Available online at www.sciencedirect.com
SCIENCE DIRECT
Discrete Applied Mathematics 154 (2006) 1254–1263
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Minmax subtree cover problem on cacti

Hiroshi Nagamochi[a], Taizo Kawada[b]

[a]*Kyoto University, Yoshida Honmachi, Sakyo, Kyoto 606-8501, Japan*
[b]*Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Aichi 441-8580, Japan*

Received 27 August 2004; received in revised form 24 October 2005; accepted 27 October 2005
Available online 7 February 2006

Abstract

Let $G = (V, E)$ be a connected graph such that edges and vertices are weighted by nonnegative reals. Let p be a positive integer. The minmax subtree cover problem (MSC) asks to find a pair $(\mathscr{X}, \mathscr{T})$ of a partition $\mathscr{X} = \{X_1, X_2, \ldots, X_p\}$ of V and a set $\mathscr{T}$ of p subtrees $T_1, T_2, \ldots, T_p$, each T_i containing X_i so as to minimize the maximum cost of the subtrees, where the cost of T_i is defined to be the sum of the weights of edges in T_i and the weights of vertices in X_i. In this paper, we propose an $O(p^2n)$ time $(4 - 4/(p + 1))$-approximation algorithm for the MSC when G is a cactus.

Keywords: Approximation algorithm; Cactus; Graph partition; NP-hard; Tree; Vehicle routing

1. Introduction

Given a graph, the p-traveling salesmen problem (p-TSP) asks to find a set of p tours that cover all vertices in the graph, minimizing a given objective function. This type of problems arises in numerous applications such as the multi-vehicle scheduling problem [5]. Graphs are restricted to be paths or trees in some applications such as the task sequencing problem, the delivery scheduling by ships on a shoreline [12] and the scheduling of automated guided vehicles. The 1-TSP or p-TSP on paths or trees and analogous routing problems have been studied extensively (e.g., [1,2,6,7,12]).

Among these problems, this paper considers the minmax subtree cover problem, which is defined in the following.

Let $G = (V, E)$ be an undirected graph, where we may denote the vertex set and the edge set of G by $V(G)$ and $E(G)$, respectively. Let $n = |V(G)|$ and $m = |E(G)|$. We denote by (G, w, h) a graph G such that each edge e and each vertex v are weighted by nonnegative reals $w(e)$ and $h(v)$, respectively. A collection $\mathscr{X}$ of disjoint subsets $X_1, X_2, \ldots, X_k$ of V is called a *partition* of V if their union is V, where some X_i may be empty. A collection $\mathscr{X}$ of V is called a *p-partition* of V if $|\mathscr{X}| = p$. We denote $\sum_{v \in X} h(v)$ for a vertex set X by $h(X)$ and $\sum_{v \in F} w(v)$ for an edge set F by $w(F)$. For a weighted graph (H, w, h), we may denote $w(E(H))$ by $w(H)$ and $w(H) + h(V(H))$ by $\hat{w}(H)$.

Then the minmax subtree cover problem is described as follows.

Minmax subtree cover problem (*MSC*):

Input: An instance $I = (G, w, h, p)$ which consists of a connected graph G, an edge weight w, a vertex weight h and an integer $p \in [2, n]$.

E-mail addresses: nag@amp.i.kyoto-u.ac.jp (H. Nagamochi), taizo@pro.ics.tut.ac.jp (T. Kawada).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.10.013

Feasible solution: A p-partition $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ of V and a set of p subtrees $\mathcal{T} = \{T_1, T_2, \ldots, T_p\}$ of G such that $X_i \subseteq V(T_i)$.

Goal: Minimize $cost(\mathcal{X}, \mathcal{T}) := \max_{1 \leqslant i \leqslant p}\{w(T_i) + h(X_i)\}$.

That is, the MSC asks to find a set of p subtrees such that the union of the subtrees covers all vertices in V so as to minimize the maximum cost of the subtrees, where the cost of a subtree T_i is the sum of the weights of edges in T_i and the weights of vertices that are covered by T_i (each vertex is covered by exactly one of the subtrees). Subtrees in $\mathcal{T}$ are not necessarily edge-disjoint or vertex-disjoint. The MSC has an application in the multi-vehicle scheduling problem [6]. The next problem is closely related to the MSC.

Minmax rooted-subtree cover problem (*MRSC*):

Input: An instance $I = (G, w, h, r, p)$ which consists of a connected graph G, an edge weight w, a vertex weight h, a vertex r designated as a root, and an integer $p \in [2, n]$.

Feasible solution: A p-partition $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ of V and a set of p subtrees $\mathcal{T} = \{T_1, T_2, \ldots, T_p\}$ of G such that $X_i \cup \{r\} \subseteq V(T_i)$.

Goal: Minimize $cost(\mathcal{X}, \mathcal{T}) := \max_{1 \leqslant i \leqslant p}\{w(T_i) + h(X_i)\}$.

The MRSC asks to find a set of p subtrees such that each subtree contains r and the union of the subtrees covers all vertices in V, where the objective is to minimize the maximum cost of the subtrees. The MSC and MRSC are both NP-hard. If G is a tree, then there are several approximation algorithms to these problems. For the MSC on a tree G, Averbakh and Berman [4] presented a $(2 - 2/(p + 1))$-approximation algorithm that runs in $O(p^{p-1}n^{p-1})$ time, and Nagamochi and Okada [9] recently gave a polynomial time $(2 - 2/(p + 1))$-approximation algorithm with time complexity $O(p^2 n)$.

Averbakh and Berman [3] have given a linear time $\frac{4}{3}$-approximation algorithm for the MRSC with $p = 2$ on a tree G, and Nagamochi and Okada [9] proposed an $O(n \log \log_{1+\varepsilon/2} 3)$ time $(2 + \varepsilon)$-approximation algorithm for the MRSC with any integer $p \in [2, n]$ on a tree G, where $\varepsilon > 0$ is a prescribed number. Very recently, Nagamochi [8] proposed an $O(m + n \log n)$ time $(3 - 2/(p + 1))$-approximation algorithm for the MRSC on an arbitrary connected graph G.

Note that, for any solution to the MRSC, the set of edges used in the subtrees induces a connected spanning subgraph from G, implying that, for a minimum spanning tree T^* of G, $(w(T^*) + h(V))/p$ is a lower bound on the optimal value to the MRSC. However, no such conventional lower bound is known for the MSC, and no polynomial approximation algorithm has been obtained for the MSC on any class of non-tree graphs so far.

In this paper, we establish a framework for designing approximation algorithms for the MSC on arbitrary graphs, and then give an $O(p^2 n)$ time $(4 - 4/(p + 1))$-approximation algorithm for the MSC on a cactus G. This is the first approximation algorithm for the MSC on a class of non-tree graphs.

The rest of the paper is organized as follows. Section 2 introduces some terminology and presents preliminary results on the MSC. Section 3 gives a framework of designing approximation algorithms for the MSC. Section 4 analyzes the ratio between lower bounds on trees and cacti in order to design a $(4 - 4/(p + 1))$-approximation algorithm for the MSC on a cactus. Section 5 describes some concluding remarks.

2. Preliminaries

We denote by (G, w) an edge weighted graph G such that each edge e is weighted by a nonnegative real $w(e)$, where weight $w(e)$ for an edge $e = (u, v)$ with end vertices $u, v \in V$ may be denoted by $w(u, v)$. A vertex with degree 1 is called a *leaf* in G, and the set of leaves in G is denoted by $L(G)$. For a subgraph H of (G, w, h), (H, w) means a graph H in which each edge has the same weight in G (i.e., the edge weight of H is the restriction of w on $E(H)$). Similarly, we define (H, w, h) for a subgraph H of G. Let $h_{\max} = \max_{v \in V} h(v)$.

Let T be a tree. For a subset $X \subseteq V(T)$ of vertices, let $T\langle X\rangle$ denote the minimal subtree of T that contains X (where the leaves of $T\langle X\rangle$ will be vertices in X). In this paper, we say that $T\langle X\rangle$ is *induced* from T by X. A graph G is called a *cactus* if no two cycles in G share more than one vertex (see Fig. 1(a)).

We call an instance $I = (G, w, h, p)$ of the MSC a *tree instance* (resp., *cactus instance*) if G is a tree (resp., cactus), and denote the optimal value to I by $opt(I)$. We easily observe the following property. For a tree instance $I = (T, w, h, p)$ of the MSC,

$$opt(I) \geqslant \max\left\{\frac{\hat{w}(T)}{p}, h_{\max}\right\}, \tag{1}$$

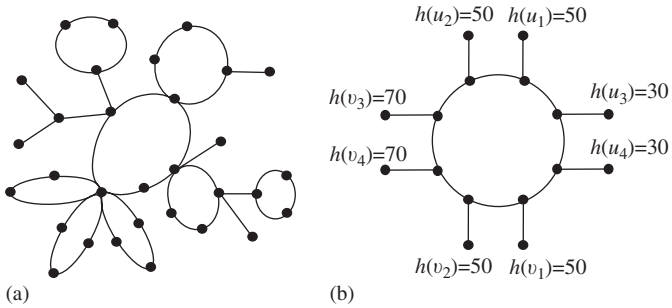

Fig. 1. Examples of cacti: (a) $G = (V, E)$ and (b) $I = (G, w, h, p = 4)$.

provided that there is an optimal solution $(\mathscr{X}, \mathscr{T})$ to I such that each edge is contained in some subtree $T_i \in \mathscr{T}$. In general, (1) does not hold. Nagamochi and Okada [10] have introduced the following lower bound on $opt(I)$. Given an instance $I = (G, w, h, p)$, a *valued subtree collection* of I is defined to be a set $\mathscr{S}$ of vertex-disjoint subtrees $S_1, S_2, \ldots, S_k \subseteq G$ such that each S_i is weighted by a positive integer p_{S_i} with $\sum_{S_i \in \mathscr{S}} p_{S_i} = p$. For a valued subtree collection $\mathscr{S}$ of I, consider a solution $(\mathscr{X}, \mathscr{T})$ such that $\bigcup_{T_i \in \mathscr{T}} E(T_i)$ induces from G components $S_1, S_2, \ldots, S_{|\mathscr{S}|}$ and each S_i contains p_{S_i} subtrees from $\mathscr{T}$. Then we see that

$$\lambda(\mathscr{S}) = \max\left\{ \frac{\hat{w}(S_i)}{p_{S_i}} \mid S_i \in \mathscr{S} \right\}$$

is a lower bound on the minimum $cost(\mathscr{X}, \mathscr{T})$ over all such solutions. Therefore, by considering the minimum $\lambda(\mathscr{S})$ over all valued subtree collections $\mathscr{S}$ of I, we obtain a lower bound on $opt(I)$. Define

$$\lambda^*(I) = \min\{\lambda(\mathscr{S}) \mid \text{ all valued subtree collections } \mathscr{S} \text{ of } I\}.$$

Lemma 1 (*Nagamochi and Okada [10]*). *For any instance I of the MSC, $opt(I) \geqslant \lambda^*(I)$.*

Nagamochi and Okada [10] have shown the following result.

Theorem 2 (*Nagamochi and Okada [10]*). *For a tree instance $I = (T, w, h, p)$ of the MSC, there exists a solution $(\mathscr{X}, \mathscr{T})$ such that $cost(\mathscr{X}, \mathscr{T}) \leqslant \max\{(2 - 2/(p+1))\lambda^*(I), h_{\max}\}$ holds and any two subtrees $T_i, T_j \in \mathscr{T}$ are edge-disjoint.*

To find such a solution in this theorem in polynomial time, they investigated a relation between the MSC and a problem of minimizing the maximum cost of vertex-disjoint subtrees. Based on the next result, an $O(p^2 n)$ time algorithm for constructing a solution in Theorem 2 has been obtained [9].

Theorem 3 (*Perl and Vishkin [11]*). *Let $(T = (V, E), w, h)$ be a weighted tree. For a given integer $p \geqslant 2$, a set F of $(p-1)$ edges that minimizes the maximum weight $\hat{w}(T')$ of subtrees T' in $(V, E - F)$ can be found in $O(n + \rho p(p + \log \Delta))$ time, where ρ and Δ denote the radius and the maximum degree of tree T, respectively.*

We remark that there is a cactus instance $I = (G, w, h, p)$ of the MSC such that, for any optimal solution $(\mathscr{X}, \mathscr{T})$, the set of all edges in subtrees in $\mathscr{T}$ does not induce a forest from G (i.e., all edges of some cycle in G are used in $\mathscr{T}$). Fig. 1(b) shows a cactus instance $I = (G, w, h, p = 4)$ of the MSC such that an optimal solution uses all edges in the cactus, where $h(v)$=0 for non-leaves v and $h(u_1)$=$h(u_2)$=30, $h(v_1)$=$h(v_2)$=70, and $h(u_3)$=$h(u_4)$=$h(v_3)$=$h(v_4)$=50 for leaves, and $w(e) = 1$ for all edges. Observe that $opt(I) = 105$ and an optimal solution consists of four subtrees T_i $(1 \leqslant i \leqslant 4)$ such that T_i is a path of five edges connecting u_i and v_i.

To utilize Theorem 2 to approximate the MSC on a cactus G, one may try to transform an optimal solution $(\mathscr{X}, \mathscr{T})$ of a cactus instance $I = (G, w, h, p)$ into an approximate solution of a tree instance $I' = (T, w, h, p)$ for a spanning tree T of G. However, there may be a subtree $T_i \in \mathscr{T}$ which contains an edge e not in T, and transforming T_i by replacing

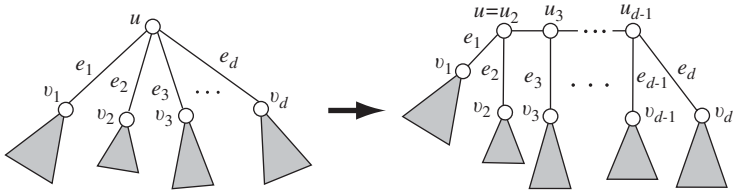

Fig. 2. Splitting a vertex u into $d-2$ vertices of degree 3.

e with other edges in the cycle C containing e may increase the cost of T_i by an arbitrarily large amount compared to its original cost.

For this, we examine the relation between the lower bounds $\lambda^*(I)$ and $\lambda^*(I')$ for cactus instances I and tree instances I'.

Lemma 4. *For an instance $I=(G=(V,E),w,h,p)$ with a connected graph G, there is a spanning subtree T of G such that $\lambda^*(I')\leqslant\lambda^*(I)$ holds for the tree instance $I'=(T,w,h,p)$.*

Proof. Let $\mathscr{S}$ be a valued subtree collection in I such that $\lambda(\mathscr{S})=\lambda^*(I)$. Since G is connected, there is a spanning tree T of G such that $E(T)\supseteq\bigcup_{S\in\mathscr{S}}E(S)$. Since $\lambda^*(I')\leqslant\lambda(\mathscr{S})$ holds for the tree instance $I'=(T,w,h,p)$, we have $\lambda^*(I')\leqslant\lambda(\mathscr{S})=\lambda^*(I)$ for the tree T. □

In general, it seems hard to find such a spanning tree T in Lemma 4 without knowing an optimal valued subtree collection $\mathscr{S}$ of I. When G is a tree-like graph, it might be possible to choose a spanning tree T of G such that $\lambda^*(I')$ approximates $\lambda^*(I)$. For an instance $I=(G=(V,E),w,h,p)$, let T be a spanning tree of G, and consider the tree instance $I'=(T=(V,E'),w,h,p)$. Let $\alpha\geqslant 1$ be a number such that

$$\lambda^*(I')\leqslant\alpha\cdot\lambda^*(I). \tag{2}$$

In Section 4, we shall see that factor α in (2) can be chosen as 2 when G is a cactus.

3. Algorithm for MSC

In this section, we give a framework for designing approximation algorithms for the MSC on arbitrary graphs, based on an approximation algorithm for the MSC on trees [9]. We first convert a given tree instance $I=(T,w,h,p)$ into another tree instance $\tilde{I}=(\tilde{T},w,h,p)$ by the following procedure.

LOWER_DEGREE

Input: A tree $(T=(V,E),w,h)$ with an edge weight w and a vertex weight h.

Output: A tree $(\tilde{T},w,h)$ with an edge weight w and a vertex weight h.

Step 1: For each non-leaf $v\in V(T)-L(T)$, we rename v by v', set $h(v'):=0$, and add a new leaf, which we now call v, introducing a new edge $e_v=(v',v)$ with $w(e_v)=0$, where we let the new v have the same weight $h(v)$ as before.

Step 2: For each vertex u with degree $d\geqslant 4$, execute the following procedure.

Let $e_1,e_2,\ldots,e_d$ be the edges incident to u. Split u into $d-2$ vertices $u_2(=u),u_3,\ldots,u_{d-1}$ introducing new vertices $u_3,u_4,\ldots,u_{d-1}$. Replace the end vertex u of each e_i, $i=2,3,\ldots,d-1$ (resp., of e_1 and e_d) with u_i (resp., with u_2 and u_{d-1}). Join the split vertices $u_2,u_3,\ldots,u_{d-1}$ by $d-3$ new edges $(u_2,u_3),\ldots,(u_{d-1},u_d)$ (see Fig. 2). Let weights of all introduced vertices and edges be zero, while the edges $e_1,e_2,\ldots,e_d$ have the same weights as before.

Let $\tilde{I}=(\tilde{T},w,h,p)$ be the resulting instance, in which every vertex has degree at most three. Then the next two lemmas hold.

Lemma 5. *For a tree instance $I=(T,w,h,p)$, let $\tilde{I}=(\tilde{T},w,h,p)$ be a tree instance obtained by* LOWER_DEGREE. *Then $\lambda^*(\tilde{I})\leqslant\lambda^*(I)$.*

Proof. Let $\mathscr{S} = \{S_1, S_2, \ldots, S_k\}$ be a valued subtree collection of I with $\lambda(\mathscr{S}) = \lambda^*(I)$, where each S_i is weighted by p_{S_i}. For each $S_i \in \mathscr{S}$, consider the subtree $\tilde{S}_i = \tilde{T}\langle V(S_i)\rangle$ of $\tilde{T}$. By the construction of $\tilde{I}$, $\tilde{S}_i$ has cost $w(\tilde{S}_i) + h(V(\tilde{S}_i)) = w(S_i) + h(V(S_i))$. Suppose that there is a vertex $z \in V(\tilde{T}) - V(T)$ that is not covered by any subtree $\tilde{S}_i$. By construction, such a vertex z satisfies $h(z) = 0$ and is connected to a vertex $u \in V(T)$ in $\tilde{T}$ via a path that consists of edges of weight zero. Thus by attaching such uncovered vertices to some subtrees $\tilde{S}_i$, we can obtain a valued subtree collection $\tilde{\mathscr{S}}$ of $\tilde{I}$ with $\lambda(\tilde{\mathscr{S}}) = \lambda^*(I)$. This implies $\lambda^*(\tilde{I}) \leqslant \lambda(\tilde{\mathscr{S}}) = \lambda^*(I)$. □

Lemma 6 (*Nagamochi and Okada [9]*). *Let $\tilde{I} = (\tilde{T}, w, h, p)$ be a tree instance of the MSC in which every vertex is of degree at most* 3 *and every internal vertex u satisfies $h(u) = 0$. For a solution $(\tilde{\mathscr{X}}, \tilde{\mathscr{T}})$ to $\tilde{I}$ such that $\tilde{\mathscr{T}}$ consists of edge-disjoint subtrees, there is a solution $(\mathscr{X}^*, \mathscr{T}^*)$ to $\tilde{I}$ with $cost(\mathscr{X}^*, \mathscr{T}^*) \leqslant cost(\tilde{\mathscr{X}}, \tilde{\mathscr{T}})$ such that $\mathscr{T}^*$ consists of vertex-disjoint subtrees.*

Based on the above properties, we obtain the following algorithm for the MSC.

Algorithm APPROX

Input: An instance $I = (G, w, h, p)$ of the MSC such that G is connected.

Output: A solution $(\mathscr{X}, \mathscr{T})$ to I.

Step 1: Find a spanning tree T of G, and let α be a factor such that (2) holds for the tree instance $I' = (T, w, h, p)$.

Step 2: Convert the tree instance $I' = (T, w, h, p)$ of the MSC into a tree instance $\tilde{I} = (\tilde{T}, w, h, p)$ by procedure LOWER_DEGREE.

Step 3: Find a set of p vertex-disjoint subtrees T_i^*, $i = 1, 2, \ldots, p$, in $\tilde{I}$ that minimizes $\max_{1 \leqslant i \leqslant p} \hat{w}(T_i^*)$. Let $\mathscr{X}^* = \{X_i^* = V(T_i^*) \mid i = 1, 2, \ldots, p\}$.

Step 4: Output solution $(\mathscr{X}, \mathscr{T})$ such that $\mathscr{X} = \{X_i = X_i^* \cap V(T) \mid i = 1, 2, \ldots, p\}$ and $\mathscr{T} = \{T\langle X_i\rangle \mid i = 1, 2, \ldots, p\}$.

Theorem 7. *For a given instance $I = (G, w, h, p)$ of the MSC,* APPROX *delivers in* $O(\tau_1 + p^2 n)$ *time a solution $(\mathscr{X}, \mathscr{T})$ such that*

$$cost(\mathscr{X}, \mathscr{T}) \leqslant \alpha \cdot \left(2 - \frac{2}{p+1}\right) opt(I) \tag{3}$$

and $\mathscr{T}$ consists of edge-disjoint subtrees, where τ_1 denotes the time to execute Step 1.

Proof. For the instance I' obtained in Step 1, $\lambda^*(I') \leqslant \alpha \cdot \lambda^*(I) \leqslant \alpha \cdot opt(I)$ holds by Lemma 1 and condition (2). By Lemma 5, $\lambda^*(\tilde{I}) \leqslant \lambda^*(I')$ holds for the instance $\tilde{I}$ in Step 2. By Theorem 2, there exists a solution $(\tilde{\mathscr{X}}, \tilde{\mathscr{T}})$ with

$$cost(\tilde{\mathscr{X}}, \tilde{\mathscr{T}}) \leqslant \max\left\{\left(2 - \frac{2}{p+1}\right)\lambda^*(\tilde{I}), h_{\max}\right\}$$

such that $\tilde{\mathscr{T}}$ consists of edge-disjoint subtrees in $\tilde{I}$. By Lemma 6, instance $\tilde{I}$ has a solution $(\mathscr{X}^*, \mathscr{T}^*)$ such that $cost(\mathscr{X}^*, \mathscr{T}^*) \leqslant cost(\tilde{\mathscr{X}}, \tilde{\mathscr{T}})$ and $\mathscr{T}^*$ consists of vertex-disjoint subtrees in $\tilde{I}$. Such a solution can be found in Step 3 in $O(p^2 n)$ time by using Theorem 3. Note that $\mathscr{T}$ in Step 4 is obtained from $\mathscr{T}^*$ in Step 3 by contracting all edges (of zero) introduced in the construction of $\tilde{T}$ from T. Hence $\mathscr{T}$ consists of edge-disjoint subtrees and satisfies $cost(\mathscr{X}, \mathscr{T}) = cost(\mathscr{X}^*, \mathscr{T}^*)$. Therefore, from the above inequalities and $opt(I) \geqslant h_{\max}$, we have (3). We easily see that APPROX can be implemented to run in $O(\tau_1 + p^2 n)$ time. □

4. Factor α for cacti

In this section, we show that every cactus G contains a spanning tree T such that (2) holds for factor $\alpha = 2$.

Lemma 8. *For an instance $I = (G, w, h, p)$ such that G is a cactus, let (T, w) be a minimum spanning tree of the edge weighted graph (G, w). Then $\lambda^*(I') \leqslant 2\lambda^*(I)$ holds for the tree instance $I' = (T, w, h, p)$.*

This lemma and Theorem 7 imply the next result.

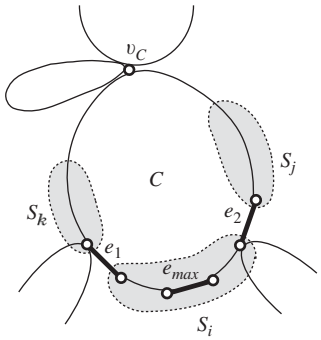

Fig. 3. Illustration for subtrees $S_i, S_j, S_k \in \mathcal{S}$ and edges $e_{\max}, e_1, e_2 \in E(C)$.

Corollary 1. *For a given instance $I = (G, w, h, p)$ of the MSC such that G is a cactus, a solution $(\mathcal{X}, \mathcal{T})$ with $cost(\mathcal{X}, \mathcal{T}) \leqslant (4 - 4/(p+1))\, opt(I)$ can be obtained in $O(p^2 n)$ time.*

In the rest of this section, we prove Lemma 8. Let $\lambda^* = \lambda^*(I)$, $\mathcal{S}$ be a valued subtree collection to $I = (G, w, h, p)$ with $\lambda(\mathcal{S}) = \lambda^*$, and (T, w) be a minimum spanning tree of (G, w). For notational simplicity, we assume that a given cactus G has no bridge, i.e., G consists only of cycles (if necessary, we add to each bridge $e = (u, v)$ a new edge $e' = (u, v)$ with $w(e') = w(e)$, and we can assume that no new edge is included in any subtree in $\mathcal{S}$).

Let $\overline{E} = E(G) - E(T)$, where, by the minimality of T, each edge $e \in \overline{E}$ has the maximum edge weight among edges in the cycle containing e. Let $E(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} E(S)$. Notice that $\mathcal{S}$ may not be a valued subtree collection of $I' = (T, w, h, p)$, i.e., some edges in $\overline{E}$ may be used in subtrees in $\mathcal{S}$.

We now give an algorithm for transforming $\mathcal{S}$ into a valued subtree collection $\mathcal{S}'$ of I'. The algorithm consists of two phases. In the first phase, we break a subtree $S \in \mathcal{S}$ into two *fractions* (or into a smaller subtree S and a fraction), where a fraction is a triplet (E', V', p') of an edge set $E' \subseteq E(S)$, a vertex set $V' \subseteq V(S)$ and an integer $p' \in [0, p_S]$. For a fraction $\delta = (E', V', p')$, define $\hat{w}(\delta) = w(E') + w(V')$ and $p(\delta) = p'$, respectively. During the first phase, we maintain a set $\mathcal{S}$ of vertex-disjoint subtrees S, each weighted by a positive integer p_S such that

$$\bigcup_{S \in \mathcal{S}} V(S) \subseteq V, \quad \sum_{S \in \mathcal{S}} p_S \leqslant p, \quad \lambda(\mathcal{S}) \leqslant \lambda^*. \tag{4}$$

The resulting $\mathcal{S}$ after the first phase is not a valued subtree collection to I' if $\bigcup_{S \in \mathcal{S}} V(S) \neq V$ or $\sum_{S \in \mathcal{S}} p_S < p$. In the second phase, we modify subtrees $S_i \in \mathcal{S}$ by attaching fractions δ to them so that the set of the resulting subtrees becomes a valued subtree collection $\mathcal{S}'$ of I' and $\lambda(\mathcal{S}') \leqslant 2\lambda^*$ holds.

4.1. Phase-1

We now describe a procedure for the first phase. For each subtree $S_i \in \mathcal{S}$, let $\Delta_i := \emptyset$, where Δ_i stores a set of fractions δ that will be added to S_i in the second phase.

Choose a cycle C_0 in G as a root cycle, and define distance $dist(v)$ for a vertex $v \in V$ to be the number of cycles which share edges with a simple path from a vertex in C_0 to v in G. For each cycle C, define $dist(C) = \min_{v \in V(C)} dist(v)$ and call the vertex $v \in V(C)$ with $dist(v) = dist(C)$ the *parent* of C. Then we number all cycles as $C_0, C_1, \ldots, C_r$ such that $dist(C_i) \leqslant dist(C_j)$ for any $i < j$.

For each cycle $C = C_r, C_{r-1}, \ldots, C_1$ in this order, we apply the following procedure CUT if there is a subtree $S_i \in \mathcal{S}$ with $E(S_i) \cap \overline{E} \cap E(C) \neq \emptyset$; we skip applying CUT to cycle C otherwise.

Procedure CUT

Let $e_{\max}$ be the edge in $E(S_i) \cap \overline{E} \cap E(C)$, and $v_C \in V(C)$ be the parent of C. Let $e_1, e_2 \in E(C) - E(S_i)$ be the edges incident to S_i (possibly $e_1 = e_2$), and S_k (resp., S_j) be the subtree that is adjacent to S_i via edge e_1 (resp., e_2) (possibly $S_k = S_j$). See Fig. 3.

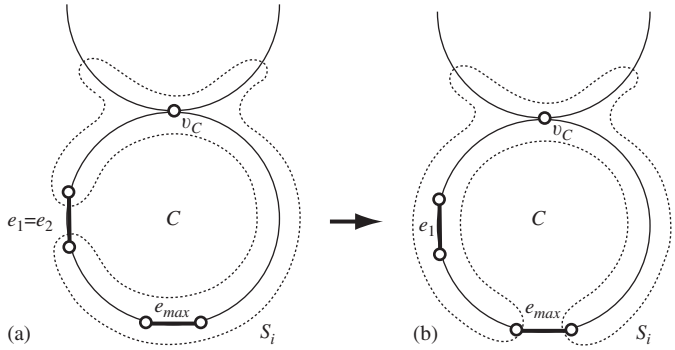

Fig. 4. Illustrations for Case 1 with $e_1 = e_2$.

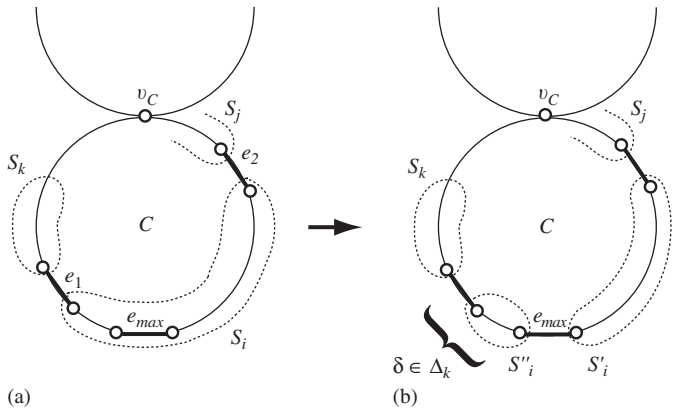

Fig. 5. Illustration for Case 2.

Case 1: $e_1 = e_2$ (hence $V(S_i) = V(C)$) (Fig. 4(a)). Let S_i be the subtree obtained from S_i by replacing $e_{\max}$ with e_1 (Fig. 4(b)).

Case 2: S_j or S_k does not contain $v_C \in V(C)$ (assume $v_C \notin V(S_k)$ without loss of generality) (Fig. 5(a)). Let S'_i and S''_i be two subtrees obtained from S_i by removing edge $e_{\max}$; S''_i is assumed to be adjacent to S_k without loss of generality. Let

$$p_{S'_i} := \left\lceil \frac{\hat{w}(S'_i)}{\hat{w}(S_i)} p_{S_i} \right\rceil, \quad p_{S''_i} := p_{S_i} - \left\lceil \frac{\hat{w}(S'_i)}{\hat{w}(S_i)} p_{S_i} \right\rceil.$$

Let Δ'' be the set of fractions in Δ_i that are created from the descendents of S''_i.

$$\delta := (V(S''_i), E(S''_i) \cup \{e_1\}, p_{S''_i}), \quad \Delta_k := \Delta_k \cup \{\delta\}\Delta'', \quad \Delta_i := \Delta_i - \Delta''$$

and

$$S_i := S'_i, \quad p_{S_i} := p_{S'_i}$$

(see Fig. 5(b)).

Case 3: $S_j = S_k$ and $v_C \in V(S_j)$ (Fig. 6(a)). Let

$$\delta := (V(S_i), (E(S_i) - \{e_{\max}\}) \cup \{e_1, e_2\}, p_{S_i}), \quad \Delta_j := \Delta_j \cup \{\delta\}$$

and

$$\mathcal{S} := \mathcal{S} - \{S_i\}, \quad \Delta_j := \Delta_j \cup \Delta_i$$

(see Fig. 6(b)).

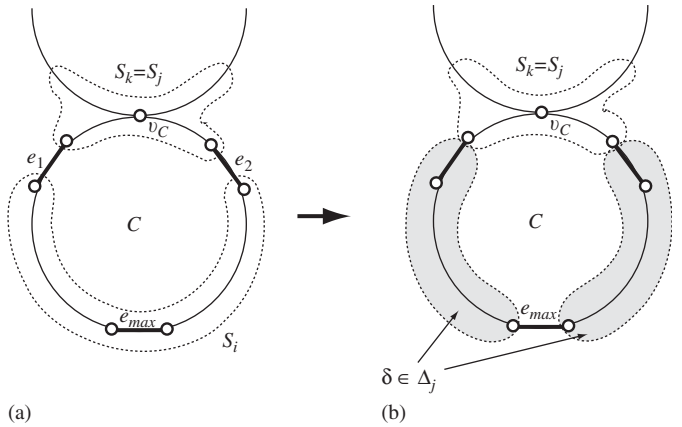

Fig. 6. Illustration for Case 3.

Claim 1. *A set $\mathscr{S}$ of subtrees obtained after Phase-*1 *satisfies* (4).

Proof. In Case 3, subtree S_i is simply removed from the current $\mathscr{S}$. In other cases, S_i and p_{S_i} will be modified. In Case 1, p_{S_i} remains unchanged, while $\hat{w}(S_i)$ never increases since $w(e_{\max}) \geqslant w(e_1)$. In Case 2, subtree S_i is modified into subtree S_i' such that

$$\frac{\hat{w}(S_i')}{p_{S_i'}} = \frac{\hat{w}(S_i')}{\lceil(\hat{w}(S_i')/\hat{w}(S_i))p\rceil} \leqslant \frac{\hat{w}(S_i')}{(\hat{w}(S_i')/\hat{w}(S_i))p_{S_i}} = \frac{\hat{w}(S_i)}{p_{S_i}},$$

which preserves property $\lambda(\mathscr{S}) \leqslant \lambda^*$. We easily observe that the first two conditions in (4) remain valid after each application of CUT. □

Claim 2. *Any fraction δ created during Phase-*1 *satisfies $\hat{w}(\delta)/p(\delta) \leqslant 2\lambda^*$ if $p(\delta) \geqslant 1$ and $\hat{w}(\delta) < \lambda^*$ if $p(\delta) = 0$.*

Proof. By Claim 1, any subtree $S \in \mathscr{S}$ after each application of CUT satisfies $p_S \geqslant 1$ and $\hat{w}(S)/p_S \leqslant \lambda^*$. Then a fraction $\delta = (V(S_i), (E(S_i) - \{e_{\max}\}) \cup \{e_1, e_2\}, p_{S_i})$ in Case 3 satisfies

$$\frac{\hat{w}(\delta)}{p(\delta)} = \frac{\hat{w}(S_i) - w(e_{\max}) + w(e_1) + w(e_2)}{p_{S_i}} \leqslant \frac{\hat{w}(S_i) + w(e_2)}{p_{S_i}} \leqslant 2\frac{\hat{w}(S_i)}{p_{S_i}} \leqslant 2\lambda^*.$$

We next consider a fraction $\delta = (V(S_i''), E(S_i'') \cup \{e_1\}, p_{S_i''})$ in Case 2. Note that $p_{S_i''} = p_{S_i} - \lceil p_{S_i}\hat{w}(S_i')/\hat{w}(S_i)\rceil = \lfloor p_{S_i}(\hat{w}(S_i'') + w(e_{\max}))/\hat{w}(S_i)\rfloor$. If $p_{S_i''} \geqslant 1$, then it holds

$$\frac{\hat{w}(\delta)}{p(\delta)} = \frac{\hat{w}(S_i'') + w(e_1)}{\lfloor((\hat{w}(S_i'') + w(e_{\max}))/\hat{w}(S_i))p_{S_i}\rfloor} \leqslant 2 \cdot \frac{\hat{w}(S_i'') + w(e_{\max})}{((\hat{w}(S_i'') + w(e_{\max}))/\hat{w}(S_i))p_{S_i}} = 2 \cdot \frac{\hat{w}(S_i)}{p_{S_i}} \leqslant 2\lambda^*.$$

On the other hand, if $p_{S_i''} < 1$, i.e., $p_{S_i}(\hat{w}(S_i'') + w(e_{\max}))/\hat{w}(S_i) < 1$, then we have

$$\hat{w}(\delta) = \hat{w}(S_i'') + w(e_{\max}) < \frac{\hat{w}(S_i)}{p_{S_i}} \leqslant \lambda^*.$$

This proves the claim. □

Claim 3. *For each $S_i \in \mathscr{S}$ obtained after Phase-*1, *Δ_i contains at most one fraction δ with $p(\delta) = 0$.*

Proof. Any fraction δ with $p(\delta) = 0$ is created in Case 2 when a cycle C is being scanned, and added to Δ_k of a subtree S_k with $V(S_k) \cap V(C) \neq \emptyset$ and $v_C \notin V(S_k)$. This implies that S_k cannot receive any other fraction δ' with $p(\delta') = 0$. Therefore, Δ_i contains at most one fraction δ with $p(\delta) = 0$. □

4.2. Phase-2

In the second phase, we paste fractions in Δ_i to the subtree $S_i \in \mathcal{S}$ by the following procedure.

Procedure PASTE

Let $\mathcal{S}$ be the set of subtrees obtained in Phase-1. For each subtree $S_i \in \mathcal{S}$ with $\Delta_i \neq \emptyset$, we modify S_i as follows: for $\Delta_i = \{(V'_j, E'_j, p'_j) \mid j = 1, 2, \ldots, k\}$, let

$$S_i^* := \left(V(S_i) \cup \bigcup_{1 \leqslant j \leqslant k} V'_j, E(S_i) \cup \bigcup_{1 \leqslant j \leqslant k} E'_j \right)$$

and

$$p_{S_i^*} := p_{S_i} + \sum\nolimits_{1 \leqslant j \leqslant k} p'_j.$$

Let $\mathcal{S}^*$ be the set of resulting subtrees.

By construction of fractions in CUT, we easily see that S_i computed by PASTE is a subtree and that the resulting set $\mathcal{S}^*$ is a set of vertex-disjoint subtrees of T such that $\bigcup_{S \in \mathcal{S}^*} V(S) = V$ and $\sum_{S \in \mathcal{S}^*} p_S = p$, i.e., $\mathcal{S}^*$ is a valued subtree collection of $I' = (T, w, h, p)$.

Lemma 9. *Let $\mathcal{S}^*$ be a valued subtree collection of $I' = (T, w, h, p)$ computed after Phase-2. Then $\lambda(\mathcal{S}^*) \leqslant 2\lambda^*$.*

Proof. Let $\mathcal{S}$ be a set of subtrees obtained after Phase-1. Consider a subtree $S_i^* \in \mathcal{S}^*$, and let $\Delta_i = \{\delta_1, \delta_2, \ldots, \delta_k\}$, where $\hat{w}(S_i^*) = \hat{w}(S_i) + \hat{w}(\delta_1) + \cdots + \hat{w}(\delta_k)$ and $p_{S_i^*} = p_{S_i} + p(\delta_1) + \cdots + p(\delta_k)$ hold for $S_i \in \mathcal{S}$. Then it suffices to show that

$$\frac{\hat{w}(S_i^*)}{p_{S_i^*}} = \frac{\hat{w}(S_i) + \hat{w}(\delta_1) + \cdots + \hat{w}(\delta_k)}{p_{S_i} + p(\delta_1) + \cdots + p(\delta_k)} \leqslant 2\lambda^*. \tag{5}$$

If $p(\delta_j) \neq 0$ for all $\delta_j \in \Delta_i$, then it holds

$$\frac{\hat{w}(S_i) + \hat{w}(\delta_1) + \cdots + \hat{w}(\delta_k)}{p_{S_i} + p(\delta_1) + \cdots + p(\delta_k)} \leqslant \max\left\{ \frac{\hat{w}(S_i)}{p_{S_i}}, \frac{\hat{w}(\delta_1)}{p(\delta_1)}, \ldots, \frac{\hat{w}(\delta_k)}{p(\delta_k)} \right\} \leqslant 2\lambda^*$$

by Claim 2. We now consider the case where Δ_i contains a fraction δ with $p(\delta) = 0$. By Claim 3, Δ_i contains exactly one such fraction, say δ_1 with $p(\delta_1) = 0$. Therefore, by Claim 2, we have

$$\begin{aligned} \frac{\hat{w}(S_i^*)}{p_{S_i^*}} &\leqslant \max\left\{ \frac{\hat{w}(S_i) + \hat{w}(\delta_1)}{p_{S_i}}, \frac{\hat{w}(\delta_2)}{p(\delta_2)}, \ldots, \frac{\hat{w}(\delta_k)}{p(\delta_k)} \right\} \\ &\leqslant \max\left\{ \frac{\hat{w}(S_i) + \lambda^*}{p_{S_i}}, 2\lambda^* \right\} \leqslant 2\lambda^*, \end{aligned}$$

as required. □

This completes the proof of Lemma 8.

5. Concluding remarks

In this paper, we have designed a framework for designing approximation algorithms for the MSC on an arbitrary graph, and have given an $O(p^2n)$ time $(4-4/(p+1))$-approximation algorithm for the MSC on a cactus. Our framework for designing algorithms for the MSC seems effective on classes of graphs which have a similar structure with trees. It would be interesting to investigate factors α in (2) for those classes such as outerplanar graphs.

Acknowledgments

This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Culture, Sports, Science and Technology of Japan. The authors would like to thank anonymous referees for their valuable comments.

References

[1] T. Asano, N. Katoh, K. Kawashima, A new approximation algorithm for the capacitated vehicle routing problem on a tree, J. Combin. Optim. 5 (2001) 213–231.
[2] I. Averbakh, O. Berman, Sales-delivery man problems on treelike networks, Networks 25 (1995) 45–58.
[3] I. Averbakh, O. Berman, A heuristic with worst-case analysis for minmax routing of two travelling salesmen on a tree, Discrete Appl. Math. 68 (1996) 17–32.
[4] I. Averbakh, O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, Discrete Appl. Math. 75 (1997) 201–216.
[5] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis, Time constrained routing and scheduling, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), Handbooks in Operations Research and Management Science, vol. 8, Network Routing, North-Holland, Amsterdam, 1995, 35–139.
[6] Y. Karuno, H. Nagamochi, 2-approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times, Discrete Appl. Math. 129 (2003) 433–447.
[7] Y. Karuno, H. Nagamochi, An approximability result of the multi-vehicle scheduling problem on a path with release and handling times, Theoret. Comput. Sci. A312 (2004) 267–280.
[8] H. Nagamochi, Approximating the minmax rooted-subtree cover problem, Electron. Inform. Comm. Eng. Trans. Fund. E88-A (2005) 1335–1338.
[9] H. Nagamochi, K. Okada, Polynomial time 2-approximation algorithms for the minmax subtree cover problem, Lecture Notes in Computer Science, vol. 2906, Springer, Berlin, 2003, pp. 138–147.
[10] H. Nagamochi, K. Okada, A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree, Discrete Appl. Math. 140 (2004) 103–114.
[11] Y. Perl, U. Vishkin, Efficient implementation of a shifting algorithm, technique for the partitioning, Discrete Appl. Math. 12 (1985) 71–80.
[12] H. Psaraftis, M. Solomon, T. Magnanti, T. Kim, Routing and scheduling on a shoreline with release times, Management Sci. 36 (1990) 212–223.