The subdivision-constrained minimum spanning tree problem

Jianping Li *, Weidong Li, Tongquan Zhang, Zhongxu Zhang

Department of Mathematics, Yunnan University, Kunming 650091, PR China

Abstract
Motivated by the constrained minimum spanning tree (CST) problem in Hassin and Levin [R. Hassin, A. Levin, An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection, SIAM Journal on Computing 33 (2) (2004) 261–268], we study a new combinatorial optimization problem in this paper, called the general subdivision-constrained minimum spanning tree problem (GSCST): given a graph $G = (V, E; w, c)$ with two nonnegative integers $w(e)$ and $c(e)$ for each edge $e \in E$, two positive integers $B$ and $d$, the GSCST problem is to first find a spanning tree $T = (V, E_T)$ of $G$ with weight $\sum_{e \in E_T} w(e) \leq B$ and then to insert some new vertices on some suitable edges in $T$ such that each edge in the subdivision tree $T'$ of $T$ has its weight not beyond $d$. The objective is to minimize the cost $\sum_{e \in E_T} \text{insert}(e)c(e)$ of such new vertices inserted on the suitable edges among all spanning trees of $G$ subject to the two preceding constraints, where a subdivision tree $T'$ of $T$ is constructed by inserting some new vertices on the suitable edges in $T$, the value $\text{insert}(e) = \lceil \frac{w(e)}{d} \rceil - 1$ is the least number of vertices inserted and $c(e)$ is the cost of each vertex inserted on the edge $e$. We obtain the following main results: (1) the GSCST problem and its variant are still NP-hard, by a reduction from the 0–1 knapsack problem, respectively; (2) the GSCST problem as well as its variant is polynomially equivalent to the CST problem, which implies the existence of a polynomial time approximation scheme to solve the GSCST problem and its variant; (3) we finally design three strongly polynomial time algorithms to solve the special versions of the GSCST problem and its variant, respectively.

1. Introduction

In this paper, we only consider finite simple graphs and use Bondy and Murty [3] for terminology and notation not defined here.

The minimum spanning tree problem (MST) is one of very important combinatorial optimization problems, and it has many applications in some domains. Many algorithms and applications about the MST problem can be found in [3,10,12].

Related to the MST problem, Megiddo [9] studied the minimum ratio spanning tree problem (MRST): given a graph $G = (V, E; c, d)$, two nonnegative integers $c(e)$ and $d(e)$ for each edge $e \in E$, find a spanning tree $T = (V, E_T)$ to minimize $c(T)/d(T)$, where $c(T) = \sum_{e \in E_T} c(e)$ and $d(T) = \sum_{e \in E_T} d(e)$. This model is that of minimizing cost-to-time ratio spanning tree in a graph. By utilizing binary method, Megiddo [9] designed a polynomial time algorithm to solve the MRST problem.

Aggarwal et al. [1] first introduced the constrained minimum spanning tree problem (CST); given a graph $G = (V, E; w, c)$ with $n$ vertices and $m$ edges, two nonnegative integers $w(e)$ and $c(e)$ for each edge $e \in E$, and a bound $W$, find a spanning tree $T = (V, E_T)$ of $G$ such that $\sum_{e \in E_T} w(e) \leq W$ and $\sum_{e \in E_T} c(e)$ is minimized. For simplicity of exposition, we find $G = (V, E; w, c); W)_{\text{CST}}$ to represent an instance of the CST problem.
Although the MRST problem is solved by a polynomial time algorithm [9], the CST problem is NP-hard [1,4], by a reduction from the 0–1 knapsack problem. By utilizing Lagrangian relaxation method and exploiting adjacency relations for matroids, Goemans and Ravi [5] designed a polynomial time approximation scheme (PTAS) for the CST problem, i.e., a \((1 + \varepsilon)\)-approximation algorithm for every \(\varepsilon > 0\), with time complexity \(\Theta(n^{\frac{1}{2}} \log c_{\max})\), where \(c_{\max}\) is the largest cost. Hong et al. [7] recently designed a pseudopolynomial time algorithm for the CST problem, which is based on a two-variable extension of the tree-matrix theorem. By adopting the basic ideas in [5] and adding them to a novel application of a matroid intersection algorithm, Hassin and Levin [6] also designed a PTAS for the CST problem, with time complexity \(\Theta((\frac{1}{2})^{\frac{3}{2}} n^4)\) for every \(\varepsilon > 0\); but the question whether there exists a fully polynomial time approximation scheme (FPTAS) for the CST problem is still open [6].

For the CST problem, there are no relations between the two nonnegative integers \(w(e)\) and \(c(e)\) for each edge \(e\) in \(G\), it follows that the CST problem is NP-hard. In this paper, we shall study the problem involving some relations between the two nonnegative integers \(w(e)\) and \(c(e)\) for each edge \(e\) in \(G\), and we expect that there are some polynomial time algorithms to solve the new combinatorial optimization problems related to the CST problem and others.

Motivated by a PTAS in [6] for the CST problem, we study the new combinatorial optimization problem, called the general subdivision-constrained minimum spanning tree problem (GSCST): given a graph \(G = (V, E; w, c)\) with \(n\) vertices and \(m\) edges, two nonnegative integers \(w(e)\) and \(c(e)\) for each edge \(e\) in \(E\), and two positive integers \(B\) and \(d\), find a spanning tree \(T = (V, E_T)\) of \(G\) to satisfy the two constraints: (i) \(\sum_{e \in E_T} w(e) \leq B\), and (ii) then inserting some new vertices on some suitable edges in \(T = (V, E_T)\) such that each edge in the subdivision tree \(T'\) of \(T\) has its weight not beyond the constant \(d\). The objective is to minimize the cost \(\sum_{e \in E_T} \text{insert}(e) c(e)\) of such new vertices inserted on the suitable edges in \(T\), where minimum is taken over all spanning trees of \(G\) subject to the two preceding constraints and \(\text{insert}(e) = \lceil \frac{w(e)}{d} \rceil - 1\). Here, a subdivision tree \(T'\) of \(T\) is constructed by inserting some new vertices on the suitable edges in \(T\) such that each edge in \(T'\) has its weight at most \(d\), and for each edge \(e\) in \(G\), \(c(e)\) is treated as a unit cost when we insert a new vertex on \(e\). Particularly, if \(e\) is selected as an edge in the spanning tree \(T\), there are at least \(\text{insert}(e)\) vertices inserted on \(e\) such that each edge in \(T'\) has its weight not beyond \(d\). For simplicity of exposition, we use \(G = (V, E; w, c); B; d)_{\text{GSCST}}\) to represent an instance of the GSCST problem.

We also study a variant of the GSCST problem, which maintains the same constraints in the GSCST problem, but the objective becomes to minimize the value \(\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))\), where minimum is taken over all spanning trees of the graph \(G\). We call this new problem as a variant of the general subdivision-constrained minimum spanning tree problem (VGSCST). For simplicity of exposition, we use \(G = (V, E; w, c); B; d)_{\text{VGSCST}}\) to represent an instance of the VGSCST problem.

For each edge \(e\) in the spanning tree \(T\), we show how to insert new \(\text{insert}(e)\) vertices on suitable edge \(e\) (if possible) in the following steps: for each edge \(e = uv\) with weight \(w(e) > d\), insert sequentially the \(\text{insert}(e)\) vertices on \(e\) from the vertex \(u\) to the vertex \(v\) such that each new edge with the consecutive vertices on \(e\) exactly has the weight \(d\), except the final new edge with an end-vertex \(v\) (for convenience, we call such new edges as the subdivision edges); for each edge \(e\) with weight \(w(e) \leq d\), it needs no vertex inserted on \(e\). Thus, each subdivision edge on \(e\) exactly has its weight \(d\), unless the final subdivision edge with an end-vertex \(v\) has its weight at most \(d\), which ensures that each edge in the subdivision tree \(T'\) of \(T\) has weight not beyond \(d\).

Since Hassin and Levin [6] proposed an open question whether or not there exists an FPTAS for the CST problem, this is the reason why we are interested in the GSCST problem and its variant. As far as we know by now, there are no results for the GSCST problem and its variant. In this paper, we directly prove that the GSCST problem and its variant are still NP-hard, by a reduction from the 0–1 knapsack problem [4], respectively. In addition, motivated by an interest in the polynomial equivalence between the Hitchcock problem and the minimum-cost flow problem [10], we present the fact that the GSCST problem and its variant are both polynomially equivalent to the CST problem, which leads that there is a PTAS to solve both new problems, by utilizing an algorithm [6] to solve CST problem. Since the GSCST problem and its variant are NP-hard, there are no polynomial time algorithms to solve them exactly, unless \(P = \mathcal{NP}\), then we study the some special versions of the GSCST problem and its variant, which will be solved optimally by some strongly polynomial time algorithms.

When we are interested in the least number vertices inserted in the spanning tree \(T\) for the GSCST problem, equivalently, we take care of same unit cost \(c(e) = k\) for each edge \(e\) in the spanning tree \(T\) in this case to satisfy the preceding constraints, we call this version of the GSCST problem as the subdivision-constrained minimum spanning tree problem in the first version (the SCST-1 problem), and then we can design a strongly polynomial time to solve the SCST-1 problem. For simplicity of exposition, we use \(G = (V, E; w, k); B; d)_{\text{GSCST}}\) to represent an instance of the SCST-1 problem.

In addition, when we are interested in the same unit cost \(c(e) = k\) for each edge \(e\) in the spanning tree \(T\) for the VGSCST problem to satisfy the preceding constraint, i.e., the constraint \(\sum_{e \in E_T} w(e) \leq B\) still maintains for a spanning tree \(T\), but the objective becomes to minimize the value \(\sum_{e \in E_T} (w(e) + k \cdot \text{insert}(e))\), where minimum is taken over all spanning trees of the graph \(G\), we call this version of the VGSCST problem as the subdivision-constrained minimum spanning tree problem in the second version (the SCST-2 problem), and then we can present a strongly polynomial time to solve the SCST-2 problem. For simplicity of exposition, and we use \(G = (V, E; w, k); B; d)_{\text{VGSCST}}\) to represent an instance of the SCST-2 problem.

On the other hand, when we are interested in different unit costs \(c(e)\) to different edges when we insert some vertices on the suitable edges in a spanning tree \(T = (V, E_T)\) of \(G\), and we ignore the constraint \(\sum_{e \in E_T} w(e) \leq B\), i.e., the constraint \(\sum_{e \in E_T} w(e) \leq B\) is omitted, but the objective becomes to minimize either the value \(\sum_{e \in E_T} \text{insert}(e)c(e)\) or the value \(\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))\) of the vertices inserted in the spanning tree \(T\) to maintain the constraint that each edge
The Fig. 4 solution such that case, but the right oblique edge $e$.

Denote sizes $a_i \in \mathbb{Z}^+$ and integer profits $p_i \in \mathbb{Z}^+$, and a ‘knapsack capacity’ $W$, find a subset $S \subseteq \{a_i \leq W \}$ and $\sum_{i \in S} p_i$ is maximized.

We construct an instance $\tau(I)$ of the GSCST problem: a graph $G$ with $2n+1$ vertices is shown in Fig. 1(a), where each edge $e$ in the graph carries two nonnegative integers corresponding to $w(e)$ and $c(e)$, and each basic block is defined as Fig. 1(b). Denote $a_{\text{max}} = \max(a_i | 1 \leq i \leq n)$, $p_{\text{max}} = \max(p_i | 1 \leq i \leq n)$ and $d = a_{\text{max}} + 1$. For each $1 \leq i \leq n$, the lower horizontal edge $e_i$ has weight $w(e_i) = a_i + d + 1$ and unit cost $c(e_i) = p_{\text{max}} - p_i$, here $w(e_i) = 1$ in this case. And for each $1 \leq i \leq n$, the left oblique edge $e_{n+2i-1}$ has weight $w(e_{n+2i-1}) = 1$ and unit cost $c(e_{n+2i-1}) = p_{\text{max}}$, implying $w(e_{n+2i-1}) = 0$ in this case, but the right oblique edge $e_{n+2i}$ has weight $w(e_{n+2i}) = d + 1$ and unit cost $c(e_{n+2i}) = p_{\text{max}}$, implying $w(e_{n+2i}) = 1$ in this case. Put the bound $B = n(d + 2) + W$ for the instance $\tau(I)$. The objective is to find a spanning tree $T = \{V, E_T\}$ of $G$ such that $\sum_{e \in E_T} w(e) \leq B$ and $\sum_{e \in E_T} w(e)c(e)$ is minimized.

Now, we shall obtain the following result: the instance $I$ of the 0–1 knapsack problem has an optimal solution $OPT_{0-1} = \{i_1, i_2, \ldots, i_k\}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, if and only if the instance $\tau(I)$ of the GSCST problem has an optimal solution $OPT_{\text{GSCST}} = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\} \cup \{e_{n+2j-1} : 1 \leq j \leq n\} \cup \{e_{n+2j} : j \in S - \{i_1, i_2, \ldots, i_k\}\}$; moreover, the two optimal values must satisfy the equality $\sum_{e \in OPT_{\text{GSCST}}} c(e) = np_{\text{max}} - \sum_{i \in OPT_{0-1}} p_i$.

In fact, if the 0–1 knapsack problem has an optimal solution $OPT_{0-1}$ with the constraint $\sum_{i \in OPT_{0-1}} a_i \leq W$ and the optimal value $\sum_{i \in OPT_{0-1}} p_i$, by finding a spanning tree $OPT_{\text{GSCST}}$ in the graph $G$ indicated as before, we obtain the constraint $w(OPT_{\text{GSCST}}) = \sum_{e \in OPT_{\text{GSCST}}} w(e) = \sum_{i \in OPT_{0-1}} (a_i + d + 1) + n + \sum_{j \in S - OPT_{0-1}} (d + 1) = n(d + 2) + \sum_{i \in OPT_{0-1}} a_i \leq n(d + 2) + W = B$, with the cost value $c(OPT_{\text{GSCST}}) = \sum_{e \in OPT_{\text{GSCST}}} c(e) = \sum_{i \in OPT_{0-1}} (p_{\text{max}} - p_i) + 0 + \sum_{j \in S - OPT_{0-1}} p_{\text{max}} = np_{\text{max}} - \sum_{i \in OPT_{0-1}} p_i$.

Hence the optimal solution $OPT_{0-1}$ of the instance $I$ to the 0–1 knapsack problem implies that the spanning tree $OPT_{\text{GSCST}}$ in the graph $G$ is also an optimal solution to the instance $\tau(I)$ of the GSCST problem. And vice versa.

Since the 0–1 knapsack problem is $NP$-hard in Garey and Johnson [4], then the GSCST problem is $NP$-hard. Hence, we reach the conclusion of this theorem.

By utilizing the argument similar to the proof of the $NP$-hardness of the GSCST problem, we can prove that the VGSCT problem is $NP$-hard, by transforming the 0–1 knapsack problem to the GSCST problem, too.

2. $NP$-hardness of the GSCST and its variant

By utilizing a transformation from the 0–1 knapsack problem [4] to the GSCST problem, we can present a proof of the $NP$-hardness for the GSCST problem.

**Theorem 1.** The GSCST problem is $NP$-hard.

**Proof.** The $NP$-hardness of the GSCST problem will be proved, by a reduction from the 0–1 knapsack problem [4] in polynomial operations.

Consider any instance $I$ of the 0–1 knapsack problem [4]: given a set $S = \{1, 2, \ldots, n\}$ of $n$ objects, with specified integer sizes $a_i \in \mathbb{Z}^+$ and integer profits $p_i \in \mathbb{Z}^+$, and a ‘knapsack capacity’ $W$, find a subset $S' \subseteq S$ such that $\sum_{i \in S'} a_i \leq W$ and $\sum_{i \in S'} p_i$ is maximized.

In the subdivision tree $T'$ of $T$ has its weight not beyond $d$, we call the former version as the subdivision-constrained minimum spanning tree problem in the third version (the SCST-3 problem) and the latter version as the subdivision-constrained minimum spanning tree problem in the fourth version (the SCST-4 problem), and then we can present two strongly polynomial time algorithms to solve the SCST-3 problem and the SCST-4 problem, respectively. For simplicity of exposition, we use $(G = (V, E; w, c); +\infty; d)$ $\text{SCST}$, $(\{G = (V, E; w, c); +\infty; d\}$ $\text{VGSCT}$, respectively) to represent an instance of the SCST-3 problem (the SCST-4 problem, respectively).

This paper is divided into the following sections. In Section 2, we prove the $NP$-hardness of the GSCST problem and the VGSCT problem, by a reduction from the 0–1 knapsack problem, respectively. In Section 3, we present the proofs of polynomial equivalences among the GSCST problem, the VGSCT problem and the CST problem, and then we can design a PTAS to solve the GSCST problem and the VGSCT problem, by utilizing an algorithm [6] to solve CST problem. In Section 4, we present a polynomial time algorithm to solve the SCST-1 problem and then prove the fact that the same algorithm also solves the SCST-2 problem. In Section 5, we design two polynomial time algorithms to solve the SCST-3 problem and the SCST-4 problem, respectively. In Section 6, we conclude this paper with some remarks and discussion on future study.
The VGSCST problem is NP-hard.

Proof. We can prove the NP-hardness of the VGSCST problem, by transforming the 0–1 knapsack problem [4] to the VGSCST problem in polynomial operations.

For any instance \(I\) of the 0–1 knapsack problem [4], we construct an instance \(\tau(I)\) of the VGSCST problem as indicated in the proof of Theorem 1: a graph \(G\) with \(2n + 1\) vertices is shown in Fig. 1(a), where each edge \(e\) in the graph carries two numbers corresponding to \(w(e)\) and \(c(e)\), and each basic block is defined as Fig. 1(b). Denote \(a_{\max} = \max\{a_i | 1 \leq i \leq n\}\) and \(d = a_{\max} + 1\). For each \(1 \leq i \leq n\), the lower horizontal edge \(e_i\) has weight \(w(e_i) = a_i + d + 1\) and unit cost \(c(e_i) = a_{\max} + p_{\max} - a_i - p_i\). Here insert \((e_i) = 1\) in this case. And for each \(1 \leq i \leq n\), the left oblique edge \(e_{n+2i-1}\) has weight \(w(e_{n+2i-1}) = 1\) and unit cost \(c(e_{n+2i-1}) = a_{\max} + p_{\max}\) implying insert \((e_{n+2i-1}) = 0\) in this case, but the right oblique edge \(e_{n+2i}\) has weight \(w(e_{n+2i}) = d + 1\) and unit cost \(c(e_{n+2i}) = a_{\max} + p_{\max}\) implying insert \((e_{n+2i}) = 1\) in this case. Put the bound \(B = n(d + 2) + W\) for the instance \(\tau(I)\). The objective is to find a spanning tree \(T = (V, E_T)\) of \(G\) such that \(\sum_{e \in E_T} w(e) \leq B\) and \(\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))\) is minimized.

With the argument similar to the proof of Theorem 1, we obtain the following result: the instance \(I\) of the 0–1 knapsack problem has an optimal solution \(OPT_{I} = \{i_1, i_2, ..., i_k\}\), where \(1 \leq i_1 < i_2 < \cdots < i_k \leq n\), if and only if the instance \(\tau(I)\) of the VGSCST problem has an optimal solution \(OPT_{VGSCST} = \{e_{i_1}, e_{i_2}, ..., e_{i_k}\} \cup \{e_{n+j} : 1 \leq j \leq n\} \cup \{e_{n+j} : j \in \mathbb{S} - \{i_1, i_2, ..., i_k\}\}\). Moreover, the two optimal values must satisfy the equality \(\sum_{e \in \text{OPT}_{VGSCST}} w(e) + \text{insert}(e)c(e)\) = \(n(a_{\max} + p_{\max} + d + 2) - \sum_{i \in \text{OPT}_{I}} p_i\).

Thus, the optimal solution \(OPT_{I}\) of the instance \(I\) to the 0–1 knapsack problem implies that the spanning tree \(OPT_{GVSCST}\) in the graph \(G\) has an optimal solution to the instance \(\tau(I)\) of the VGSCST problem. And vice versa.

Since the 0–1 knapsack problem is NP-hard in Garey and Johnson [4], then the VGSCST problem is still NP-hard. Hence, we reach the conclusion of this theorem. (The details will be found in the Appendix.)

3. The equivalences among the three problems

Motivated by an interest in the polynomial equivalence between the Hitchcock problem and the minimum-cost flow problem in Papadimitriou and Steiglitz [10], we prove that the GSCST problem is polynomially equivalent to the CST problem, which leads the proof of Theorem 3; moreover, we present the fact that the VGSCST problem is also polynomially equivalent to the CST problem, which forms the proof of Theorem 4.

Theorem 3. The GSCST problem is polynomially equivalent to the CST problem.

Proof. To prove the assertion, it is sufficient to prove that the GSCST problem can be transformed to the CST problem in polynomial operations and vice versa.

For any instance \(J\), say \(|G = (V, E; w, c); B; d|GSCST\), of the GSCST problem, we construct an instance \(\tau(J)\) of the CST problem: a graph \(G' = (V, E; w', c')\) consists of the same structure as the graph \(G\) with \(n\) vertices and \(m\) edges, the bound \(W = B\), and for each edge \(e \in E\), define two nonnegative integers \(w'(e) = w(e)\) and \(c'(e) = \text{insert}(e)c(e)\), where \(\text{insert}(e) = \lceil \frac{w(e)}{d} \rceil - 1\). The objective is to find a spanning tree \(T' = (V, E_{T'})\) of \(G'\) such that \(\sum_{e \in E_{T'}} w'(e) \leq W\) and the value \(\sum_{e \in E_{T'}} c'(e)\) is minimized.

It is easy to see that there is an optimal solution for the instance \(J\) of the GSCST problem with the optimal value \(k\) if and only if there is an optimal solution for the instance \(\tau(J)\) of the CST problem with the optimal value \(k\). In addition, this transformation is executed in polynomial operations.

On the converse direction, for any instance \(J\), say \(|G = (V, E; w, c); W|CST\), of the CST problem, we construct an instance \(\alpha(J)\) of the GSCST problem: a graph \(G' = (V, E; w', c')\) consists of the same structure as the graph \(G\) with \(n\) vertices and \(m\) edges, a positive integer \(d = \max\{w(e) | e \in E\}\) and the bound \(W = B = W + (n - 1)d\), and for each edge \(e \in E\), define the two nonnegative integers \(w'(e) = w(e) + d\) and \(c'(e) = c(e)\), here \(d = \max\{w(e) | e \in E\}\) implies \(\text{insert}(e) = 1\) in this case. The objective is to find a spanning tree \(T' = (V, E_{T'})\) of \(G'\) such that \(\sum_{e \in E_{T'}} w'(e) \leq B\) and the value \(\sum_{e \in E_{T'}} c'(e)\) is minimized, equivalently, \(\sum_{e \in E_{T'}} c'(e)\) is minimized.

It is easy to see that there is an optimal solution to the instance \(J\) of the CST problem with the optimal value \(k\) if and only if there is an optimal solution to the instance \(\alpha(J)\) of the GSCST problem with the optimal value \(k\). In addition, this transformation is executed in polynomial operations.

Hence, we reach the conclusion of this theorem. ■

Theorem 4. The VGSCST problem is polynomially equivalent to the CST problem.

Proof. To prove the theorem, it suffices to show that the VGSCST problem can be transformed to the CST problem in polynomial operations and vice versa.

For any instance \(J\), say \(|G = (V, E; w, c); B; d|VGSCST\), of the VGSCST problem, we construct an instance \(\tau(J)\) of the CST problem: a graph \(G' = (V, E; w', c')\) consists of the same structure as the graph \(G\) with \(n\) vertices and \(m\) edges, the bound \(W = B\), and for each edge \(e \in E\), define the two nonnegative integers \(w'(e) = w(e)\) and \(c'(e) = w(e) + \text{insert}(e)c(e)\), where \(\text{insert}(e) = \lceil \frac{w(e)}{d} \rceil - 1\). The objective is to find a spanning tree \(T' = (V, E_{T'})\) of \(G'\) such that \(\sum_{e \in E_{T'}} w'(e) \leq W\) and the value \(\sum_{e \in E_{T'}} c'(e)\) is minimized, equivalently, \(\sum_{e \in E_{T'}} (w(e) + \text{insert}(e)c(e))\) is minimized.
It is easy to see that there is an optimal solution for the instance \(I\) of the VGSCST problem with the constraint \(\sum_{e \in E_T} w(e) \leq B\) and the optimal value \(k\) if and only if there is an optimal solution for the instance \(\tau(I)\) of the CST problem with the constraint \(\sum_{e \in E_T} w(e) \leq W\) and the optimal value \(k\). In addition, this transformation is executed in polynomial operations.

On the converse direction, for any instance \(J\), say \(G = (V, E; w, c); W\) of the CST problem, we construct an instance \(\alpha(J)\) of the VGSCST problem: a graph \(G' = (V, E; w', c')\) consists of the same structure as the graph \(G\) with \(n\) vertices and \(m\) edges, a positive integer \(d = \max\{|w(e)| \in E\}\) and the bound \(B = W + d(n - 1)\), and for each edge \(e \in E\), define the two nonnegative integers \(w'(e) = w(e) + d(\leq 2d)\) and \(c'(e) = c(e) + 2d - w'(e)(\geq 0)\), here \(d = \max\{|w(e)| \in E\}\) implies \(insert'(e) = 1\) in this case. The objective is to find a spanning tree \(T' = (V, E_T')\) of \(G'\) such that \(\sum_{e \in E_{T'}} w'(e) \leq B\) and the total cost value \(\sum_{e \in E_T}(w'(e) + insert'(e)c'(e))\) is minimized, equivalently, \(2d(n - 1) + \sum_{e \in E_{T'}} c(e)\) is minimized, where minimum is taken over all spanning trees of the graph \(G'\).

It is easy to see that there is an optimal solution for the instance \(J\) of the CST problem with the constraint \(\sum_{e \in E_T} w(e) \leq W\) and the optimal value \(k\) if and only if there is an optimal solution for the instance \(\alpha(J)\) of the VGSCST problem with the constraint \(\sum_{e \in E_T} w'(e) \leq B\) and the optimal value \(2d(n - 1) + k\). In addition, this transformation is executed in polynomial operations.

Hence, we reach the conclusion of this theorem. \(\blacksquare\)

Since Theorem 3 implies that the GSCST problem is polynomially equivalent to the CST problem and Theorem 4 implies that the VGSCST problem is also polynomially equivalent to the CST problem, thus each of these three problems is polynomially equivalent to one of the other two.

For any instance \(I\) of the GSCST problem, by constructing an instance \(\tau(I)\) of the CST problem in the proof of Theorem 3 and utilizing an algorithm [6] to solve the CST problem, we can design a PTAS to solve the GSCST problem, and the computational complexity is the same as that of the algorithm given in [6]; moreover, for any instance \(J\) of the VGSCST problem, by constructing an instance \(\alpha(J)\) of the CST problem in the proof of Theorem 4 and utilizing the PTAS to solve CST problem, we can present a PTAS to solve the VGSCST problem, and the computational complexity is the same as that of the algorithm given in [6]. We omit our algorithms here, the details can be found in Hassin and Levin [6].

4. The SCST-1 and SCST-2 problems

Since the GSCST problem is \(NP\)-hard by Theorem 1, there is no polynomial time algorithm to solve it optimally, unless \(P = NP\). In this section, we study the special version of the GSCST problem which is called as the SCST-1 problem, where each unit cost \(c(e)\) for each edge \(e\) in \(G\) is the same \(k\), and then we can design a polynomial time algorithm to optimally solve the SCST-1 problem, whose objective is to minimize the value of \(k\) times the number of vertices inserted in a spanning tree \(T\) of \(G\), equivalently, to minimize the number of vertices inserted in a spanning tree \(T\) of \(G\). Our strategy is to choose a minimum spanning tree \(T\) in the graph \(G = (V, E; w)\), and then insert the least vertices on the suitable edges in such a minimum spanning tree \(T\) to maintain the preceding constraints, and we finally obtain the spanning tree as desired.

We know that there are many polynomial time algorithms to solve the minimum spanning tree problem. To simply state our polynomial time algorithm to solve the SCST-1 problem, we need a polynomial time algorithm due to Berge and Ghouila-Houri [2] or Kruskal [8] or Prim [11] to solve the minimum spanning tree problem, whose running times are \(O(|E| \log |V|) [2]\) or \(O(|V|^2) [8, 11]\), respectively. For simplicity of exposition, we denote this algorithm as the MST.

Moreover, we need the following lemma [10] which plays an important role to ensure the correctness of our algorithm.

Lemma 1 ([10]). For a weighted graph \(G = (V, E; l)\), where \(l : E \rightarrow R^+\) is a weight function, if a minimum spanning tree \(T = (V, E_T)\) of \(G\) has the edge set \(E_T = \{e_1, e_2, \ldots, e_{n-1}\}\) to satisfy the property \(l(e_1) \leq l(e_2) \leq \cdots \leq l(e_{n-1})\), and if any spanning tree \(T_0 = (V, E_{T_0})\) of \(G\) has the edge set \(E_{T_0} = \{e'_1, e'_2, \ldots, e'_{n-1}\}\) to satisfy the property \(l(e'_1) \leq l(e'_2) \leq \cdots \leq l(e'_{n-1})\), then the inequality \(l(e'_k) \leq l(e_k)\) holds for each \(k = 1, 2, \ldots, n - 1\). \(\blacksquare\)

By utilizing the algorithm MST as a subroutine, we design a polynomial time algorithm for the SCST-1 problem in the following structural form:

Algorithm: SCST-1

Input: a weighted graph \(G = (V, E; w)\) and two integers \(B, d\);

Output: a minimum spanning tree \(T = (V, E_T)\) in \(G\) and the subdivision tree \(T'\) of \(T\) such that \(\sum_{e \in E_T} w(e) \leq B\) and each new edge in \(T'\) has its weight not beyond \(d\).

Begin

Step 1 Utilize the algorithm MST to compute a minimum spanning tree \(T = (V, E_T)\) in \(G\), depending on the weight function \(w : E \rightarrow R^+\);

Step 2 If \(w(T) = \sum_{e \in E_T} w(e) > B\) then output “infeasible”, stop;

Step 3 For each edge \(e\) in \(T = (V, E_T)\), having its weight \(w(e) > d\), insert the \(insert(e)\) vertices on the edge \(e\), depending on the preceding insertion processes, such that each subdivision edge in subdivision tree \(T'\) of \(T\) has weight at most \(d\);

Step 4 Output the spanning tree \(T\) and its subdivision tree \(T'\).

End of Algorithm SCST-1
Utilizing the algorithm SCST-1, we obtain the following result for the SCST-1 problem.

**Theorem 5.** The algorithm SCST-1 is a strongly polynomial time algorithm to solve the SCST-1 problem, its computational complexity is as the same as that of the algorithm MST, i.e., $\Theta(|E| \log |V|)$ [2] or $\Theta(|V|^2)$ [8,11], respectively.

**Proof.** Suppose that $T$ is a spanning tree produced by the algorithm SCST-1 and $T^*$ is an optimal spanning tree to the instance $G$ for the SCST-1 problem, i.e., the insert $(T^*) = \sum_{e \in E_{T^*}} \text{insert}(e)$ is minimum among all spanning trees of $G$. We shall prove that $T$ is also an optimal spanning tree to the instance $G$ for the SCST-1 problem with the optimal value $\text{insert}(T) = \sum_{e \in E_T} \text{insert}(e)$, equivalently, $\text{insert}(T) = \text{insert}(T^*)$.

Since the algorithm SCST-1 utilizes the algorithm MST, depending on the weight function $w : E \rightarrow R^+$, and the algorithm MST produces a minimum spanning tree $T = (V, E_T)$, without loss of generality, we may assume that $T = (V, E_T)$ has the edge set $E_T = \{e_1, e_2, \ldots, e_{n-1}\}$ with $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_{n-1})$. For any optimal spanning tree $T^* = (V, E_{T^*})$ to the instance $G$ of the SCST-1 problem, without loss of generality, we may also assume that $T^*$ has the edge set $E_{T^*} = \{e^*_1, e^*_2, \ldots, e^*_{n-1}\}$ with $w(e^*_1) \leq w(e^*_2) \leq \cdots \leq w(e^*_{n-1})$. Lemma 1 implies that the inequality $w(e_k) \leq w(e^*_k)$ holds for each $k = 1, 2, \ldots, n - 1$. Thus, by the property of vertex-insertion where $\text{insert}(e_k) = \lceil \frac{w(e_k)}{d} \rceil - 1$ and $\text{insert}(e^*_k) = \lceil \frac{w(e^*_k)}{d} \rceil - 1$, our strategy to the vertex-insertion processes implies that the inequality $\text{insert}(e_k) \leq \text{insert}(e^*_k)$ holds for each $k = 1, 2, \ldots, n - 1$.

Hence, we have $\text{insert}(T) = \sum_{e \in E_T} \text{insert}(e) = \sum_{k=1}^{n-1} \text{insert}(e_k) \leq \sum_{k=1}^{n-1} \text{insert}(e^*_k) = \sum_{e \in E_{T^*}} \text{insert}(e) = \text{insert}(T^*)$.

This shows that $T$ is an optimal spanning tree to the instance $G$ for the SCST-1 problem, too.

This establishes the conclusion of the theorem. ■

Now, we remark the fact that the SCST-2 problem keeps the constraint $\sum_{e \in E_T} w(e) \leq B$ and the objective is to minimize the value $\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))$, where minimum is taken over all spanning trees of the graph $G$. Using the reduction from the SCST-2 (as a special version of the VGSCST problem) to the SCST-1 (as a special version of the GSCST problem), we obtain the fact that the SCST-2 problem is polynomially equivalent to the SCST-1 problem by Theorems 3 and 4, then there is a polynomial time algorithm to solve the SCST-2 problem, by utilizing the algorithm SCST-1 as a subroutine. In fact, we can prove that the algorithm SCST-1 also directly solves the SCST-2 problem in the following theorem (the proof can be found in the Appendix).

**Theorem 6.** The algorithm SCST-1 is a strongly polynomial time algorithm to directly solve the SCST-2 problem, its computational complexity is as the same as that of the algorithm MST, i.e., $\Theta(|E| \log |V|)$ [2] or $\Theta(|V|^2)$ [8,11], respectively. ■

5. The SCST-3 and SCST-4 problems

In this section, we study the algorithm SCST-3 problem and the SCST-4 problem which are the special versions of the GSCST problem and the VGSCST problem, respectively, where we ignore the constraint $\sum_{e \in E_T} w(e) \leq B$ and maintain the different unit costs for different edges in a spanning tree $T$, the objective is to minimize either the value $\sum_{e \in E_T} \text{insert}(e)c(e)$ or the value $\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))$.

For any instance $\{G = (V, E; w, c); +\infty; d\}_\text{GSCST}$ of the SCST-3 problem $\{G = (V, E; w, c); +\infty; d\}_\text{VGSCST}$ of the SCST-4 problem, respectively, our strategy to optimally solve the SCST-3 problem (the SCST-4 problem, respectively) is to construct a new weighted graph $G' = (V, E; w')$ as the following method: the new graph $G' = (V, E; w')$ is the same structure of the graph $G = (V, E)$, and for each edge $e$ in $G' = (V, E; w')$, we define the new weight $w'(e) = \text{insert}(e)c(e)$ for the SCST-3 problem $w'(e) = w(e) + \text{insert}(e)c(e)$ for the SCST-4 problem, respectively, where $\text{insert}(e) = \lceil \frac{w(e)}{d} \rceil - 1$ is the number of the vertices inserted on the edge $e$ in this case. Then we utilize the algorithm MST [2,8,11] to find a minimum spanning tree in the graph $G' = (V, E; w')$, depending on the weight function $w' : E \rightarrow R^+$, and we finally obtain an optimal spanning tree as desired.

Our algorithm for the SCST-3 problem is presented in the following structural form:

**Algorithm:** SCST-3

**Input:** a weighted graph $G = (V, E; w, c)$ and a constant $d$;

**Output:** a spanning tree $T$ in $G$ and its subdivision tree $T'$ such that each subdivision edge in $T'$ of $T$ has its weight not beyond $d$.

Begin
Step 1 Construct the new graph $G' = (V, E; w')$ as well as $G$, where $w'(e) = \text{insert}(e)c(e)$ for each edge $e$;
Step 2 Utilize the algorithm MST to compute a minimum spanning tree $T' = (V, E; w')$;
Step 3 Construct the subdivision tree $T'$ of $T$ in the following steps: for each edge $e$ in $T$, having $\text{insert}(e) > 0$, insert the $\text{insert}(e)$ vertices on the edge $e$ of $T$, and for each edge $e$ in $T$, having $\text{insert}(e) = 0$, insert no vertex on the edge $e$ of $T$.

Step 4 Output the spanning tree $T$ of $G$ and the subdivision tree $T'$.

End of Algorithm SCST-3

Utilizing the algorithm SCST-3, we can obtain the main result for the SCST-3 problem.

Theorem 7. The algorithm SCST-3 is a strongly polynomial time algorithm to solve the SCST-3 problem, its computational complexity is as the same as that of the algorithm MST, i.e., $\Theta(|E| \log |V|)$ [2] or $\Theta(|V|^2)$ [8,11], respectively.

To solve the SCST-4 problem, we can change the first step in the algorithm SCST-3, i.e., we define new weight $w'(e) = w(e) + \text{insert}(e)c(e)$ for each edge $e$ at the step 1 in the algorithm SCST-3, then we can design a strongly polynomial time algorithm, denoted as the algorithm SCST-4, to solve the SCST-4 problem (the algorithm SCST-4 in details is omitted here).

Theorem 8. The algorithm SCST-4 is a strongly polynomial time algorithm to solve the SCST-4 problem, its computational complexity is as the same as that of the algorithm MST, i.e., $\Theta(|E| \log |V|)$ [2] or $\Theta(|V|^2)$ [8,11], respectively.

6. Conclusion

We study the general subdivision-constrained spanning tree problem (GSCST) and its variant problem (VGSCST) in this paper, and then we prove that these two problems are NP-hard, by a reduction from the 0–1 knapsack problem, respectively; we also present the fact that the GSCST problem and the VGSCST problem are both polynomially equivalent to the CST problem, respectively, which implies that each of these three problems is polynomially equivalent to one of the other two, and then there exists a PTAS to solve each of them; moreover, we design three polynomial time algorithms to solve the four special versions of the GSCST problem and the VGSCST problem, respectively, and the three algorithms run in the time $\Theta(|E| \log |V|)$ or $\Theta(|V|^2)$, respectively, heavily depending on the algorithm [2,8,11] to solve the minimum spanning tree problem. If we find other polynomial time algorithms of lower computational complexity to solve the minimum spanning tree problem, the computational complexity of our three polynomial time algorithms designed in this paper would be decreased.

Motivated by the open question whether there exists an FTPAS for the CST problem in Hassin and Levin [6], we would try to design an FTPAS for these two new NP-hard problems, as well as for the CST problem, which would give an affirmative answer to the Hassin–Levin’s open question [6] and finally further improve our results in this paper.

Acknowledgements

The authors are grateful to the two kind anonymous referees whose comments and suggestions have led to a substantially improved presentation of the paper.

Appendix. Proofs of Theorems 2 and 6–8

Theorem 2. The VGSCST problem is NP-hard.

Proof. We can prove the NP-hardness of the VGSCST problem, by transforming the 0–1 knapsack problem [4] to the VGSCST problem in polynomial operations.

Consider any instance $I$ of the 0–1 knapsack problem: given a set $S = \{1, 2, \ldots, n\}$ of $n$ objects, with specified sizes $a_i \in \mathbb{Z}^+$ and profits $p_i \in \mathbb{Z}^+$, and a ‘knapsack capacity’ $W$, find a subset $S' \subseteq S$ such that $\sum_{i \in S'} a_i \leq W$ and $\sum_{i \in S'} p_i$ is maximized.

We construct an instance $(J, I)$ of the VGSCST problem: a graph $G$ with $2n + 1$ vertices as shown in Fig. 1(a), where each edge $e$ in the graph carries two numbers corresponding to $w(e)$ and $c(e)$, each basic block is defined as Fig. 1(b). Denote $a_{\max} = \max\{a_i | 1 \leq i \leq n\}$, $p_{\max} = \max\{p_i | 1 \leq i \leq n\}$ and $d = a_{\max} + 1$. For each $1 \leq i \leq n$, the lower horizontal edge $e_i$ has weight $w(e_i) = a_i + d + 1$ and unit cost $c(e_i) = a_{\max} + p_{\max} - a_i - p_i$, here $\text{insert}(e_i) = \lceil \frac{w(e_i)}{d} \rceil - 1$ in this case. And for each $1 \leq i \leq n$, the left oblique edge $e_{n+2i-1}$ has weight $w(e_{n+2i-1}) = 1$ and unit cost $c(e_{n+2i-1}) = a_{\max} + p_{\max}$, implying $\text{insert}(e_{n+2i-1}) = 0$ in this case, but the right oblique edge $e_{n+2i}$ has weight $w(e_{n+2i}) = d + 1$ and unit cost $c(e_{n+2i}) = a_{\max} + p_{\max}$, implying $\text{insert}(e_{n+2i}) = 1$ in this case. Put the bound $B = W + n(d + 2)$ for the instance $J$. The objective is to find a spanning tree $T = \langle V, E_T \rangle$ of $G$ such that $\sum_{e \in E_T} w(e) \leq B$ and $\sum_{e \in E_T} (w(e) + \text{insert}(e)c(e))$ is minimized.

Now, we shall obtain the following result: the instance $I$ of the 0–1 knapsack problem has an optimal solution $OPT_{0-1} = \{i_1, i_2, \ldots, i_k\}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, if and only if the instance $(J, I)$ of the VGSCST problem has an optimal solution $OPT_{VGSCST} = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}, e_{n+2j+1} : 1 \leq j \leq n\} \cup \{e_{n+j} : j \in S - \{i_1, i_2, \ldots, i_k\}\}$; moreover, the two optimal values must satisfy the equality $\sum_{e \in OPT_{VGSCST}} (w(e) + \text{insert}(e)c(e)) = n(a_{\max} + p_{\max} + d + 2) - \sum_{i \in OPT_{0-1}} p_i$. 


In fact, if the 0–1 knapsack problem has an optimal solution \( \text{OPT}_{0-1} \) with the constraint \( \sum_{i \in \text{OPT}_{0-1}} a_i \leq W \) and the optimal value \( \sum_{i \in \text{OPT}_{0-1}} b_i \), by finding a spanning tree \( \text{OPT}_{\text{SCST}} \) in the graph \( G \) indicated as before, we obtain the constraint \( w(\text{OPT}_{\text{SCST}}) = \sum_{e \in \text{OPT}_{\text{SCST}}} w(e) = \sum_{i \in \text{OPT}_{0-1}} (a_i + d + 1) + n + \sum_{j \in \text{OPT}_{0-1}} d + 1 = \sum_{i \in \text{OPT}_{0-1}} a_i + n(d + 2) \leq W + n(d + 2) = B \), with the total cost value \( c(\text{OPT}_{\text{SCST}}) = \sum_{e \in \text{OPT}_{\text{SCST}}} c(e) = \sum_{i \in \text{OPT}_{0-1}} (a_i + d + 1) + (a_{\text{max}} + p_{\text{max}} - a_i - p_i) + n + \sum_{j \in \text{OPT}_{0-1}} (d + 1) + (a_{\text{max}} + p_{\text{max}}) = n(a_{\text{max}} + p_{\text{max}} + d + 2) - \sum_{i \in \text{OPT}_{0-1}} p_i \).

Hence the optimal solution \( \text{OPT}_{0-1} \) of the instance \( t \) to the 0–1 knapsack problem implies that the spanning tree \( \text{OPT}_{\text{SCST}} \) in the graph \( G \) is also an optimal solution to the instance \( \tau(1) \) of the VGCST problem. And vice versa.

Since the 0–1 knapsack problem is NP-hard in Garey and Johnson [4], then the VGCST problem is NP-hard. Hence, we reach the conclusion of this theorem.

\[ \blacksquare \]

**Theorem 6.** The algorithm SCST-1 is a strongly polynomial time algorithm to directly solve the SCST-2 problem, its computational complexity is as the same as that of the algorithm MST, i.e., \( \Theta(|E| \log |V|) \) [2] or \( \Theta(|V|^2) \) [8,11], respectively.

**Proof.** Suppose that \( T = (V, E_T) \) is a spanning tree produced by the algorithm SCST-1 and \( T^* = (V, E_{T^*}) \) is an optimal spanning tree to the instance \( G \) for the SCST-2 problem, i.e., the \( \sum_{e \in E_G} (w(e) + k \cdot \text{insert}(e)) \) is minimum among all spanning trees of \( G \). We shall prove that \( T \) is also an optimal spanning tree to the instance \( G \) for the SCST-2 problem, equivalently, \( \sum_{e \in E_T} (w(e) + k \cdot \text{insert}(e)) = \sum_{e \in E_{T^*}} (w(e) + k \cdot \text{insert}(e)) \).

Since the algorithm SCST-1 uses the algorithm MST, and the algorithm MST produces a minimum spanning tree \( T \), without loss of generality, we may assume that \( T = (V, E_T) \) has the edge set \( E_T = \{e_1, e_2, \ldots, e_{\text{max}}\} \) with \( w(e_1) \leq w(e_2) \leq \cdots \leq w(e_{\text{max}}) \). For any optimal spanning tree \( T^* = (V, E_{T^*}) \) to the instance \( G \) for the SCST-2 problem, without loss of generality, we may also assume that \( T^* = (V, E_{T^*}) \) has the edge set \( E_{T^*} = \{e'_1, e'_2, \ldots, e'_{\text{max}}\} \) with \( w(e'_1) \leq w(e'_2) \leq \cdots \leq w(e'_{\text{max}}) \). Lemma 1 implies that the inequality \( w(e'_r) \leq w(e'_r) \) holds for each \( r = 1, 2, \ldots, n - 1 \). Thus, by the property of vertex-insertion

where \( \text{insert}(e_r) = \lceil \frac{w(e_r)}{d} \rceil - 1 \) and \( \text{insert}(e'_r) = \lceil \frac{w(e'_r)}{d} \rceil - 1 \), our strategy to the vertex-insertion processes implies that the inequality \( \text{insert}(e_r) \leq \text{insert}(e'_r) \) holds for each \( r = 1, 2, \ldots, n - 1 \). So we obtain the facts \( k \cdot \text{insert}(e_r) \leq k \cdot \text{insert}(e'_r) \) for each \( r = 1, 2, \ldots, n - 1 \).

Hence, we have \( \sum_{e \in E_T} (w(e) + k \cdot \text{insert}(e)) = \sum_{r=1}^{n-1} w(e_r) + k \cdot \sum_{r=1}^{n-1} \text{insert}(e_r) \leq \sum_{r=1}^{n-1} w(e'_r) + k \cdot \sum_{r=1}^{n-1} \text{insert}(e'_r) = \sum_{e \in E_{T^*}} (w(e) + k \cdot \text{insert}(e)) \). This shows that \( T \) is an optimal spanning tree to the instance \( G \) for the SCST-2 problem, too.

The complexity analysis in the proof in Theorem 5 implies the whole algorithm runs in time \( \Theta(|E| \log |V|) \) [2] or \( \Theta(|V|^2) \) [8,11], respectively.

This establishes the conclusion of the theorem.

\[ \blacksquare \]

**Theorem 7.** The algorithm SCST-3 is a strongly polynomial time algorithm to solve the SCST-3 problem, its computational complexity is as the same as that of the algorithm MST, i.e., \( \Theta(|E| \log |V|) \) [2] or \( \Theta(|V|^2) \) [8,11], respectively.

**Proof.** Suppose that \( T \) is a spanning tree produced by the algorithm MST in the graph \( G' \) (also \( G \)) and \( T^* \) is an optimal spanning tree to the instance \( G \) for the SCST-3 problem, i.e., the optimal value \( \sum_{e \in E(T')} \text{insert}(e)c(e) \) is minimum among all spanning trees of \( G \). We shall prove that \( T \) is also an optimal spanning tree to the instance \( G \) for the SCST-3 problem, equivalently, \( \sum_{e \in E(T)} \text{insert}(e)c(e) = \sum_{e \in E(T')} \text{insert}(e)c(e) \).

The algorithm MST produces a minimum spanning tree \( T = (V, E_T) \) in \( G' = (V, E; w') \), without loss of generality, we may assume that \( T \) has the edge set \( E_T = \{e_1, e_2, \ldots, e_{\text{max}}\} \) with \( w'(e_1) \leq w'(e_2) \leq \cdots \leq w'(e_{\text{max}}) \). For any optimal spanning tree \( T^* = (V, E_{T^*}) \) to the instance \( G \) for the SCST-3 problem whose optimal value is \( \sum_{e \in E(T')} \text{insert}(e)c(e) \), without loss of generality, we may assume that \( T^* \) has the edge set \( E_{T^*} = \{e'_1, e'_2, \ldots, e'_{\text{max}}\} \) with \( w'(e'_1) \leq w'(e'_2) \leq \cdots \leq w'(e'_{\text{max}}) \), i.e., \( \text{insert}(e'_1)c(e'_1) \leq \text{insert}(e'_2)c(e'_2) \leq \cdots \leq \text{insert}(e'_{\text{max}})c(e'_{\text{max}}) \). Lemma 1 implies that the inequality \( w'(e_k) \leq w'(e_k) \), i.e., \( \text{insert}(e_k)c(e_k) \leq \text{insert}(e_k)c(e_k) \), holds for each \( k = 1, 2, \ldots, n - 1 \).

Hence, we have \( \sum_{e \in E(T)} \text{insert}(e)c(e) = \sum_{k=1}^{n-1} \text{insert}(e_k)c(e_k) \leq \sum_{k=1}^{n-1} \text{insert}(e_k)c(e_k) = \sum_{e \in E(T')} \text{insert}(e)c(e) \). This shows that \( T \) is also an optimal spanning tree to the instance \( G \) of the SCST-3 problem.

The computational complexity of the algorithm SCST-3 comes from the following analyses: (1) the construction of the graph \( G' \) needs \( \Theta(|E|) \) steps; (2) the algorithm MST implies that the step 2 needs \( \Theta(|E| \log |V|) \) [2] or \( \Theta(|V|^2) \) [8,11], respectively, steps to compute a minimum spanning tree \( T \), depending on the weight function \( w' : E \rightarrow R^+ \); (3) for the method to insertion, by treating one step to insert all \( \text{insert}(e) \) vertices on each edge \( e \), the step 3 needs at most \( \Theta(n) \) steps to insert such vertices on some suitable edges. Hence, the whole algorithm runs in time \( \Theta(|E| \log |V|) \) [2] or \( \Theta(|V|^2) \) [8,11], respectively.

This establishes the conclusion of the theorem.

\[ \blacksquare \]

We present our algorithm for the SCST-4 problem in the following structural form:

**Algorithm: SCST-4**

**Input:** a weighted graph \( G = (V, E; w, c) \) and a constant \( d \);

**Output:** a spanning tree \( T \) in \( G \) and its subdivision \( T' \) such that each new edge in the subdivision \( T' \) of \( T \) has its weight not beyond \( d \).
Theorem 8. The algorithm SCST-4 is a strongly polynomial time algorithm to solve the SCST-4 problem, its computational complexity is as the same as that of the algorithm MST, i.e., $O(|E| \log |V|)$ [2] or $O(|V|^2)$ [8,11], respectively.

Proof. Suppose that $T$ is a spanning tree produced by the algorithm MST in $G'$ (also $G$) and $T^*$ is an optimal spanning tree to the instance $G$ for the SCST-4 problem, i.e., the optimal value $\sum_{e \in E(T^*)} (w(e^*) + \text{insert}(e^*)c(e^*))$ is minimum among all spanning trees of $G$. We shall prove that $T$ is also an optimal spanning tree to the instance $G$, equivalently, $\sum_{e \in E(T)} (w(e) + \text{insert}(e)c(e)) = \sum_{e \in E(T^*)} (w(e^*) + \text{insert}(e^*)c(e^*))$.

The algorithm MST produces a minimum spanning tree $T = (V, E_T)$ in $G' = (V, E; w')$, without loss of generality, we may assume that $T$ has the edge set $E_T = \{e_1, e_2, \ldots, e_{n-1}\}$ with $w'(e_1) \leq w'(e_2) \leq \cdots \leq w'(e_{n-1})$. For any optimal spanning tree $T^* = (V, E_{T^*})$ to the instance $G$ for the SCST-4 problem whose optimal value is $w(T^*) + \sum_{e \in E(T^*)} (w(e^*) + \text{insert}(e^*)c(e^*))$, without loss of generality, we may assume that $T^*$ has the edge set $E_{T^*} = \{e_{j_1}, e_{j_2}, \ldots, e_{j_{n-1}}\}$ with $w'(e_{j_1}) \leq w'(e_{j_2}) \leq \cdots \leq w'(e_{j_{n-1}})$, i.e., $w(e_{j_1}') + \text{insert}(e_{j_1})c(e_{j_1}') \leq w(e_{j_2}') + \text{insert}(e_{j_2})c(e_{j_2}') \leq \cdots \leq w(e_{j_{n-1}}') + \text{insert}(e_{j_{n-1}})c(e_{j_{n-1}}')$. Lemma 1 implies that the inequality $w'(e_{j_r}) \leq w'(e_{j_r}')$, i.e., $w(e_{j_r}) + \text{insert}(e_{j_r})c(e_{j_r}) \leq w(e_{j_r}') + \text{insert}(e_{j_r})c(e_{j_r}')$, holds for each $r = 1, 2, \ldots, n-1$.

Hence, we have $\sum_{e \in E(T)} (w(e) + \text{insert}(e)c(e)) = \sum_{r=1}^{n-1} (w(e_{j_r}) + \text{insert}(e_{j_r})c(e_{j_r})) \leq \sum_{r=1}^{n-1} (w(e_{j_r}') + \text{insert}(e_{j_r}')c(e_{j_r}')) = \sum_{e \in E(T^*)} (w(e^*) + \text{insert}(e^*)c(e^*))$. This shows that $T$ is also an optimal spanning tree to the instance $G$ for the SCST-4 problem.

The computational complexity of the algorithm SCST-4 comes from the following analyses: (1) the construction of the graph $G'$ needs $O(|E|)$ steps; (2) the algorithm MST implies that the step 2 needs $O(|E| \log |V|)$ [2] or $O(|V|^2)$ [8,11], respectively, steps to compute a minimum spanning tree $T$, depending on the weight function $w : E \to R^+$; (3) for the method to insertion, by treating one step to insert all insert ($e$) vertices on each edge $e$, the step 3 needs at most $O(n)$ steps to insert such vertices on some suitable edges. Hence, the whole algorithm needs its running time as the same as the algorithm MST, i.e., $O(|E| \log |V|)$ [2] or $O(|V|^2)$ [8,11], respectively.

This establishes the conclusion of the theorem. ■