatorial optimization, vol 1. Springer, New York

Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc 7(1):48–50

Lau LC, Singh M (2007) Iterative rounding and relaxation (combinatorial). ACM Symp Theory Comput (STOC) 651:660
Marathe M, Ravi R, Ravi S, Rosenkrantz D, Hunt H (1995) Bicriteria network design problems. In: Automata, languages and programming, pp 487–498
Narula SC, Ho CA (1980) Degree-constrained minimum spanning tree. Comput Oper Res 7(4):239–249
Prim RC (1957) Shortest connection networks and some generalizations. Bell Labs Tech J 36(6):1389–1401
Ravi R, Goemans M (1996) The constrained minimum spanning tree problem. In: Algorithm theory SWAT'96, pp 66–75