Journal of Algorithms **36,** 182–204 (2000)
doi:10.1006/jagm.2000.1088, available online at http://www.idealibrary.com on IDEAL®

A Polynomial Time Approximation Scheme for Optimal Product-Requirement Communication Spanning Trees

Bang Ye Wu

Department of Computer Science and Information Engineering, Shu-Te University, Yen Chau, Kao Shiung, Taiwan 824, Republic of China
E-mail: bangye@ms16.hinet.net

Kun-Mao Chao

Department of Life Science, National Yang-Ming University, Taipei, Taiwan 112, Republic of China
E-mail: kmchao@ym.edu.tw

and

Chuan Yi Tang

Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, Republic of China
E-mail: cytang@cs.nthu.edu.tw

Received November 2, 1998

Given an undirected graph with nonnegative edge lengths and nonnegative vertex weights, the routing requirement of a pair of vertices is assumed to be the product of their weights. The routing cost for a pair of vertices on a given spanning tree is defined as the length of the path between them multiplied by their routing requirement. The optimal product-requirement communication spanning tree is the spanning tree with minimum total routing cost summed over all pairs of vertices. This problem arises in network design and computational biology. For the special case that all vertex weights are identical, it has been shown that the problem is NP-hard and that there is a polynomial time approximation scheme for it. In this paper we show that the generalized problem also admits a polynomial time approximation scheme. © 2000 Academic Press

0196-6774/00 $35.00

1. INTRODUCTION

Consider the following problem in network design. We are given an undirected graph with a nonnegative length on each edge. The vertices may represent the cities and the edges represent the possible links to be built. For each vertex, there is a nonnegative weight representing the population of the city. For any pair of vertices, the routing requirement is assumed to be the product of the weights of the two vertices. The routing cost for a pair of vertices on a given spanning tree is defined as their routing requirement times the length of the path between them. The goal is to find a spanning tree such that the total routing cost summed over all pairs of vertices is the minimum.

The above problem is called the *optimal product-requirement communication spanning tree* (PROCT) problem. The PROCT problem is a generalization of the *minimum routing cost spanning tree* (MRCT) problem (also called the *shortest total path length spanning tree* problem in [2]), in which the weights on vertices are all identical. Both PROCT and MRCT problems are special cases of the *optimum communication spanning tree* problem, in which the routing requirement of a pair of vertices may be any nonnegative value [3]. The MRCT problem is NP-hard [2] and a 2-approximation algorithm was given in [5]. Recently, it was shown that the MRCT problem can be approximated within $(4/3 + \varepsilon)$ in polynomial time for any fixed $\varepsilon > 0$ [6]. More recently, a *polynomial time approximation scheme* (PTAS) for the MRCT problem and its possible application to computational biology were presented in [7].

In [8], two vertex-weighted generalizations of the MRCT problem were studied; one is the PROCT problem and the other is the *optimal sum-requirement communication spanning tree* (SROCT) problem in which the routing requirement of a pair of vertices is defined to be the sum of their weights. A 1.577-approximation algorithm for the PROCT problem and a 2-approximation algorithm for the SROCT problem were given in the paper [8]. Balancing between the total edge weights and the routing costs of spanning trees was also studied. In [9], an algorithm was given for constructing spanning trees with small routing costs and small total edge weights.

In [7], a PTAS for the minimum routing cost spanning tree problem, the unweighted version of PROCT, is presented by showing the following properties:

1. The MRCT problem with general inputs is equivalent to the problem with metric inputs (a complete graph in which edge lengths obey the triangle inequality).

2. A k-star is a spanning tree with at most k internal nodes. The minimum (routing cost) k-star is a good approximation solution for the metric MRCT problem. Its approximation ratio can be arbitrarily close to 1 as k becomes sufficiently large.

3. For a fixed k, the minimum k-star on a metric can be found in polynomial time.

In fact, the first and the second properties remain true for the PROCT problem. These can be obtained by straightforward generalizations of the previous results and will be shown later in the paper. However, there is no obvious method to generalize the algorithm for the minimum k-star to the weighted case. The *core* of a k-star is the subtree obtained by deleting all of the leaves of the k-star. The algorithm for an unweighted minimum k-star is to try all possible cores and determine the best connections of the leaves for each core. For a specified core, a configuration is to indicate how many leaves are connected to each internal node. When the core and configuration are fixed, it was shown that the best leaf connection can be found by solving an *assignment* problem. The number of possible cores is $\binom{n}{k}k^{k-2}$, which comes from the $\binom{n}{k}$ possible k-nodes subset and the k^{k-2} possible tree topologies of k nodes. In the unweighted case, the number of configurations is $O(n^{k-1})$ which is the number of ways to partition an integer $n-k$ into k different parts. Therefore, the total time is polynomial for fixed k since the assignment problem can be solved in $O(n^3)$ time [1]. However, in the weighted case there are k^{n-k} possible configurations (the same as the number of possible connections), hence such an algorithm will be exponential in time.

In [8], the 1.577-approximation algorithm for the PROCT problem is to find the minimum routing cost 2-star. For the minimum 2-star with a specified core, it is shown that the best leaf connection can be found by solving a *minimum cut* problem. Thus, the minimum 2-star can be found in polynomial time by trying all of the possible cores. But it is still unknown how to find the minimum k-star when $k > 2$ in polynomial time. In this paper, we show that the PROCT problem admits a PTAS by a technique that we call *scaling and rounding*. Scaling the input instances is a technique that has been used to balance the running time and the approximation ratio. For example, Lawler used the scaling technique to develop a PTAS for the knapsack problem [4].

The paper is organized as follows: In Section 2 we give some definitions and notations. The PTAS for the PROCT problem is presented in Section 3 and the lemmas to show the time complexity and approximation ratio are given in Sections 4, 5, 6, and 7.

2. PRELIMINARIES

For a graph G, $V(G)$ and $E(G)$ denote its vertex set and edge set, respectively. In this paper, we assume that the vertex and the edge weight functions are always nonnegative. Let r be a vertex weight function and V be a vertex set. We use $r(V)$ to denote the total weight of the vertices in V. For a graph S, we use $r(S)$ to denote $r(V(S))$. The input graph of the PROCT problem is assumed to be simple, undirected, and connected. We use n and R to denote the number of vertices and the total vertex weight of the input graph. It is assumed that there is no zero length edge in the graph. This assumption does not affect the generality of the problem since we can merge two vertices if they are connected by a zero length edge in the metric graph.

DEFINITION 1. Let $G = (V, E, w)$, S be a subgraph of G, and $i, j \in V$. By $\mathrm{SP}_S(i, j)$ we denote the shortest path from i to j on S. Let $w(S) = \sum_{e \in E(S)} w(e)$. The distance of i and j in S is $d_S(i, j) = w(\mathrm{SP}_S(i, j))$. We define $d_G(i, S) = \min_{j \in V(S)} d_G(i, j)$.

DEFINITION 2. The *metric closure* of G is the complete graph $\overline{G}$ with vertex set $V(G)$ and edge weight w, where $w(i, j) = d_G(i, j)$ for all $i, j \in V(G)$.

DEFINITION 3. Given a graph $G = (V, E)$ with edge weight function w and vertex weight function r, the PROCT problem is to find a spanning tree T of G such that $c(T, r) = \sum_{i, j \in V} r(i) r(j) d_T(i, j)$ is the minimum.

The definition of the MRCT problem is the same as that of PROCT except that each vertex weight is restricted to 1.

DEFINITION 4. For a graph G, a vertex weight r is canonical if $0 \le r(v) \le 1$ for every $v \in V(G)$ and $\max_{v \in V(G)}\{r(v)\} = 1$.

For the PROCT problem, we assume that the input vertex weight is canonical since we can divide each weight by the maximum one in a preprocessing stage.

DEFINITION 5. Let r be a vertex weight of a graph G and q be a positive number. We use $q \cdot r$ to denote the vertex weight function defined by $(q \cdot r)(v) = q \times r(v)$ for every $v \in V$.

Let T be any spanning tree of a graph and r_1, r_2 be vertex weight functions. By definition, it is easy to see that $c(T, q \cdot r) = q^2 c(T, r)$ for any nonnegative number q. Also, if $r_1(v) \le r_2(v)$ for any vertex v, then $c(T, r_2) \le c(T, r_2)$.

DEFINITION 6. A k-star is a spanning tree with at most k internal nodes. The core of a tree is the subtree obtained by deleting all of the

leaves from the tree, and the edges on a core are called the core edges of the core.

DEFINITION 7. Let A be a tree and V a vertex set containing no vertex of A. We use $\text{Star}(A, V)$ to denote the set of trees, in which every tree contains A as a subtree and all vertices in V are leaves of the tree.

3. THE PTAS FOR THE PROCT PROBLEM

A *balanced k-star* is a special kind of k-star. Similar to the result for the MRCT problem, the minimum routing cost balanced k-star is a good approximate solution. The difficulty of approximating the PROCT is how to find the minimum routing cost k-star. Assume that the vertex weights are all integers. We may find the minimum routing cost k-star of the weighted case by transforming the instance into an unweighted case. For a vertex with weight x, we generate x copies of the vertex and connect the copies by edges of zero length. After the transformation, one may find the minimum routing cost k-star by the algorithm developed in [7] for the unweighted case. However, the time complexity depends on the total vertex weight instead of the input size.

The key point of our PTAS is to find a k-star whose routing cost approximates the cost of the minimum balanced k-star and whose approximation ratio can be arbitrarily close to 1. The approximation is based on a technique that we call *scaling and rounding*. Let G be a metric graph and A be the core of a balanced k-star. For each vertex v not in $V(A)$ we need to determine a vertex in A and connect v to it. By a selected threshold, we first divide $V(G) \setminus V(A)$ into a light part and a heavy part according to their weights. Then, for each vertex in the heavy part, we enlarge their weights by a scaling factor and round them to integers. When the weights are all integers, the number of configurations is polynomial in the total enlarged weight of the vertices in the heavy parts. Therefore, the best connection (with respect to the enlarged weights) can be determined by an algorithm similar to the one in the unweighted case. Finally, each vertex in the light part is connected to its closest vertex in A. It will be shown that the approximation ratio and the time complexity are determined by k, the scaling factor, and by the threshold for dividing the vertices into the light and heavy parts. The PTAS is given below.

ALGORITHM. PTAS_PROCT.
Input: A graph G with a canonical vertex weight r, a positive number $\lambda < 1$, and two positive integers q and k.
Output: An approximate product-requirement communication spanning tree of G.

Step 1. Construct the metric closure $\overline{G} = (V, E, w)$ of G.
Step 2. /* Step 2 is performed on $\overline{G}$. */
For each possible tree A spanning no more than k vertices do
/* assume $V(A) = \{a_i \mid 1 \le i \le \bar{k}\}$, $\bar{k} \le k$, and $V \setminus V(A) = \{1 \dots n - \bar{k}\}$ */
Step 2.1. Sort and relabel the vertices in $V \setminus V(A)$ such that $r(i) \le r(i+1)$ $\forall i$
Step 2.2. Find the maximum j such that $r(\{1 \dots j\}) \le \lambda R$.
Let $V_L = \{1 \dots j\}$ and $V_H = \{j+1 \dots n - \bar{k}\}$ and $\mu = r(j+1)$.
Also let $R_L = r(V_L)$.
Step 2.3. Let $\bar{r}(v) = \lfloor qr(v)/\mu \rfloor$, $\forall v \in V_H$, and $\bar{r}(v) = qr(v)/\mu$, $\forall v \in V(A)$.
Step 2.4. Find the tree $T_1 \in \text{Star}(A, V_H)$ with minimum cost $c(T_1, \bar{r})$.
Step 2.5. Construct T_A from T_1 by connecting v to the closest vertex in A, $\forall v \in V_L$.
Step 3. Let T^* be the tree constructed in Step 2 with minimum $c(T^*, r)$. Transform T^* to a spanning tree T of G such that $c(T, r) \le c(T^*, r)$.
Step 4. Output T.

In order to show the time complexity and approximation ratio, we shall prove the following:

1. Let T^* be a $(1+\varepsilon)$-approximation solution of the PROCT problem with input $\overline{G}$. In $O(n^3)$ time, it can be transformed into a $(1+\varepsilon)$-approximation solution of the PROCT problem with input G (Section 4, Corollary 4).

2. There exists a *balanced* k-star X of $\overline{G}$, which is a $(k+3)/(k+1)$-approximation solution. A balanced k-star is a special k-star and will be defined later (Section 5, Lemma 5).

3. Let T_A be the tree constructed in Step 2.5 and X_A be the minimum balanced k-star with core A. $c(T_A, r) \le ((1+q^{-1})^2 + \lambda(k+3)^2/(k+1))c(X_A, r)$, whenever X_A exists (Section 6, Lemma 8).

4. The time complexity of Step 2.4 is $O((qR/\mu)^k)$ (Section 7, Lemma 13).

Based on the above results, we show the approximation ratio and time complexity in the following theorem:

THEOREM 1. **PTAS_PROCT** *is a PTAS for the PROCT problem with time complexity* $O(k^{k-1}n^{2k}(q/\lambda)^k)$ *and approximation ratio* $((1+q^{-1})^2 +$

$\lambda((k+3)^2/(k+1)))((k+3)/(k+1))$ *for any positive integers* q *and* k *and positive number* $\lambda < 1$.

Proof. The approximation ratio follows directly from the above results, which will be proved later. Also by the above results, the time complexity is $O(k^{k-1}n^k(qR/\mu)^k)$. From the choice of μ in Step 2.2, $(j+1)\mu \geq \sum_{1 \leq i \leq j+1} r(i) > \lambda R$. We have $\mu > R/n$. Thus, the time complexity is $O(k^{k-1}n^{2k}(q/\lambda)^k)$. It can be easily shown that the approximation ratio approaches 1 as q, λ^{-1}, and k go to infinity. Therefore, for any desired approximation ratio $1+\varepsilon > 1$, we can choose suitable q, λ, and k and the time complexity is polynomial when they are fixed. ∎

4. REDUCTION TO A METRIC

Let $G = (V, E, w)$ and h be the edge weight of its metric closure $\overline{G}$. In this section we show how a spanning tree of $\overline{G}$ can be transformed into a spanning tree of G without increasing cost. The transformation algorithm was proposed in [7] for the MRCT problem and was shown to hold for the PROCT problem in [8]. The proof in [8] is a straightforward generalization of the one in [7]. We only describe how it works and the proof is given in Appendix A.

Any edge (a, b) in $\overline{G}$ is called a *bad edge* if $(a, b) \notin E$ or $w(a, b) > h(a, b)$. Given any spanning tree T of $\overline{G}$, the algorithm iteratively replaces the bad edges until there are no bad edges left. Since the resulting tree has no bad edge, it can be thought of as a spanning tree of G with the same cost. It was proved in [8] that there will be at most $O(n^2)$ iterations and the total time complexity is $O(n^3)$. Furthermore, the cost never increases while replacing the bad edges. The result is summarized in the following lemma:

LEMMA 2. *Given a spanning tree* T *of* $\overline{G}$, *there is an algorithm which can construct a spanning tree* Y *of* G *with* $c(Y, r) \leq c(T, r)$ *in* $O(n^3)$ *time*.

Let $\mathrm{PROCT}(G, r)$ denote the optimal solution of the PROCT problem with input graph G and vertex weight r. The above lemma implies that

$$c(\mathrm{PROCT}(G, r), r) \leq c\big(\mathrm{PROCT}(\overline{G}, r), r\big).$$

It is easy to see that the optimal cost for input G is no less than the one for input $\overline{G}$. Therefore, we have the following corollaries.

COROLLARY 3. $c(\mathrm{PROCT}(G, r), r) = c(\mathrm{PROCT}(\overline{G}, r), r)$.

COROLLARY 4. *Let T^* be a $(1 + \varepsilon)$-approximation solution of the PROCT problem with input $\overline{G}$. In $O(n^3)$ time, it can be transformed into a $(1 + \varepsilon)$-approximation solution of the PROCT problem with input G.*

5. THE BALANCED k-STARS

We now give the definition of the balanced k-stars.

DEFINITION 8. Let T be a spanning tree of G, S a connected subgraph of T, and H the subgraph obtained by removing S from T. Let $\delta \leq 1/2$ be a positive real number. S is a *δ-separator* of T if $r(B) \leq \delta R$ for every connected component in H. A δ-separator S is *minimal* if any proper subgraph of S is not a δ-separator of T.

DEFINITION 9. Let k be a positive integer. A balanced k-star is a spanning tree with at most k internal nodes and its core is a minimal $(2/(k + 3))$-separator of the spanning tree.

Essentially, Lemma 5 is a natural generalization of the one in [7]. In [7], it was proved that there is a k-star X which is a $(k + 3)/(k + 1)$-approximation solution for the MRCT problem. The k-star X is also a balanced k-star. The modification of the definition is for the sake of showing the approximation ratio. The proof can be obtained by just replacing the vertex cardinalities with the total weight of the corresponding vertex sets in all relevant definitions and lemmas. It is given in Appendix B.

LEMMA 5. *There exists a balanced k-star of $\overline{G}$ which is a $(k + 3)/(k + 1)$-approximation solution of the PROCT problem with input $\overline{G}$.*

6. APPROXIMATION RATIO

In this section, we show the approximation ratio of the PTAS. We first define the routing load and the routing cost on an edge and show that the cost of a tree can be computed by summing up the routing cost on all tree edges.

DEFINITION 10. Let T be any spanning tree of a graph G and r be a vertex weight function. For any edge $e = (u, v) \in E(T)$, we define the *routing load* on the edge e to be $l(T, r, e) = 2r(T_u)r(T_v)$, where T_u and T_v are the two subtrees resulting from deleting e. The *routing cost* on the edge e is, defined to be $l(T, r, e)w(e)$.

LEMMA 6. *Let T be any spanning tree of a graph $G = (V, E, w)$ and r be a vertex weight function. $c(T, r) = \sum_{e \in E(T)} l(T, r, e)w(e)$.*

Proof.

$$\begin{aligned} c(T,r) &= \sum_{i,j \in V} r(i)r(j)d_T(i,j) \\ &= \sum_{i,j \in V} r(i)r(j)\left(\sum_{e \in SP_T(i,j)} w(e) \right) \\ &= \sum_{e \in E(T)} \left(\sum_{e \in SP_T(i,j)} r(i)r(j) \right) w(e) \\ &= \sum_{e \in (T)} l(T,r,e)w(e). \end{aligned}$$

∎

Note that the above lemma also implies that the routing cost of a tree can be computed in linear time. The following lemma shows that the load on each core edge will not be increased too much by the insertion of the light leaves.

LEMMA 7. *Let T_1 and T_A be the trees constructed in Steps* 2.4 *and* 2.5 *in Algorithm* **PTAS_PROCT** *respectively. $(l(T_A, r, e) - l(T_1, r, e)) \le 2RR_L$, for every edge $e \in E(A)$.*

Proof. Consider the light leaves one by one as they are inserted. For any $e \in E(A)$ and any $v \in V_L$, whenever v is inserted the load increase on e is no more than $2r(v)R$. Summing over all $v \in V_L$, the total load increase on e is no more than $2RR_L$. ∎

Now we show the approximation ratio below:

LEMMA 8. *Let T_A be the tree constructed in Step* 2.5 *and X_A be the minimum balanced k-star with core A. $c(T_A, r) \le ((1 + q^{-1})^2 + \lambda(k + 3)^2/(k + 1))c(X_A, r)$, whenever X_A exists.*

Proof. Let $U = V(A) \cup V_H$. T_1 and X_1 are the trees obtained by deleting the leaf set V_L from T_A and X_A, respectively. Note that $X_1 \in \text{Star}(A, V_H)$. Since $c(T_1, \bar{r})$ is minimum among the trees in $\text{Star}(A, V_H)$, we have

$$c(T_1, \bar{r}) \le c(X_1, \bar{r}). \tag{1}$$

Since $\bar{r}(v) \le qr(v)/\mu$ for any $v \in U$,

$$c(X_1, \bar{r}) \le c\left(X_1, \left(\frac{q}{\mu}\right) \cdot r\right) = \left(\frac{q}{\mu}\right)^2 c(X_1, r). \tag{2}$$

By (1) and (2), we obtain

$$c(T_1, \bar{r}) \le \left(\frac{1}{\mu}\right)^2 c(X_1, r). \tag{3}$$

For $v \in V_H$, $qr(v)/\mu \le \bar{r}(v) + 1 \le (1 + q^{-1})\bar{r}(v)$, and for $v \in V(A)$, $qr(v)/\mu = \bar{r}(v) \le (1 + q^{-1})\bar{r}(v)$. Therefore, $qr(v)/\mu \le (1 + q^{-1})\bar{r}(v)$ for any $v \in U$. Then

$$c\left(T_1, \frac{q}{\mu} \cdot r\right) \le c\left(T_1, \left(1 + q^{-1}\right) \cdot \bar{r}\right) = \left(1 + q^{-1}\right)^2 c(T_1, \bar{r}). \tag{4}$$

Since $c(T_1, r) = (\mu/q)^2 c(T_1, (q/\mu) \cdot r)$, by (3) and (4),

$$c(T_1, r) \le \left(\frac{\mu}{q}\right)^2 \left(1 + q^{-1}\right)^2 c(T_1, \bar{r}) \le \left(1 + q^{-1}\right)^2 c(X_1, r). \tag{5}$$

For a subset B of leaves in a tree T, let $C_L(T, B)$ denote the total routing cost on the edges connecting the leaves in B, i.e., $C_L(T, B) = 2\sum_{i \in B} r(i)(r(T) - r(i))w(i, fa(T, i))$, where $fa(T, i)$ is the neighbor of the leaf i in T. By Lemma 6,

$$c(T_A, r) = \sum_{e \in E(A)} l(T_A, r, e)w(e) + C_L(T_A, V_H) + C_L(T_A, V_L).$$

Since

$$\begin{aligned} C_L(T_A, V_H) - C_L(T_1, V_H) &= 2 \sum_{i \in V_H} r(i) R_L w(i, fa(T_A, i)) \\ &\le 2 \sum_{i \in V_H} r(i) R_L \left(w(i, fa(X_A, i)) + w(A)\right) \\ &= C_L(X_A, V_H) - C_L(X_1, V_H) + 2R_H R_L w(A), \end{aligned}$$

we have

$$\begin{aligned} c(T_A, r) \le c(T_1, r) &+ \sum_{e \in E(A)} \left(l(T_A, r, e) - l(T_1, r, e)\right)w(e) \\ &+ C_L(T_A, V_L) + C_L(X_A, V_H) - C_L(X_1, V_H) + 2R_H R_L w(A). \end{aligned} \tag{6}$$

Similarly,

$$c(X_A, r) \geq c(X_1, r) + C_L(X_A, V_H) - C_L(X_1, V_H) + C_L(X_A, V_L). \quad (7)$$

Since, on T_A, the light leaves are connected to the closest nodes in $V(A)$, $C_L(T_A, V_L) \leq C_L(X_A, V_L)$. Then by (5), (6), (7), and Lemma 7, we have

$$\begin{aligned} c(T_A, r) &\leq \left(1 + q^{-1}\right)^2 c(X_A, r) \\ &\quad + \sum_{e \in E(A)} \left(l(T_A, r, e) - l(T_1, r, e)\right) w(e) + 2R_H R_L w(A) \\ &\leq \left(1 + q^{-1}\right)^2 c(X_A, r) + 2R_L(R + R_H) w(A). \end{aligned} \quad (8)$$

If $k = 1$, the lemma holds trivially since there is no edge in A. In the following, we assume $k > 1$. For a balanced k-star, since its core is a minimal $(2/(k+3))$-separator, the routing load on any core edge is no less than $2(2R/(k+3))(R - 2R/(k+3)) = 4R^2(k+1)(k+3)^{-2}$. We have $l(X_A, r, e) \geq 4R^2(k+1)(k+3)^{-2}$ for every $e \in E(A)$. Thus, $c(X_A, r) \geq 4R^2(k+1)(k+3)^{-2} w(A)$. Since $R_L \leq \lambda R$, by (8), we have

$$\begin{aligned} c(T_A, r) &\leq \left(\left(1 + q^{-1}\right)^2 + \lambda(2 - \lambda)(k+3)^2/(2k+2)\right) c(X_A, r) \\ &\leq \left(\left(1 + q^{-1}\right)^2 + \lambda(k+3)^2/(k+1)\right) c(X_A, r). \end{aligned}$$

∎

7. BEST CONNECTION OF LEAVES WITH INTEGER WEIGHTS

In this section, we shall present how one may find the minimum routing cost spanning tree in $\mathrm{Star}(A, V_H)$. Any tree in $\mathrm{Star}(A, V_H)$ can be described by a partition $(L_1, L_2, \ldots, L_k)$ of V_H, in which L_i is the set of vertices connected to the vertex a_i of A. In [7], there is a polynomial time algorithm for the unweighted version of this subproblem. We describe it as follows:

When $|L_i| = l_i$ is fixed for each i, since the routing costs on all edges in A are also fixed, the problem is to find the partition minimizing $\sum_{1 \leq i \leq k} \sum_{v \in L_i} w(v, a_i)$. Such a partition can be solved in polynomial time by a straightforward reduction to an *assignment problem* (or to a maximum perfect matching problem). Since the number of all possible configurations $(l_1, l_2, \ldots, l_k)$ is $O(|V_H|^{k-1})$, the problem can be solved in polynomial time for a fixed k.

Now consider the case with positive integer weights on vertices in V_H. Let $\bar{R} = \bar{r}(A) + \bar{r}(V_H)$ and let $V_H^+ = \{v_i^j \mid \forall v_i \in V_H,\ 1 \le j \le \bar{r}(v_i)\}$. Construct a graph G^+ as follows:

- Vertex set: $V(A) \cup V_H^+$;
- Edge set: $E(A) \cup \{(i, j) \mid \forall i \in V_H^+,\ j \in V(A)\}$;
- Edge weight w_1: $w_1(v_i^j, a) = w(v_i, a)(\bar{R} - \bar{r}(v_i))/(\bar{R} - 1)$ for each $a \in V(A)$ and $w_1(e) = w(e)$ for each $e \in E(A)$;
- Vertex weight r_1: $r_1(a) = \bar{r}(a)$ for each $a \in V(A)$ and $r_1(v) = 1$ for each $v \in V_H^+$.

Let Y be the tree in $\mathrm{Star}(A, V_H^+)$ on graph G^+ such that $c(Y, r_1)$ is minimum. Also let the partition describing Y be $(L_1, L_2, \ldots, L_{\bar{k}})$.

LEMMA 9. *In Y, v_i^j and v_i^m are connected to the same internal node for any i, j, m.*

Proof. We prove the lemma by contradiction. Without loss of generality, assume $v_i^1 \in L_1$ and $v_i^2 \in L_2$. We shall show that Y is not minimal. Let Y_1 be the tree obtained by moving v_i^2 from L_2 to L_1 and let Y_2 be the tree obtained by moving v_i^1 from L_1 to L_2. Consider the cost of Y_1.

$$c(Y_1, r_1) = c(Y, r_1) - 2 \sum_{v \in V(G^+)} r_1(v) d_Y\left(v_i^2, v\right) + 2 \sum_{v \in V(G^+)} r_1(v) d_{Y_1}\left(v_i^2, v\right)$$

Since

$$\sum_{v \in V(G^+)} r_1(v) d_Y\left(v_i^2, v\right) = \sum_{v \neq v_i^2} r_1(v)\left(w_1\left(v_i^2, a_2\right) + d_Y(a_2, v)\right) = (\bar{R} - 1) w_1\left(v_i^2, a_2\right) + \sum_{v \neq v_i^2} r_1(v) d_Y(a_2, v)$$

and

$$\sum_{v \in V(G^+)} r_1(v) d_{Y_1}\left(v_i^2, v\right) = (\bar{R} - 1) w_1\left(v_i^2, a_1\right) + \sum_{v \neq v_i^2} r_1(v) d_{Y_1}(a_1, v),$$

we have

$$c(Y_1, r_1) = c(Y, r_1) - 2\left((\bar{R} - 1)w_1(v_i^2, a_2) + \sum_{v \neq v_i^2} r_1(v) d_Y(a_2, v)\right) + 2\left((\bar{R} - 1)w_1(v_i^2, a_1) + \sum_{v \neq v_i^2} r_1(v) d_{Y_1}(a_1, v)\right). \quad (9)$$

Similarly,

$$c(Y_2, r_1) = c(Y, r_1) - 2\left((\bar{R} - 1)w_1(v_i^1, a_1) + \sum_{v \neq v_i^1} r_1(v) d_Y(a_1, v)\right) + 2\left((\bar{R} - 1)w_1(v_i^1, a_2) + \sum_{v \neq v_i^1} r_1(v) d_{Y_2}(a_2, v)\right). \quad (10)$$

Recall that $w_1(v_i^1, a) = w_1(v_i^2, a)$ for any $a \in V(A)$. Summing up Eqs. (9) and (10), we obtain

$$\begin{aligned}
&(c(Y_1, r_1) + c(Y_2, r_1))/2 \\
&\quad = c(Y, r_1) + \left(\sum_{v \neq v_i^2} r_1(v) d_{Y_1}(a_1, v) - \sum_{v \neq v_i^1} r_1(v) d_Y(a_1, v)\right) \\
&\qquad + \left(\sum_{v \neq v_i^1} r_1(v) d_{Y_2}(a_2, v) - \sum_{v \neq v_i^2} r_1(v) d_Y(a_2, v)\right) \\
&\quad = c(Y, r_1) + \left(d_{Y_1}(a_1, v_i^1) - d_Y(a_1, v_i^2)\right) \\
&\qquad + \left(d_{Y_2}(a_2, v_i^2) - d_Y(a_2, v_i^1)\right) \\
&\quad = c(Y, r_1) + \left(w_1(a_1, v_i^1) - d_Y(a_1, a_2) - w_1(a_1, v_i^2)\right) \\
&\qquad + \left(w_1(a_2, v_i^2) - d_Y(a_2, a_1) - w_1(a_1, v_i^1)\right) \\
&\quad = c(Y, r_1) - 2d_Y(a_1, a_2).
\end{aligned}$$

Since it is assumed that there is no zero length edge, we have $\min\{c(Y_1, r_1), c(Y_2, r_1)\} < c(Y, r_1)$. This is a contradiction to the fact that the cost of Y is minimum. ∎

Let $\bar{L}_i = \{v_i \mid \exists v_i^j \in L_i\}$. By the above lemma, $(\bar{L}_1, \bar{L}_2, \ldots, \bar{L}_{\bar{k}})$ is a partition of V_H. Let $\bar{Y} \in \text{Star}(A, V_H)$ and described by $(\bar{L}_1, \bar{L}_2, \ldots, \bar{L}_{\bar{k}})$. The following lemma is immediate.

LEMMA 10. $c(Y, r_1) = c(\overline{Y}, \bar{r})$.

Proof. Since $r_1(a_i) = \bar{r}(a_i)$ and $\bar{r}(L_i) = \bar{r}(\overline{L}_i)$ for all i, the routing costs on any core edge are the same in the two trees. So,

$$c(Y, r_1) - c(\overline{Y}, \bar{r}) = 2(\overline{R} - 1) \sum_{1 \le i \le \bar{k}} \sum_{v \in L_i} w_1(v, a_i) - 2 \sum_{1 \le i \le \bar{k}} \sum_{v_j \in \overline{L}_i} \big(\overline{R} - \bar{r}(v_j)\big)\bar{r}(v_j) w(v_j, a_i).$$

By the definition of w_1 and $\overline{L}_i$,

$$\begin{aligned} &2(\overline{R} - 1) \sum_{1 \le i \le \bar{k}} \sum_{v \in L_i} w_1(v, a_i) \\ &\quad = 2(\overline{R} - 1) \sum_{1 \le i \le \bar{k}} \sum_{v_j \in \overline{L}_i} \sum_{1 \le m \le \bar{r}(v_j)} w_1\big(v_j^m, a_i\big) \\ &\quad = 2(\overline{R} - 1) \sum_{1 \le i \le \bar{k}} \sum_{v_j \in \overline{L}_i} \bar{r}(v_j) w(v_j, a_i)\big(\overline{R} - \bar{r}(v)\big)/(\overline{R} - 1) \\ &\quad = 2 \sum_{1 \le i \le \bar{k}} \sum_{v_j \in \overline{L}_i} \big(\overline{R} - \bar{r}(v)\big)\bar{r}(v_j) w(v_j, a_i). \end{aligned}$$

Therefore, $c(Y, r_1) - c(\overline{Y}, \bar{r}) = 0$. ∎

The proof of the next corollary is similar to that of the above lemma.

COROLLARY 11. *For any tree* $\overline{Z} \in \text{Star}(A, V_H)$ *on* $\overline{G}$, *there exists a tree* $Z \in \text{Star}(A, V_H^+)$ *on* G^+ *such that* $c(Z, r_1) = c(\overline{Z}, \bar{r})$.

By Lemma 10 and Corollary 11, we have the following corollary:

COROLLARY 12. $(\overline{L}_1, \overline{L}_2, \ldots, \overline{L}_{\bar{k}})$ *describes the minimum tree* $\overline{Y}$ *in* $\text{Star}(A, V_H)$ *on* $\overline{G}$.

Therefore, the problem of finding the desired tree in Step 2.4 is equivalent to that of finding the minimum cost tree Y in $\text{Star}(A, V_H^+)$ on G^+. Since $r_1(v) = 1$ for any vertex in $V(G^+) \setminus V(A)$, Y can be found by solving a series of assignment problems as in the unweighted case. It takes $O(\overline{R}^{k+2})$ time if we solve the $\binom{\overline{R}}{k-1}$ assignment problems individually. However, as in [7], there is an efficient algorithm for solving all of these assignment problems with total time complexity $O(\overline{R}^k)$. By using such an algorithm, we have Lemma 13 as the result of this section.

LEMMA 13. *The time complexity of Step* 2.4 *is* $O((\frac{qR}{\mu})^k)$.

APPENDIX A

In this appendix, we present the transformation algorithm and the related results. Let $G = (V, E, w)$ and $\overline{G} = (V, V \times V, h)$. Any edge (a, b) in $\overline{G}$ is called a *bad edge* if $(a, b) \notin E$ or $w(a, b) > h(a, b)$. For any bad edge $e = (a, b)$ there must exist a path $P \neq e$ such that $w(P) = h(a, b)$. Given any spanning tree T of $\overline{G}$, the algorithm can construct another spanning tree Y without any bad edge such that $c(Y, r) \leq c(T, r)$. Since Y has no bad edge, $h(e) = w(e)$ for every $e \in E(Y)$ and Y can be thought of as a spanning tree of G with the same cost.

ALGORITHM. Remove_bad

Input: a spanning tree T of $\overline{G}$
Output: a spanning tree Y of G (i.e. without any bad edge) such that $c(Y, r) \leq c(T, r)$.

Compute all pairs of the shortest paths of G.
while there exists a bad edge in T (1)
  Pick a bad edge (a, b). Root T at a.
  /* assume $\mathrm{SP}_G(a, b) = (a, x, \ldots, b)$
    and y is the father of x */
  if b is not an ancestor of x **then**
    $Y^* = T \cup (x, b) - (a, b)$
    $Y^{**} = Y^* \cup (a, x) - (x, y)$
  else
    $Y^* = T \cup (a, x) - (a, b)$
    $Y^{**} = Y^* \cup (b, x) - (x, y)$
  endif
  if $c(Y^*, r) < c(Y^{**}, r)$ **then**
    $Y = Y^*$
  else
    $Y = Y^{**}$
  endif
  $T = Y$ (2)
endwhile

We assume that the shortest paths obtained in the first step have the following property: If $\mathrm{SP}_G(a, b) = (a, x, \ldots, b)$, then $\mathrm{SP}_G(a, b) = (a, x) \cup \mathrm{SP}_G(x, b)$. This assumption is feasible since almost all popular algorithms for all of the pairs of the shortest paths output such a solution.

CLAIM 14. *The loop* (1) *is executed at most* $O(n^2)$ *times*.

Proof. For each bad edge $e = (a, b)$, let $l(e)$ be the number of edges in $\mathrm{SP}_G(a, b)$ and $f(T) = \sum_{\text{bad } e} l(e)$. Since $l(e) \leq n - 1$, $f(T) < n^2$. Since (a, x) is not a bad edge, it is easy to check that $f(T)$ decreases by at least 1 at each loop iteration. ∎

CLAIM 15. *Before instruction* (2) *is executed*, $c(Y, r) \leq c(T, r)$.

Proof. For any node v, let S_v be the set of vertices in the subtree rooted at v.

Case 1. $x \in S_a - S_b$. If $c(Y^*, r) \leq c(T, r)$, the result follows. Otherwise, let $U_1 = S_a - S_b$ and $U_2 = S_a - S_b - S_x$. Since the distance does not change for any two vertices both in U_1 (or both in S_b), we have

$$c(T, r) < c(Y^*, r)$$

$$\sum_{i \in U_1} \sum_{j \in S_b} r(i) r(j) d_T(i, j) < \sum_{i \in U_1} \sum_{j \in S_b} r(i) r(j) d_{Y^*}(i, j).$$

Since $d_T(i, j) = d_T(i, a) + h(a, b) + d_T(b, j)$ and $d_{Y^*}(i, j) = d_T(i, x) + h(x, b) + d_T(b, j)$ $\forall i \in U_1$, $j \in S_b$, we have

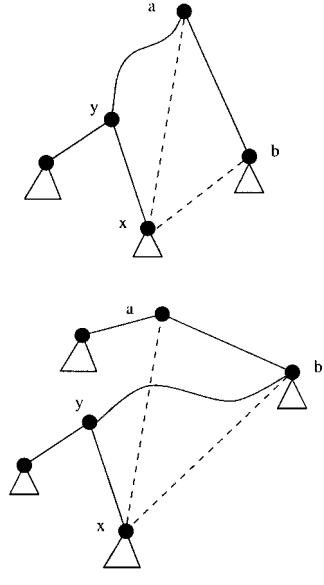

FIG. 1. Removing the bad edge (a, b). Case 1 (top) and Case 2 (bottom) are presented.

$$\sum_{i\in U_1}\sum_{j\in S_b} r(i)r(j)\big(d_T(i,a)+h(a,b)+d_T(b,j)\big)$$

$$< \sum_{i\in U_1}\sum_{j\in S_b} r(i)r(j)\big(d_T(i,x)+h(x,b)+d_T(b,j)\big)$$

$$\Rightarrow r(S_b)\sum_{i\in U_1} r(i)d_T(i,a)+r(U_1)r(S_b)h(a,b)$$

$$< r(S_b)\sum_{i\in U_1} r(i)d_T(i,x)+r(U_1)r(S_b)h(x,b)$$

$$\Rightarrow \sum_{i\in U_1} r(i)d_T(i,a)+r(U_1)h(a,b)$$

$$< \sum_{i\in U_1} r(i)d_T(i,x)+r(U_1)h(x,b)$$

$$\Rightarrow \sum_{i\in U_1} r(i)\big(d_T(i,a)-d_T(i,x)\big) < -r(U_1)h(a,x).$$

Note that $r(S_b) > 0$ since the inequality holds. Then,

$$(c(Y^{**},r)-c(T,r))/2 = \sum_{i\in U_2}\sum_{j\in S_x} r(i)r(j)\big(d_{Y^{**}}(i,j)-d_T(i,j)\big) + \sum_{i\in U_1}\sum_{j\in S_b} r(i)r(j)\big(d_{Y^{**}}(i,j)-d_T(i,j)\big).$$

Since $d(Y^{**},i,j)\le d_T(i,j)$ for $i\in U_1$ and $j\in S_b$, the second term is not positive. Since $d_T(i,j)=d_T(i,x)+d_T(x,j)$ and $d(Y^{**},i,j)=d_T(i,a)+h(a,x)+d_T(x,j)$ $\forall i\in U_2$, $j\in S_x$, we have

$$\begin{aligned}
&(c(Y^{**},r)-c(T,r))/2\\
&\quad\le \sum_{i\in U_2}\sum_{j\in S_x} r(i)r(j)\big(d_T(i,a)+h(a,x)-d_T(i,x)\big)\\
&\quad= r(S_x)\sum_{i\in U_2} r(i)\big(d_T(i,a)+h(a,x)-d_T(i,x)\big)\\
&\quad= r(S_x)\sum_{i\in U_2} r(i)\big(d_T(i,a)-d_T(i,x)\big)+r(U_2)r(S_x)h(a,x)\\
&\quad\le r(S_x)\sum_{i\in U_1} r(i)\big(d_T(i,a)-d_T(i,x)\big)+r(U_2)r(S_x)h(a,x)\\
&\quad< -r(U_1)r(S_x)h(a,x)+r(U_2)r(S_x)h(a,x)\\
&\quad\le 0.
\end{aligned}$$

Note that $\sum_{i \in U_2} r(i)(d_T(i, a) - d_T(i, x)) \leq \sum_{i \in U_1} r(i)(d_T(i, a) - d_T(i, x))$ since $U_1 - U_2 = S_x$ and $d_T(i, a) > d_T(ix)\ \forall i \in S_x$.
So, $c(Y^{**}, r) < c(T, r)$.

Case 2. $x \in S_b$. Root the tree at b and exchange the label of a and b. We can find that this case is the same as Case 1. ∎

Lemma 2 in Section 4 comes from the two claims.

APPENDIX B

In this appendix, we show the existence of the balanced k-star which is a $(k + 3)/(k + 1)$-approximation of the PROCT problem on a metric graph. Recall that G is a metric graph and that R denotes the total vertex weight $r(G)$.

DEFINITION 11. If T is a tree and $S \subset T$, $w_S(T, i, j) = w(SP_T(i, j) \cap S)$ for $i, j \in V(T)$.

DEFINITION 12. Let T be a spanning tree of G and S a connected subgraph of T. Deleting the edges in $E(S)$ from T will result in several subtrees. For a vertex $i \in V(S)$, we use $VB(T, S, i)$ to denote the vertex set of the subtree containing i.

LEMMA 16. *Let S be a minimal δ-separator of T. If i is a leaf of S,* $r(VB(T, S, i)) > \delta R$.

Proof. If S contains only one vertex, the result is trivial. Otherwise, if $r(VB(T, S, i)) > \delta R$, deleting i from S we still get a δ-separator. This is a contradiction to S being minimal.

DEFINITION 13. Let T be a tree and $P = SP_T(i, j)$ in which $r(VB(T, P, i)) \geq r(VB(T, P, j))$. We define

$$P^a = r(VB(T, P, i)),$$
$$P^b = r(VB(T, P, j)), \text{ and}$$
$$P^c = R - r(VB(T, P, i)) - r(VB(T, P, j)).$$

Assume $P = (i, m_1, m_2, \ldots, m_{h,j})$. Define $Q(P) = \sum_{1 \leq x \leq h} r(VB(T, P, m_x)) \times d_T(m_x, i)$.

DEFINITION 14. For a spanning tree T of G, and $0 < \delta \leq 0.5$, a δ-path of T is a path P on T such that $P^c \leq \delta R/2$.

DEFINITION 15. Let $0 < \delta \leq 0.5$. A δ-spine $Y = \{P_1, P_2, \ldots, P_h\}$ of T is a set of pairwise edge-disjoint δ-paths in T such that $S = \bigcup_{1 \leq i \leq h} P_i$ is a

minimal δ-separator of T. Furthermore, for any pair of distinct paths P_i and P_j in the spine, we require that either they do not intersect or, if they do, that the intersection point is an endpoint of both paths. We define the *cut and leaf set* $\mathrm{CAL}(Y)$ of a δ-spine Y to be the set of the endpoints of the paths in Y. In the case that Y is empty, $\mathrm{CAL}(Y)$ contains the vertex which itself is a minimal δ-separator.

LEMMA 17. *Let Y be a δ-spine of a spanning tree T of G and $S = \bigcup_{P \in Y} P$ be a minimal δ-separator of T. Then*

$$c(T,r)/2 \geq (1-\delta)R \sum_{v \in V} r(v) d_T(v,S) + \sum_{P \in Y} \left(P^b(P^a + P^c)w(P) + (P^a - P^b)Q(P) \right).$$

Proof. We have

$$\begin{aligned} c(T,r)/2 &= \sum_{e \in T} e^a e^b w(e) \\ &\geq \sum_{e \in T \setminus S} (1-\delta) R e^b w(e) + \sum_{e \in S} e^a e^b w(e) \\ &\geq (1-\delta)R \sum_{v \in V} r(v) d_T(v,S) + \sum_{P \in Y} \sum_{e \in P} e^a e^b w(e). \end{aligned}$$

For the second term, assume $P = (m_0, m_1, m_2, \ldots, m_h)$ in which $r(VB(T,P,m_0)) \geq r(VB(T,P,m_h))$. Let $r(VB(T,P,m_i)) = n_i$ for $1 \leq i \leq h-1$ and $e_i = (m_{i-1}, m_i)$ for $1 \leq i \leq h$.

$$\begin{aligned} \sum_{e \in P} e^a e^b w(e) &= \sum_{i=1}^{h} \left(P^a + P^c - \sum_{j=i}^{h-1} n_j \right) \left(P^b + \sum_{j=i}^{h-1} n_j \right) w(e_i) \\ &= \sum_{i=1}^{h} P^b(P^a + P^e) w(e_i) \\ &\quad + (P^a - P^b) \sum_{i=1}^{h} \sum_{j=i}^{h-1} n_j w(e_i) \\ &\quad + \sum_{i=1}^{h} \left(\sum_{j=i}^{h-1} n_j \right) \left(P^c - \sum_{j=i}^{h-1} n_j \right) w(e_i) \\ &\geq P^b(P^a + P^c) w(P) \\ &\quad + (P^a - P^b) \sum_{j=1}^{h-1} n_j \left(\sum_{i=1}^{j} w(e_i) \right) \\ &= P^b(P^a + P^c) w(P) + (P^a - P^b) Q(P). \end{aligned}$$

∎

LEMMA 18. *For any constant* $0 < \delta \leq 0.5$ *and spanning tree* T *of* G, *there exists a* δ*-spine* Y *of* T *such that* $|\mathrm{CAL}(Y)| \leq \lceil 2/\delta \rceil - 3$.

Proof. Let S be a minimal δ-separator of T. S is a tree. Let $U_1 = \{v \mid v$ is a leaf in $S\}$, $U_2 = \{v \mid v$ has more than two neighbors in $S\}$, and $U = U_1 \cup U_2$. Let $h = |U_1|$. Clearly, $|U| \leq 2h - 2$. For any $v_1, v_2 \in U$, call v_1 and v_2 neighbors in U if $v_3 \notin \mathrm{SP}_T(v_1, v_2)$ for all $v_3 \in U - \{v_1, v_2\}$. Let $Y_1 = \{\mathrm{SP}_T(v_1, v_2) \mid v_1, v_2$ are neighbors in $U\}$. For any $P \in Y_1$, if $P^c > \delta R/2$, P is called a heavy path. It is easy to check that Y_1 satisfies the requirements of a δ-spine except that there may exist some heavy paths. For any heavy path P, we can divide it into some non-heavy paths by no more than $\lceil 2P^c/(\delta R) \rceil - 1$ vertices on P. Since there are at least δR weights hanged at each leaf,

$$\sum_{P \in Y_1} P^c < R - h\delta R.$$

Assume U_3 to be the minimal vertex set for cutting the heavy paths to result in a δ-spine Y of T. We have

$$|U_3| \leq \lceil 2(R - h\delta R)/(\delta R) \rceil - 1 = \lceil 2/\delta \rceil - 2h - 1.$$

So, $|\mathrm{CAL}(Y)| = |U| + |U_3| \leq \lceil 2/\delta \rceil - 3$. ∎

LEMMA 19. *For any constant* $0 < \delta \leq 0.5$, *there exists a balanced* $(\lceil 2/\delta \rceil - 3)$*-star* X *such that* $c(X, r) \leq (1/(1 - \delta))c(\mathrm{PROCT}(G, r), r)$.

Proof. Let $T = \mathrm{PROCT}(G, r) = (V, E, w)$. Also, let $Y = \{P_i \mid 1 \leq i \leq h\}$ be a δ-spine of T in which $|\mathrm{CAL}(Y)| \leq \lceil 2/\delta \rceil - 3$, and let $S = \bigcup_{P \in Y} P$ be a minimal δ-separator of T. Assume $P_i = \mathrm{SP}_T(u_i, v_i)$ and $r(VB(T, P_i, u_i)) \geq r(VB(T, P_i, v_i))$. Construct a subgraph $M \subset G$ with vertex set $\mathrm{CAL}(Y)$ and edge set $E_m = \{(u_i, v_i) \mid 1 \leq i \leq h\}$. Trivially, M is a tree. Let $f(i)$ be an indicator variable such that if $(P_i^a - P_i^b)P_i^c w(P_i) - R(2Q(P_i) - P_i^c w(P_i)) \geq 0$ then $f(i) = 1$; else $f(i) = 0$. We construct a spanning tree X of G where the edge set E_x is determined by the following rules:

1. $M \subset X$.

2. If $q \in VB(T, S, m)$ then $(q, m) \in E_x$, for any $m \in \{u_i, v_i \mid 1 \leq i \leq h\}$.

3. For the vertex set $V_i = V - VB(T, P_i, u_i) - VB(T, P_i, v_i)$, if $f(i) = 1$ then $\{(q, u_i) \mid q \in V_i\} \subset E_x$; else $\{(q, v_i) \mid q \in V_i\} \subset E_x$. That is, the vertices in V_i are either all connected to u_i or all connected to v_i.

It is easy to see that X is a balanced $(\lceil 2/\delta \rceil - 3)$-star. Now let us compute the cost:

$$\begin{aligned} c(X,r)/2 &= \sum_{e \in E_x} e^a e^b w(e) \\ &= \sum_{e \in E_m} e^a e^b w(e) + \sum_{e \in E_x - E_m} e^a e^b w(e) \\ &\le \sum_{e \in E_m} e^a e^b w(e) + R \sum_{v \in V} r(v) d_X(v, M). \end{aligned}$$

First, for any $e = (u_i, v_i) \in E_m$,

$$\begin{aligned} e^a e^b w(e) &\le \big(P_i^a + f(i) P_i^c\big)\big(P_i^b + (1 - f(i)) P_i^c\big) w(P_i) \\ &= P_i^a P_i^b w(P_i) + \big(f(i) P_i^b + (1 - f(i)) P_i^a\big) P_i^c w(P_i). \end{aligned}$$

Second, from the triangle inequality,

$$\begin{aligned} &\sum_{v \in V} r(v) d_X(v, M) \\ &\quad \le \sum_{v \in V} r(v) d_T(v, S) \\ &\qquad + \sum_{i=1}^{h} \sum_{v \in V_i} r(v)\big(f(i) w_S(T, v, u_i) + (1 - f(i)) w_S(T, v, v_i)\big) \\ &\quad = \sum_{v \in V} r(v) d_T(v, S) \\ &\qquad + \sum_{i=1}^{h} \big(f(i) Q(P_i) + (1 - f(i))\big(P_i^c w(P_i) - Q(P_i)\big)\big). \end{aligned}$$

So,

$$\begin{aligned} c(X,r)/2 \le \sum_{i=1}^{h} \big(P_i^a P_i^b w(P_i)\big) + R \sum_{v \in V} r(v) d_T(v, S) \\ + \sum_{i=1}^{h} \min\big\{P_i^b P_i^c w(P_i) + R Q(P_i), \\ P_i^a P_i^c w(P_i) + R\big(P_i^c w(P_i) - Q(P_i)\big)\big\} \end{aligned}$$

Since the minimum of two numbers is not larger than their weighted mean, we have

$$\begin{aligned}\min\{&P_i^bP_i^cw(P_i)+RQ(P_i),P_i^aP_i^cw(P_i)+R(P_i^cw(P_i)-Q(P_i))\}\\&\le (P_i^bP_i^cw(P_i)+RQ(P_i))\frac{P_i^a}{P_i^a+P_i^b}\\&\quad+(P_i^aP_i^cw(P_i)+R(P_i^cw(P_i)-q(P_i)))\frac{P_i^b}{P_i^a+P_i^b}.\end{aligned}$$

Then,

$$\begin{aligned}c(X,r)/2&\le\sum_{i=1}^{h}(P_i^aP_i^bw(P_i))+R\sum_{v\in V}r(v)d_T(v,S)\\&\quad+\sum_{i=1}^{h}\frac{(2P_i^aP_i^bP_i^c+RP_i^bP_i^c)w(P_i)}{P_i^a+P_i^b}\\&\quad+\sum_{i=1}^{h}\frac{(P_i^a-P_i^b)RQ(P_i)}{P_i^a+P_i^b}\\&=R\sum_{v\in V}r(v)d_T(v,S)\\&\quad+\sum_{i=1}^{h}\frac{w(P_i)}{P_i^a+P_i^b}((P_i^aP_i^b+P_i^bP_i^c)R+P_i^aP_i^bP_i^c)\\&\quad+\sum_{i=1}^{h}\frac{(P_i^a-P_i^b)RQ(P_i)}{P_i^a+P_i^b}.\end{aligned}$$

By Lemma 17,

$$c(X,r)\le c(T,r)\max_{1\le i\le h}\left\{\frac{1}{1-\delta},\frac{R}{P_i^a+P_i^b}+\frac{P_i^aP_i^c}{(P_i^a+P_i^b)(P_i^a+P_i^c)}\right\}.$$

Since $P_i^c\le\delta R/2$,

$$\begin{aligned}&\frac{R}{P_i^a+P_i^b}+\frac{P_i^aP_i^c}{(P_i^a+P_i^b)(P_i^a+P_i^c)}\\&\quad\le\frac{R}{P_i^a+P_i^b}+\frac{P_i^c}{P_i^a+P_i^b}\\&\quad=\frac{R+P_i^c}{R-P_i^c}\le\frac{2+\delta}{2-\delta}\le\frac{1}{1-\delta}.\end{aligned}$$

∎

ACKNOWLEDGMENT

We thank the anonymous referees for their careful reading and many useful comments.

REFERENCES

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows—Theory, Algorithms, and Applications," Prentice-Hall, Englewood Cliffs, NJ, 1993.
2. M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco, 1979.
3. T. C. Hu, Optimum communication spanning trees, *SIAM J. Comput.*, **3** (1974), 188–195.
4. E. L. Lawler, Fast approximation algorithms for knapsack problems, *Math. Oper. Res.* **4**, No. 4 (1979), 339–356.
5. R. Wong, Worst-case analysis of network design problem heuristics, *SIAM J. Algebraic Discrete Math.* **1** (1980), 51–63.
6. B. Y. Wu, K. M. Chao, and C. Y. Tang, Approximation algorithms for the shortest total path length spanning tree problem, *Discrete Appl. Math.*, accepted for publication.
7. B. Y. Wu, G. Lancia, V. Bafna, K. M. Chao, R. Ravi, and C. Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees, *SIAM J. Comput.*, accepted for publication. [A preliminary version of this paper appears in the "Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'98)," pp. 21–32, ACM, New York, 1998].
8. B. Y. Wu, K. M. Chao, and C. Y. Tang, Approximation algorithms for some optimum communication spanning tree problems, *Discrete Appl. Math.* **102** (2000), 245–266. [a partial result of this paper appeared in the "Proceedings of the Ninth Annual International Symposium on Algorithm and Computation (ISAAC'98)," Lecture Notes in Computer Science, Vol. 1533, pp. 407–416, Springer-Verlag, New York/Berlin, 1998].
9. B. Y. Wu, K. M. Chao, and C. Y. Tang, Constructing light spanning trees with small routing cost, *in* "Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS'99)," Lecture Notes in Computer Science, Vol. 1563, pp. 334–344, Springer-Verlag, New York/Berlin, 1999.