ACADEMIC PRESS

Journal of Algorithms 44 (2002) 359–378

Journal of Algorithms

www.academicpress.com

A polynomial time approximation scheme for the two-source minimum routing cost spanning trees

Bang Ye Wu

Department of Computer Science and Information Engineering, Shu-Te University, YenChau, KaoShiung, Taiwan 824, ROC

Received 3 August 2001

Abstract

Let G be an undirected graph with nonnegative edge lengths. Given two vertices as sources and all vertices as destinations, we investigated the problem how to construct a spanning tree of G such that the sum of distances from sources to destinations is minimum. In the paper, we show the NP-hardness of the problem and present a polynomial time approximation scheme. For any $\varepsilon > 0$, the approximation scheme finds a $(1+\varepsilon)$-approximation solution in $O(n^{\lceil 1/\varepsilon+1\rceil})$ time. We also generalize the approximation algorithm to the weighted case for distances that form a metric space.

Keywords: NP-hard; Approximation algorithms; PTAS; Spanning trees

1. Introduction

Let $G = (V, E, w)$ be an undirected graph with nonnegative edge length function w. Given k vertices as sources and all vertices as destinations, the *k-source minimum routing cost spanning tree* (k-MRCT) problem is to construct a spanning tree T of G such that the sum of distances from sources to destinations is minimum. That is, we want to find a spanning tree T minimizing

E-mail address: bangye@mail.stu.edu.tw.

0196-6774/02/$ – see front matter
PII: S0196-6774(02)00205-5

$\sum_{s\in S}\sum_{v\in V} d_T(s,v)$, where S is the set of sources and $d_T(s,v)$ is the distance between s and v on T.

If there is only one source, the problem is reduced to the *shortest-path tree* problem, and it is always possible to find a spanning tree such that the path between the source and each vertex is a shortest path on the given graph. The shortest-path tree problem has been well studied and efficient algorithms were developed. For example, Dijkstra's algorithm [2] finds a shortest path tree for a graph with nonnegative weights in $O(|V|^2)$ time. The time complexity can be improved to $O(|E|+|V|\log|V|)$ by using Fibonacci heaps, and other algorithms with different considerations can be found in [1]. For undirected graph with positive integer weights, the most efficient algorithm runs in $O(|E|)$ time [7].

For the other extreme case that all vertices are sources, the problem is reduced to the *minimum routing cost spanning tree* (MRCT) problem (also called the *shortest total path length spanning tree* problem), and is therefore NP-hard [3,5]. The first constant ratio approximation algorithm for the MRCT appeared in [8]. It was shown that there is a shortest path tree which is a 2-approximation of the MRCT. The approximation ratio was improved to $(\frac{4}{3}+\varepsilon)$ for any fixed $\varepsilon>0$ [9], and then further improved to a *polynomial time approximation scheme* (PTAS) [10].

The *optimum communication spanning tree* (OCT) problem [4] is a more general version of the MRCT problem. In the OCT problem, in addition to the edge length, we are also given the requirement for each pair of vertices, and the goal is to minimize the sum of the distances multiplied by the requirements, over all pairs of vertices. In [11], two vertex-weighted generalizations of the MRCT problem were studied; one is the *optimal product-requirement communication spanning tree* (PROCT) problem and the other is the *optimal sum-requirement communication spanning tree* (SROCT) problem. In the PROCT problem, the requirement between a pair of vertices is assumed to be the product of the given weights of the two vertices; while, in the SROCT problem, the requirement is the sum of the vertex weights. Both PROCT and SROCT problems are special cases of the OCT problem. A 1.577-approximation algorithm for the PROCT problem and a 2-approximation algorithm for the SROCT problem were shown [11]. Furthermore the PROCT problem was shown to admit a PTAS [12].

The k-source MRCT problem is a special case of the SROCT problem, in which the vertex weight of each source is one and the weights of all the other vertices are zeros. The relationship of the four problems is shown in Fig. 1.

The k-MRCT problem for general k is obviously NP-hard since it contains the MRCT problem as a special case. But the previous results do not tell us the complexity of the k-MRCT problem for a fixed constant k. Since the k-MRCT problem is a special case of the SROCT problem, the 2-approximation algorithm for the SROCT problem [11] ensures the same approximation ratio of the k-MRCT. In this paper, we show that the k-MRCT problem is NP-hard even

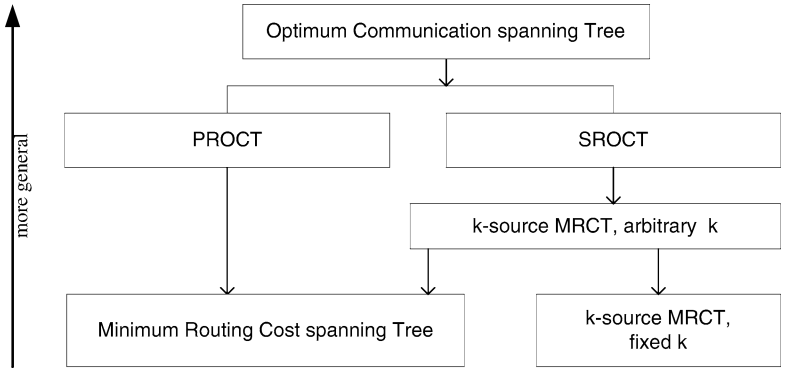

Fig. 1. The relationship of the OCT, PROCT, SROCT, MRCT, and k-MRCT problems.

for $k = 2$. We present a PTAS for the 2-MRCT problem. For any $\varepsilon > 0$, the PTAS finds a solution with approximation ratio $1 + \varepsilon$ in $O(n^{\lceil 1/\varepsilon + 1 \rceil})$ time.

We also consider a weighted version of the 2-MRCT problem. In the weighted 2-MRCT problem, there is a positive weight of each of the two sources and we want to minimize the weighted total distance from sources to all vertices, i.e., $\sum_{v \in V}(\lambda_1 d_T(s_1, v) + \lambda_2 d_T(s_2, v))$, where s_1, s_2 are source vertices and λ_1, λ_2 are the weights of the two sources. In this paper, we present a 2-approximation algorithm for general graphs and a PTAS for metric graphs. A metric graph is a complete graph with edge weights satisfying the triangle inequality.

The remaining sections are organized as follows: In Section 2, some definitions and notations are given. The NP-hardness of the 2-MRCT problem is shown in Section 3, and the PTAS for the 2-MRCT is presented in Section 4. Finally, the weighted case is in Section 5, and concluding remarks are given in Section 6.

2. Preliminaries

In this paper, a graph is a simple, connected and undirected graph. By $G = (V, E, w)$, we denote a graph G with vertex set V, edge set E, and edge length function w. The edge length function is assumed to be nonnegative. For any graph G, $V(G)$ denotes its vertex set and $E(G)$ denotes its edge set. Let w be an edge length function on a graph G. For a subgraph H of G, we define $w(H) = w(E(H)) = \sum_{e \in E(H)} w(e)$. We shall also use n to denote $|V(G)|$.

Definition 2.1. Let $G = (V, E, w)$ be a graph. For $u, v \in V$, $SP_G(u, v)$ denotes a shortest path between u and v on G. The shortest path length is denoted by $d_G(u, v) = w(SP_G(u, v))$.

Definition 2.2. Let H be a subgraph of G. For a vertex $v \in V(G)$, we use $d_G(v, H)$ to denote the shortest distance from v to H, i.e., $d_G(v, H) = \min_{u \in V(H)} d_G(v, u)$. The definition also includes the case that H is a vertex set but no edge.

We now define the k-source routing cost of a tree.

Definition 2.3. For a tree T and $S \subset V(T)$, the *routing cost* of T with source set S is defined by $c(T, S) = \sum_{s \in S} \sum_{v \in V(T)} d_T(s, v)$. When there are only two sources s_1 and s_2, we also use $c(T, s_1, s_2)$ to denote the routing cost.

Lemma 2.1. *Let T be a spanning tree of $G = (V, E, w)$ and P be the path between s_1 and s_2 on T. $c(T, s_1, s_2) = n \times w(P) + 2\sum_{v \in V} d_T(v, P)$, in which $n = |V|$.*

Proof. For any $v \in V$, $d_T(v, s_1) + d_T(v, s_2) = w(P) + 2d_T(v, P)$. Summing over all vertices in V, the result is obtained. □

Once a path P between the two sources has been chosen, by Lemma 2.1, it is obvious that the best way to extend P to a spanning tree is to add the shortest path forest using the vertices of P as multiple roots, i.e., the distance from every other vertex to the path is made as small as possible. A shortest path forest can be constructed by an algorithm similar to the shortest path tree algorithm. First we create a dummy node and the multiple roots are connected to the dummy node by edges of zero weight. Then a shortest path tree from the dummy node is constructed, and the shortest path forest is obtained by removing the dummy node and dummy edges. The time complexity is the same as the shortest path tree algorithm. However, the most time-efficient algorithm for the shortest path tree depends on the graph. In the remaining paragraphs, the time complexity for finding a shortest path tree/forest of a graph G will be denoted by $f_{SP}(G)$. In this paper, we consider only undirected graphs with nonnegatice edge weights. For general graphs, $f_{SP}(G)$ is $O(|E| + |V| \log |V|)$, and it is $O(|E|)$ for graphs with integer weights [7]. However, the time complexity is $O(n^2)$ for dense graphs, i.e., the number of edges is $\theta(n^2)$.

3. The computational complexity

In this section, we show the NP-hardness of the 2-MRCT problem by reducing the *exact cover by* 3-*sets* (X3C) problem to it.

Definition 3.1. Given a set X with $|X| = 3q$ and a collection C of 3-element subsets of X, the X3C problem asks if there is a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'.

Let $X = \{x_i\colon\ 1 \leqslant i \leqslant 3q\}$ and $C = \{Y_i\colon\ 1 \leqslant i \leqslant m\}$ be an instance of the X3C problem, in which each Y_i is a 3-element subset of X. To reduce the X3C problem to the 2-MRCT problem, we construct a complete graph $G = (V, E, w)$ as an instance of the 2-MRCT problem. The graph G consists of the following subgraphs as illustrated in Fig. 2.

- For $1 \leqslant i \leqslant q$, G_i is a complete graph with vertex set $\{v_i^j\colon\ 1 \leqslant j \leqslant m\}$, in which v_i^j corresponds to Y_j in C. Every edge in G_i has weight one.
- H is a complete graph with vertex set $\{u_i\colon\ 1 \leqslant i \leqslant 3q\}$, in which each u_i corresponds to one element of X. Every edge in H has weight one.
- G_0 contains only one vertex s_1 and G_{q+1} contains only one vertex s_2.

The weights of edges between the subgraphs are defined as follows:

- For $u \in V(G_i)$, $v \in V(G_j)$ and $i \neq j$, $w(u, v) = |i - j|$.
- For $u_i \in V(H)$ and $v_k^j \in V(G_k)$, $1 \leqslant k \leqslant q$, $w(u_i, v_k^j) = L$ if $x_i \in Y_j$; and $w(u_i, v_k^j) = L + 1$ otherwise, in which $L \geqslant q + 1$ is an integer.
- $w(u_i, s_1) = w(u_i, s_2) = L + 1$ for $u_i \in V(H)$.

Obviously the edge weights satisfy the triangle inequality, and G is a metric graph. Let $n = |V|$ be the number of vertices of G. We have $n = qm + 3q + 2 = q(m + 3) + 2$. In the next three propositions, we shall show that there is an exact cover of the X3C problem if and only if there is a spanning tree T of G such that $c(T, s_1, s_2) = n(q + 1) + 2(m - 1)q + 6qL$.

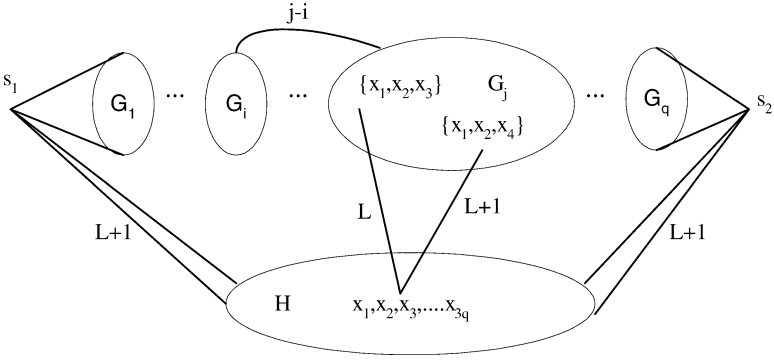

Fig. 2. The transformation of an instance of the X3C problem to an instance of the 2-MRCT problem.

Proposition 3.1. *An exact cover of the X3C problem implies a spanning tree T of G with $c(T, s_1, s_2) = n(q+1) + 2(m-1)q + 6qL$.*

Proof. Suppose that there is an exact cover of the X3C problem. Without loss of generality, let $\{Y_i\colon\ 1 \leqslant i \leqslant q\} \subset C$ be an exact cover of X. Let $P = (s_1, v_1^1, v_2^2, \ldots, v_q^q, s_2)$. Construct a spanning tree T such that T contains P and all the other vertices are connected to P by the shortest edge to P. Each vertex in $V(G_i) \setminus \{v_i^i\}$ is connected to v_i^i by an edge of weight one, for $1 \leqslant i \leqslant q$. Since $\{Y_i\colon\ 1 \leqslant i \leqslant q\}$ is an exact cover of X, each vertex in $V(H)$ is connected to some vertex of P by an edge of weight L. By Lemma 2.1,

$$\begin{aligned} c(T, s_1, s_2) &= n \times w(P) + 2\sum_{v \in V} d_T(v, P) \\ &= n(q+1) + 2(m-1)q + 6qL. \quad \square \end{aligned}$$

Proposition 3.2. *Let T be a spanning tree of G with $c(T, s_1, s_2) = n(q+1) + 2(m-1)q + 6qL$. Then, the path P between s_1 and s_2 has exactly q vertices $p_1, p_2, \ldots, p_q$ with $p_i \in V(G_i)$.*

Proof. First we shall show that P is a shortest path and $w(P) = q+1$. Consider the following two cases.

Case 1. P contains a vertex in $V(H)$. By the definition of G, $w(P) \geqslant 2L+2$. By Lemma 2.1, $c(T, s_1, s_2) = n \times w(P) + 2\sum_{v \in V} d_T(v, P)$.

$$\begin{aligned} c(T, s_1, s_2) &\geqslant n(2L+2) \geqslant n(q+1+L+2) \\ &= n(q+1) + n(L+2) \\ &= n(q+1) + 2q(m+3) + 4 + \big(q(m+3)+2\big)L \\ &> n(q+1) + 2(m-1)q + 6qL, \end{aligned}$$

when $m \geqslant 3$.

Case 2. Assume that P contains $q+k$ vertices of $\bigcup_{1 \leqslant i \leqslant q} V(G_i)$ for any $k > 0$. Since all vertices in $V(H)$ are not on the path, we have

$$\begin{aligned} &c(T, s_1, s_2) \\ &\quad \geqslant n \times w(P) + 2\sum_{v \in V} d_T(v, P) \\ &\quad \geqslant n(q+k+1) + 2\sum_{1 \leqslant i \leqslant q} \sum_{v \in V(G_i)} d_T(v, P) + 2\sum_{v \in V(H)} d_T(v, P) \\ &\quad \geqslant n(q+k+1) + 2\big(qm - (q+k)\big) + 2(3q)L \\ &\quad = n(q+1) + kn + 2q(m-1) - 2k + 6qL \end{aligned}$$

$$> n(q+1) + 2(m-1)q + 6qL.$$

For both of the two cases, we have shown that $c(T, s_1, s_2) > n(q+1) + 2(m-1)q + 6qL$. Therefore $w(P) = q+1$. By Lemma 2.1,

$$\begin{aligned} c(T, s_1, s_2) &= n \times w(P) + 2\sum_{v \in V} d_T(v, P) \\ &= n(q+1) + 2\sum_{1 \leqslant i \leqslant q} \sum_{v \in V(G_i)} d_T(v, P) + 2\sum_{v \in V(H)} d_T(v, P) \\ &\geqslant n(q+1) + 2(m-1)q + 6qL. \end{aligned}$$

The equality holds if and only if $\sum_{1 \leqslant i \leqslant q} \sum_{v \in V(G_i)} d_T(v, P) = (m-1)q$ and $\sum_{v \in V(H)} d_T(v, P) = 3qL$. Let $P = (p_0 = s_1, p_1, p_2, \ldots, s_2)$. The second equation implies that each vertex in $V(H)$ is connected to some vertex of P by an edge of weight L. That is, for each $u_i \in V(H)$, there exists $(u_i, p_j) \in E(T)$ and $w(u_i, p_j) = L$. It also implies that the path contains at least q vertices in addition to the two sources. Then, by $w(P) = q+1$, we have that $p_i \in V(G_i)$ for $1 \leqslant i \leqslant q$. □

Proposition 3.3. *Let T be a spanning tree of G with $c(T, s_1, s_2) = n(q+1) + 2(m-1)q + 6qL$. Then, there exists an exact cover of the X3C problem.*

Proof. By Proposition 3.2, the path between the two sources is $P = (p_0 = s_1, p_1, p_2, \ldots, s_2)$ with $p_i \in V(G_i)$ for $1 \leqslant i \leqslant q$. Consequently there are exact 3 vertices of $V(H)$ connected to each p_j by edges of weight L. Without loss of generality, assume that $p_i = v_i^i$ for $1 \leqslant i \leqslant q$. By the definition of G, for each $x_i \in X$, there exists Y_j such that $1 \leqslant j \leqslant q$ and $x_i \in Y_j$, which implies $\{Y_1, Y_2, \ldots, Y_q\}$ is an exact cover of X. □

The next theorem shows the complexity of the problem.

Theorem 3.1. *The two-source MRCT problem is NP-hard even for metric inputs.*

Proof. By Propositions 3.1 and 3.3, we have shown that the X3C problem can be reduced to the 2-MRCT problem in polynomial time. Since the X3C problem is NP-complete [3,6], the 2-MRCT problem is NP-hard even with metric inputs. □

4. A PTAS for 2-MRCT

Before showing our PTAS, we give a simple 2-approximation algorithm, and then generalize the idea to the PTAS.

Algorithm 1.

Input: A graph $G = (V, E, w)$ and $s_1, s_2 \in V$.
Output: A spanning tree T of G.
1. Find a shortest path P between s_1 and s_2 on G.
2. Find the shortest path forest with multiple roots in $V(P)$.
3. Let T be the union of the forest and P.

Lemma 4.1. *The Algorithm* A1 *finds a* 2*-approximation of the* 2*-MRCT of a graph* G *in* $f_{SP}(G)$ *time.*

Proof. First we show a lower bound of the optimum. Let Y be the optimal tree of the 2-MRCT problem with input G and s_1, s_2. Since $d_Y(v, s_i) \geqslant d_G(v, s_i)$ for any vertex v and $i \in \{1, 2\}$,

$$c(Y, s_1, s_2) \geqslant \sum_{v \in V} \big(d_G(v, s_1) + d_G(v, s_2)\big). \tag{1}$$

By Lemma 2.1, we have

$$c(Y, s_1, s_2) \geqslant n d_G(s_1, s_2). \tag{2}$$

By Eqs. (1) and (2), we have

$$c(Y, s_1, s_2) \geqslant \frac{1}{2} \sum_{v} \big(d_G(v, s_1) + d_G(v, s_2)\big) + \frac{n}{2} d_G(s_1, s_2). \tag{3}$$

Let T be the tree constructed by Algorithm A1 and P be the shortest between s_1 and s_2. Since each vertex is connected to P by a shortest path to any of the vertices of P at Step 2, for any vertex v,

$$d_T(v, P) \leqslant \min\big\{d_G(v, s_1), d_G(v, s_2)\big\} \leqslant \frac{1}{2}\big(d_G(v, s_1) + d_G(v, s_2)\big).$$

By Lemma 2.1,

$$\begin{aligned} c(T, s_1, s_2) &= n \times w(P) + 2 \sum_{v \in V} d_T(v, P) \\ &\leqslant n d_G(s_1, s_2) + \sum_{v} \big(d_G(v, s_1) + d_G(v, s_2)\big). \end{aligned}$$

Comparing with Eq. (3), we have $c(T, s_1, s_2) \leqslant 2c(Y, s_1, s_2)$ and T is a 2-approximation of the 2-MRCT.

As mentioned in Section 2, the shortest path forest can be constructed by an algorithm similar to the shortest path tree algorithm. The total time complexity is dominated by the time of the shortest path tree algorithm, which is $f_{SP}(G)$. □

As mentioned in Section 1, the 2-MRCT is a special case of the SROCT problem. By the result in [11], the 2-MRCT can be approximated with error

ratio 2 and time complexity $O(n^3)$. The simple algorithm ensures the same error ratio and is more efficient in time. The ratio shown in Lemma 4.1 is tight in the sense that there is an instance such that the spanning tree constructed by the algorithm has routing cost twice as the optimum. Consider the complete graph in which $w(v, s_1) = w(v, s_2) = 1$ and $w(s_1, s_2) = 2$ for each vertex v. The distance between any other pair of vertices is zero. At Step 1, the algorithm may find edge (s_1, s_2) as the path P, and then each other vertex is connected to one of the two sources. The routing cost of the constructed tree is $4n - 4$. On the optimal tree, the path between the two sources is a two-edge path, and all other vertices are connected to the middle vertex of the path. The optimal routing cost is therefore $2n$. The increased cost is due to missing the vertex on the path. The existence of the vertex reduces the cost at an amount of $w(P)$ for each vertex.

The worst case instance of the simple algorithm give us the intuition to improve the error ratio. To reduce the error, we may try to guess some vertices of the path. Let m be a vertex of the path between the two sources on the optimal tree, and U be the set of vertices connected to the path at m. If the path P found in Step 1 of the simple algorithm includes m, the distance from any vertex in U to each of the sources will be no more than the corresponding distance on the optimal tree. In addition, the vertex m partitions the path into two subpaths. The maximal increased cost by one of the vertices is the length of the subpath instead of the whole path. Our PTAS is to guess some of the vertices of the path, which partition the path such that the number of vertices connected to each subpath is small enough. We now describe the PTAS and the analysis precisely.

In the remaining paragraphs, let Y be the 2-MRCT of $G = (V, E, w)$ and $n = |V|$. Also let $P = (p_1 = s_1, p_2, p_3, \ldots, p_h = s_2)$ be the path between s_1 and s_2 on Y. Define V_i, $1 \leqslant i \leqslant h$, as the set of the vertices connected to P at p_i, and also $p_i \in V_i$. Let $k \geqslant 0$ be an integer. For $0 \leqslant i \leqslant k+1$, define $m_i = p_j$ in which j is the minimal index such that

$$\sum_{1 \leqslant q \leqslant j} |V_q| \geqslant \left\lceil i \frac{n}{k+1} \right\rceil.$$

By definition, $s_1 = m_0$ and $s_2 = m_{k+1}$. For $0 \leqslant i \leqslant k$, let $U_i = \bigcup_{a<j<b} V_j$ and P_i be the path from p_a to p_b, in which $p_a = m_i$ and $p_b = m_{i+1}$. Also let $U = V \setminus \bigcup_{0 \leqslant i \leqslant k} U_i$ and $M = \{m_i : 0 \leqslant i \leqslant k+1\}$. Note that the above definitions include the case $m_i = m_{i+1}$. In such a case, P_i contains only one vertex and U_i is empty. The definitions are shown in Fig. 3. By the above definitions, the vertex set V is partitioned into $U, U_0, U_1, \ldots, U_k$ satisfying the properties in the following lemmas.

Lemma 4.2. *For any $v \in U$, $d_G(v, M) \leqslant d_Y(v, P)$.*

Proof. Let $v \in V_i$. Since $p_i = m_j$ for some j,

$$d_Y(v, P) = d_Y(v, p_i) \geqslant d_G(v, p_i) = d_G(v, m_j) \geqslant d_G(v, M). \quad \square$$

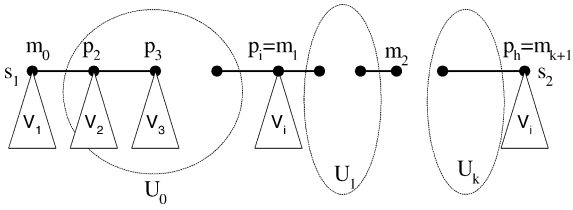

Fig. 3. The definitions of the partition of the vertices.

Lemma 4.3. $\sum_{v\in U_i} d_G(v, M) \leqslant \sum_{v\in U_i} d_Y(v, P) + \frac{n}{2(k+1)} w(P_i)$ *for any* $0 \leqslant i \leqslant k$.

Proof. For any $v \in V_j \subseteq U_i$,

$$\begin{aligned} d_G(v, M) &\leqslant \frac{1}{2}\big(d_G(v, m_i) + d_G(v, m_{i+1})\big) \\ &\leqslant \frac{1}{2}\big(\big(d_Y(v, P) + d_Y(m_i, p_j)\big) + \big(d_Y(v, P) + d_Y(p_j, m_{i+1})\big)\big) \\ &= d_Y(v, P) + \frac{1}{2} d_Y(m_i, m_{i+1}) \\ &= d_Y(v, P) + \frac{1}{2} w(P_i). \end{aligned}$$

Since $|U_i| \leqslant n/(k+1)$, we have

$$\begin{aligned} \sum_{v\in U_i} d_G(v, M) &\leqslant \sum_{v\in U_i} \left(d_Y(v, P) + \frac{1}{2} w(P_i) \right) \\ &\leqslant \sum_{v\in U_i} d_Y(v, P) + \frac{n}{2(k+1)} w(P_i). \quad \square \end{aligned}$$

The vertex set M is defined to partition the vertices into small pieces. Our goal is to correctly guess $m_1, m_2, \ldots, m_k$ and construct a tree X spanning these vertices along with s_1 and s_2, with the property that $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$. Once such a tree X has been constructed, we extend it to a spanning tree T by adding the shortest path forest with vertices in $V(X)$ as the multiple roots. In the next lemma, we show that T is a $(\frac{k+2}{k+1})$-approximation.

Lemma 4.4. *Let X be a tree spanning M and $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$. The spanning tree T, which is the union of X and the shortest path forest with vertices in $V(X)$ as the multiple roots, is a $(\frac{k+2}{k+1})$-approximation of the* 2*-MRCT.*

Proof. By definition, the vertices $m_1, m_2, \ldots, m_k$ partition the vertex set V into $(U, U_0, U_1, \ldots, U_k)$ and the subsets satisfy the properties in Lemmas 4.2 and 4.3. Since $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$, we have

$$\begin{aligned} c(T, s_1, s_2) &= \sum_{v\in V}\big(d_T(v, s_1) + d_T(v, s_2)\big) \\ &\leqslant \sum_{v\in V}\big(2d_T(v, X) + w(P)\big) \\ &= n \times w(P) + 2\sum_{v\in V} d_T(v, X) \\ &\leqslant n \times w(P) + 2\sum_{v\in U} d_T(v, X) + 2\sum_{0\leqslant i\leqslant k}\sum_{v\in U_i} d_T(v, X). \end{aligned} \tag{4}$$

Since $d_T(v, X) = d_G(v, X) \leqslant d_G(v, M)$ for any $v \in U$, by Lemma 4.2, we have

$$\sum_{v\in U} d_T(v, X) \leqslant \sum_{v\in U} d_G(v, M) \leqslant \sum_{v\in U} d_Y(v, P) \tag{5}$$

Similarly $d_T(v, X) \leqslant d_G(v, M)$ for any $v \in U_i$, and, by Lemma 4.3, we have

$$\begin{aligned} \sum_{0\leqslant i\leqslant k}\sum_{v\in U_i} d_T(v, X) &\leqslant \sum_{0\leqslant i\leqslant k}\left(\sum_{v\in U_i} d_Y(v, P) + \frac{n}{2(k+1)}w(P_i)\right) \\ &\leqslant \sum_{0\leqslant i\leqslant k}\sum_{v\in U_i} d_Y(v, P) + \frac{n}{2(k+1)}w(P). \end{aligned} \tag{6}$$

By Eqs. (4)–(6),

$$\begin{aligned} c(T, s_1, s_2) &\leqslant n \times w(P) + 2\sum_{v\in U} d_Y(v, P) + 2\sum_{0\leqslant i\leqslant k}\sum_{v\in U_i} d_Y(v, P) \\ &\quad + \frac{n}{k+1}w(P) \\ &\leqslant \frac{k+2}{k+1}n \times w(P) + 2\sum_{v\in V} d_Y(v, P) \\ &\leqslant \frac{k+2}{k+1}c(Y, s_1, s_2), \end{aligned}$$

since $c(Y, s_1, s_2) = n \times w(P) + 2\sum_{v\in V} d_Y(v, P)$ by Lemma 2.1. □

The PTAS is listed below.

Algorithm 2.

Input: A graph $G = (V, E, w)$, $s_1, s_2 \in V$, and an integer $k \geqslant 0$.
Output: A spanning tree T of G.
For each k-tuple $(m_1, m_2, \ldots, m_k)$ of not necessarily distinct vertices, use the following steps to find a spanning tree T, and output the tree of minimal routing cost.

1. Let $m_0 = s_1$ and $m_{k+1} = s_2$.
2. Find a tree X by the following substeps:
2.1. Initially X contains only one vertex m_0. /* $\delta = 0$. */
2.2. For $i = 0$ to k do
2.3. Find any shortest path Q from m_i to m_{i+1}.
Let $Q = (q_0 = m_i, q_1, \ldots, q_h = m_{i+1})$.
2.4. For $j = 0$ to $h - 1$ do
Let $\overline{X} = X$.
2.5. Add edge (q_j, q_{j+1}) to X. /* $\delta = \delta + w(q_j, q_{j+1})$. */
2.6. If X contains a cycle $(a_0 = q_{j+1}, a_1, a_2, \ldots, q_j, a_0)$ then
2.7. [Case 1.] $q_{j+1} \in V(SP_{\overline{X}}(s_1, q_j))$:
delete the edge (a_b, a_{b+1}) such that both $d_X(a_0, a_b)$ and $d_X(a_{b+1}, a_0)$ are no more than one half of the cycle length.
2.8. [Case 2.] $q_{j+1} \notin V(SP_{\overline{X}}(s_1, q_j))$: delete edge (a_0, a_1).
3. Find the shortest path forest spanning $V(G)$ with all vertices in $V(X)$ as the multiple roots.
4. Let T be the union of the forest and X.

The properties of X constructed at Step 2 are shown in the next lemma.

Lemma 4.5. *Suppose that $m_1, m_2, \ldots, m_k$ be k vertices such that P connects the consecutive m_i. The graph X constructed at Step 2 is a tree and $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$. Furthermore it takes $O(kn^2)$ to construct X.*

Proof. Starting from a single vertex, X is repeatedly augmented edge by edge. As in the comment of Step 2.5, let δ be the total weight of all edge had been added to X so far. When X is completed,

$$\delta = \sum_i SP_G(m_i, m_{i+1}) \leqslant \sum_i w(P_i) = w(P)$$

It is sufficient to show that the following two properties are kept at the end of Step 2.8.

- X is a tree.
- $d_X(v, s_1) + d_X(v, q_{j+1}) \leqslant \delta$ for any vertex $v \in V(X)$ so far, where (q_j, q_{j+1}) is the last edge added to X.

Initially the properties are true since X contains only one vertex. To avoid confusion, let $\overline{X}$ denote the constructed tree before edge (q_j, q_{j+1}) is added, and X denote the tree at the end of Step 2.8. Similarly let $\bar{\delta}$ and δ be the value before and after adding the edge respectively. It is obvious that X remains a tree if $\overline{X}$ is a tree, since it is connected and contains no cycle. If the edge does not cause a cycle, the second property is also straightforward. Now we consider that there is

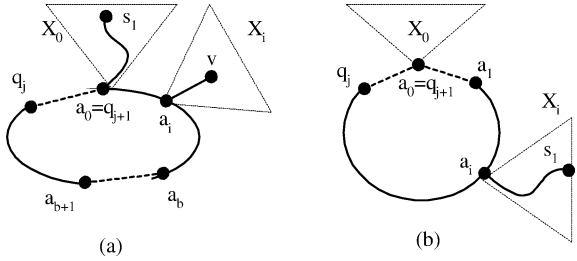

Fig. 4. Removing an edge from the cycle.

a cycle $(a_0 = q_{j+1}, a_1, a_2, \ldots, q_j, a_0)$. Let $X_i \subset V(X)$ denote the set of vertices which are connected to the cycle at a_i (Fig. 4).

Suppose that, for any vertex $v \in V(\overline{X})$,

$$d_{\overline{X}}(v, s_1) + d_{\overline{X}}(v, q_j) \leqslant \bar{\delta}. \tag{7}$$

We shall show that

$$d_X(v, s_1) + d_X(v, q_{j+1}) \leqslant \bar{\delta} + w(q_j, q_{j+1}) = \delta. \tag{8}$$

There are two cases.

Case 1. $q_{j+1} \in V(SP_{\overline{X}}(s_1, q_j))$, i.e., $s_1 \in X_0$ (Fig. 4(a)). First we show that the edge (a_b, a_{b+1}) always exists and can be found by traveling the cycle. Starting from a_0, we travel the cycle and compute the distance from a_0. We can always find vertex a_b such that $w(a_0, a_1, \ldots, a_b) \leqslant L/2$ and $w(a_0, a_1, \ldots, a_b, a_{b+1}) > L/2$, where L is the cycle length. After removing edge (a_b, a_{b+1}), $d_X(a_0, a_b) \leqslant L/2$ and $d_X(a_{b+1}, a_0) < L/2$. It implies that, for any vertex a_i on the cycle,

$$d_X(a_i, q_{j+1}) \leqslant \frac{L}{2}. \tag{9}$$

If $v \in X_0$, both the distances from v to s_1 and q_{j+1} do not change, and $d_X(v, q_{j+1}) \leqslant d_{\overline{X}}(v, q_j)$. Therefore Eq. (8) is true.

Otherwise, assume $v \in X_i$, $d_{\overline{X}}(v, s_1) = d_{\overline{X}}(v, a_i) + d_{\overline{X}}(a_i, q_{j+1}) + d_{\overline{X}}(q_{j+1}, s_1)$ and $d_{\overline{X}}(v, q_j) = d_{\overline{X}}(v, a_i) + d_{\overline{X}}(a_i, q_j)$. By Eq. (7), we have

$$\begin{aligned}
&d_{\overline{X}}(v, s_1) + d_{\overline{X}}(v, q_j) \leqslant \bar{\delta},\\
&2d_{\overline{X}}(v, a_i) + d_{\overline{X}}(a_i, q_{j+1}) + d_{\overline{X}}(q_{j+1}, s_1) + d_{\overline{X}}(a_i, q_j) \leqslant \bar{\delta},\\
&2d_{\overline{X}}(v, a_i) + \big(L - w(q_j, q_{j+1})\big) + d_{\overline{X}}(q_{j+1}, s_1) \leqslant \bar{\delta},\\
&2d_{\overline{X}}(v, a_i) + L + d_{\overline{X}}(q_{j+1}, s_1) \leqslant \delta.
\end{aligned}$$

By Eq. (9),

$$\begin{aligned}
d_X(v, s_1) + d_X(v, q_{j+1}) &= 2d_X(v, a_i) + 2d_X(a_i, q_{j+1}) + d_X(q_{j+1}, s_1)\\
&\leqslant 2d_{\overline{X}}(v, a_i) + L + d_{\overline{X}}(q_{j+1}, s_1) \leqslant \delta
\end{aligned}$$

Case 2. $q_{j+1} \notin V(SP_{\overline{X}}(s_1, q_j))$ (Fig. 4(b)). In this case edge (a_0, a_1) is removed. If $v \notin X_0$, $d_X(v, s_1) = d_{\overline{X}}(v, s_1)$ and $d_X(v, q_{j+1}) = d_{\overline{X}}(v, q_j) + w(q_j, q_{j+1})$. By Eq. (7), Eq. (8) is true.

Otherwise $v \in X_0$. Assume $s_1 \in X_i$. $d_{\overline{X}}(v, s_1) = d_{\overline{X}}(v, a_0) + d_{\overline{X}}(a_0, a_i) + d_{\overline{X}}(a_i, s_1)$ and $d_{\overline{X}}(v, q_j) = d_{\overline{X}}(v, a_0) + d_{\overline{X}}(a_0, q_j)$. By Eq. (7),

$$
\begin{aligned}
&d_{\overline{X}}(v, s_1) + d_{\overline{X}}(v, q_j) \leqslant \bar{\delta},\\
&2d_{\overline{X}}(v, a_0) + d_{\overline{X}}(a_i, s_1) + d_{\overline{X}}(a_0, a_i) + d_{\overline{X}}(a_0, q_j) \leqslant \bar{\delta},\\
&2d_{\overline{X}}(v, a_0) + d_{\overline{X}}(a_i, s_1) + d_{\overline{X}}(a_0, a_i) + L \leqslant \delta.
\end{aligned}
$$

The last step is obtained by $d_{\overline{X}}(a_0, q_j) + w(q_j, q_{j+1}) = L$. We have

$$
\begin{aligned}
d_X(v, s_1) + d_X(v, q_{j+1}) &= 2d_X(v, a_0) + d_X(a_0, a_i) + d_X(a_i, s_1)\\
&\leqslant 2d_{\overline{X}}(v, a_0) + L + d_{\overline{X}}(a_i, s_1) \leqslant \delta.
\end{aligned}
$$

We have shown Eq. (8) for both of the cases. By induction, $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$ at the end of Step 2. The number of vertices on the path $SP_G(m_i, m_{i+1})$ is at most $O(n)$. For each vertex on the path, we check if it causes a cycle and remove an edge from the cycle if it exists. All these can be done in $O(n)$ time. Consequently the time complexity is $O(kn^2)$. □

The main result of this section is concluded in the next theorem.

Theorem 4.1. *The* 2*-MRCT problem admits a PTAS. For any constant* $\varepsilon > 0$*, a* $(1 + \varepsilon)$*-approximation of the* 2*-MRCT of a graph* G *can be found in polynomial time. The time complexity is* $O(n^{\lceil 1/\varepsilon + 1 \rceil})$ *for* $0 < \varepsilon < 1$*, and* $f_{SP}(G)$ *for* $\varepsilon = 1$.

Proof. For any $\varepsilon > 0$, we choose an integer $k = \lceil 1/\varepsilon - 1 \rceil$ and run Algorithm A2. For $\varepsilon \geqslant 1$, i.e., $k = 0$, the algorithm is equivalent to Algorithm A1 and finds a 2-approximation in $f_{SP}(G)$ time.

For $0 < \varepsilon < 1$, i.e., $k \geqslant 1$, the algorithm tries each possible k-tuple in each iteration. There exists a k-tuple $(m_1, m_2, \ldots, m_k)$ which partitions the vertex set V into $(U, U_0, U_1, \ldots, U_k)$ and the subsets satisfy the properties in Lemmas 4.2 and 4.3. By Lemma 4.5 and Steps 3 and 4, the output T is a tree and therefore feasible. Also by Lemma 4.5, $d_X(v, s_1) + d_X(v, s_2) \leqslant w(P)$ for any $v \in V(X)$. Then, by Lemma 4.4, the spanning tree T constructed at Step 4 is a $(\frac{k+2}{k+1})$-approximation of the 2-MRCT. The number of all possible k-tuples is $O(n^k)$ for constant k. Similar to the proof of Lemma 4.1 and by Lemma 4.5, each iteration takes $O(kn^2)$ time. The time complexity of the algorithm is therefore $O(n^{k+2})$. That is, the ratio is $\frac{k+2}{k+1} = 1 + \varepsilon$, and the time complexity is $O(n^{\lceil 1/\varepsilon + 1 \rceil})$, which is polynomial for constant ε. □

5. The weighted 2-MRCT problem

In this section, we consider the weighted 2-MRCT. That is, we want to minimize $\sum_{v\in V}(\lambda_1 d_T(s_1, v) + \lambda_2 d_T(s_2, v))$, in which λ_1 and λ_2 are given positive real number. Without loss of generality, we define the objective function as $c(T, s_1, s_2, \lambda) = \sum_{v\in V}(\lambda d_T(s_1, v) + d_T(s_2, v))$, in which $\lambda \geqslant 1$. First we shall consider the case that the input is a general graph, and give a 2-approximation algorithm. Then we show that the problem admits a PTAS if the input is restricted to metric graphs.

5.1. On general graphs

First we present the 2-approximation algorithm for general graphs. Basically each vertex is greedily connected to one of the two sources, and then the two sources are connected by a shortest path.

Algorithm 3.

Input: A graph $G = (V, E, w)$, $s_1, s_2 \in V$, and a number $\lambda \geqslant 1$.
Output: A spanning tree T of G.

1. For each $v \in V$,
  compute $D_1(v) = (\lambda + 1)d_G(v, s_1) + d_G(s_1, s_2)$;
  compute $D_2(v) = (\lambda + 1)d_G(v, s_2) + \lambda d_G(s_1, s_2)$.
2. Let $Z_1 = \{v |\, D_1(v) \leqslant D_2(v)\}$ and $Z_2 = V \setminus Z_1$;
3. For each Z_i, with s_i as the root, find a shortest-path tree T_i spanning all vertices of Z_i.
4. Find a shortest path $Q = SP_G(s_1, s_2)$.
Let $Q = (q_0 = s_1, q_1, q_2, \ldots, q_j, q_{j+1}, \ldots s_2)$ in which q_{j+1} is the first vertex not in Z_1.
5. $T = T_1 \cup T_2 \cup (q_j, q_{j+1})$.

Before showing the performance of the algorithm, we show the correctness of Step 3.

Lemma 5.1. *At Step* 3, T_i *exists and is a shortest path tree.*

Proof. It is sufficient to show that, for each vertex $v \in Z_i$ and $Q = SP_G(v, s_i)$, $V(Q) \subset Z_i$. We show the case $i = 1$. The other case can be shown similarly. Since $v \in Z_1$,

$$(\lambda + 1)d_G(v, s_1) + d_G(s_1, s_2) \leqslant (\lambda + 1)d_G(v, s_2) + \lambda d_G(s_1, s_2)$$

For any $u \in V(Q)$, $d_G(v, s_1) = d_G(v, u) + d_G(u, s_1)$ and $d_G(v, s_2) \leqslant d_G(v, u) + d_G(u, s_2)$. Therefore, $(\lambda + 1)(d_G(v, u) + d_G(u, s_1)) + d_G(s_1, s_2) \leqslant (\lambda + 1) \times (d_G(v, u) + d_G(u, s_2)) + \lambda d_G(s_1, s_2)$. Then

$$(\lambda + 1)d_G(u, s_1) + d_G(s_1, s_2) \leqslant (\lambda + 1)d_G(u, s_2) + \lambda d_G(s_1, s_2).$$

It implies that $u \in Z_1$. □

Lemma 5.2. *Let T be the tree constructed at Step* 5. *The path between s_1 and s_2 on T is a shortest path. That is, $d_T(s_1, s_2) = d_G(s_1, s_2)$.*

Proof. Since T_1 is shortest path tree and q_j is a vertex on T_1, $d_T(s_1, q_j) = d_G(s_1, q_j)$. Similarly $d_T(s_2, q_{j+1}) = d_G(s_2, q_{j+1})$. Therefore $d_T(s_1, s_2) = d_T(s_1, q_j) + w(q_j, q_{j+1}) + d_T(q_{j+1}, s_2) = w(Q) = d_G(s_1, s_2)$. □

The approximation ratio and the time complexity are shown in the following theorem.

Theorem 5.1. *For a graph G, Algorithm* A3 *finds a* 2*-approximation of the weighted* 2*-MRCT with time complexity $f_{SP}(G)$.*

Proof. Similar as the proof of Lemma 4.1, the time complexity is dominated by the shortest path tree and therefore is $f_{SP}(G)$. We shall show that the algorithm finds a 2-approximation of the optimal. Let Y be the optimal tree and P be the path from s_1 to s_2 on Y. Let $f_1(v) = d_Y(v, s_1) - d_Y(v, P)$ and $f_2(v) = d_Y(v, s_2) - d_Y(v, P)$ for each vertex v. By the definition of routing cost, we have

$$c(Y, s_1, s_2, \lambda) = \sum_{v \in V} \big((\lambda + 1)d_Y(v, P) + \lambda f_1(v) + f_2(v)\big). \tag{10}$$

By Lemmas 5.1 and 5.2, each vertex is connected to either s_1 or s_2 by a shortest path, and $d_T(s_1, s_2) = d_G(s_1, s_2)$. Consider the cost in the case that vertex v is connected to s_1 by a shortest path. Since $d_G(v, s_1) \leqslant d_Y(v, P) + f_1(v)$ and $d_G(s_1, s_2) \leqslant w(P) = f_1(v) + f_2(v)$, we have

$$\begin{aligned}
&(\lambda + 1)d_G(v, s_1) + d_G(s_1, s_2) \\
&\quad \leqslant (\lambda + 1)d_Y(v, P) + (\lambda + 2) f_1(v) + f_2(v) \\
&\quad = \lambda d_Y(v, s_1) + d_Y(v, s_2) + 2 f_1(v).
\end{aligned} \tag{11}$$

That is, the cost is increased by at most $2 f_1(v)$. Similarly, in the case that v is connected to s_2 by a shortest path, we have

$$\begin{aligned}
&(\lambda + 1)d_G(v, s_2) + \lambda d_G(s_1, s_2) \\
&\quad \leqslant (\lambda + 1)d_Y(v, P) + \lambda f_1(v) + (2\lambda + 1) f_2(v) \\
&\quad = \lambda d_Y(v, s_1) + d_Y(v, s_2) + 2\lambda f_2(v).
\end{aligned} \tag{12}$$

The cost is increased by at most $2\lambda f_2(v)$. Since the cost $\lambda d_T(v, s_1) + d_T(v, s_2)$ is the minimum of the costs in the two cases,

$$\begin{aligned}&\big(\lambda d_T(v, s_1) + d_T(v, s_2)\big) - \big(\lambda d_Y(v, s_1) + d_Y(v, s_2)\big)\\ &\quad\leqslant \min\big\{2f_1(v), 2\lambda f_2(v)\big\}. \end{aligned} \tag{13}$$

By taking a weighted mean of the two numbers in Eq. (13), we have

$$\begin{aligned}\min\big\{2f_1(v), 2\lambda f_2(v)\big\} &\leqslant \frac{\lambda^2}{\lambda^2+1}\big(2f_1(v)\big) + \frac{1}{\lambda^2+1}\big(2\lambda f_2(v)\big)\\ &= \frac{2\lambda}{\lambda^2+1}\big(\lambda f_1(v) + f_2(v)\big)\\ &\leqslant \lambda f_1(v) + f_2(v). \end{aligned} \tag{14}$$

The last step is obtained by $2\lambda \leqslant \lambda^2 + 1$ since $\lambda^2 + 1 - 2\lambda = (\lambda - 1)^2 \geqslant 0$.

By Eqs. (13), (14), and summing over all vertices,

$$c(T, s_1, s_2, \lambda) - c(Y, s_1, s_2, \lambda) \leqslant \sum_{v\in V}\big(\lambda f_1(v) + f_2(v)\big).$$

Comparing with Eq. (10), we have $c(T, s_1, s_2, \lambda) \leqslant 2c(Y, s_1, s_2, \lambda)$, and therefore the approximation ratio is 2. □

5.2. On metric graphs

A metric graph is a complete graph and the edge weights satisfy the triangle inequality. In this subsection, we present a PTAS for the weighted 2-MRCT on metric graphs.

The main idea of the PTAS for the weighted case is similar to the unweighted one. We also try to guess k vertices of the path between the two sources on the optimal tree. But the algorithm is more simple since the input is a metric graph, which means that the edge between any pair of vertices is a shortest path. The algorithm is listed below.

Algorithm 4.

Input: A metric graph $G = (V, E, w)$, $s_1, s_2 \in V$, a number $\lambda \geqslant 1$, and an integer $k \geqslant 0$.
Output: A spanning tree T of G.
For each k-tuple $(m_1, m_2, \ldots, m_k)$ of not necessarily distinct vertices, use the following steps to find a spanning tree T, and output the tree of minimal cost $c(T, s_1, s_2, \lambda)$.
1. Let $m_0 = s_1$ and $m_{k+1} = s_2$.
2. Construct path $Q = (m_0, m_1, \ldots, m_{k+1})$ and set $T = Q$.
3. For each vertex v,
4. find m_i such that $(\lambda + 1)w(v, m_i) + \lambda d_Q(m_i, s_1) + d_Q(m_i, s_2)$ is

minimum, choosing the smaller index to break the tie;
5. insert edge (m_i, v) into T.

In the remaining paragraphs, we shall use the following notations. Let Y be the optimal tree and P be the path from s_1 to s_2 on Y. As defined in Section 4, let $M = \{m_0 = s_1, m_1, m_2, \ldots m_k, m_{k+1} = s_2\}$ be a set of not necessarily distinct vertices of $V(P)$, which partitions V into $U, U_0, U_1, \ldots, U_k$ such that $|U_i| \leqslant n/(k+1)$. The definitions of P_i, V_i are also the same as in the previous section. Since the algorithm tries all possible k-tuples and output the best of found trees, we may assume that T is the tree constructed by choosing the correct M.

Lemma 5.3. $\sum_{v\in U}(\lambda d_T(v,s_1) + d_T(v,s_2)) \leqslant \sum_{v\in U}(\lambda d_Y(v,s_1) + d_Y(v,s_2))$.

Proof. By definition, U is the set of the vertices which is connected to P at a vertex in M. Since $w(m_i, m_{i+1}) \leqslant w(P_i)$ for each i, $d_T(m_i,s_1) \leqslant d_Y(m_i,s_1)$ and $d_T(m_i,s_2) \leqslant d_Y(m_i,s_2)$. For any $v \in V_i \subseteq U$, since $p_i \in M$,

$$\begin{aligned}
&\lambda d_T(v,s_1) + d_T(v,s_2)\\
&\quad = \min_j\{(\lambda+1)w(v,m_j) + \lambda d_T(m_j,s_1) + d_T(m_j,s_2)\}\\
&\quad \leqslant \min_j\{(\lambda+1)d_Y(v,m_j) + \lambda d_Y(m_j,s_1) + d_Y(m_j,s_2)\}\\
&\quad \leqslant (\lambda+1)d_Y(v,p_i) + \lambda d_Y(p_i,s_1) + d_Y(p_i,s_2)\\
&\quad = \lambda d_Y(v,s_1) + d_Y(v,s_2).
\end{aligned}$$

The result is obtained by summing the inequality over all vertices of U. □

Lemma 5.4. $\sum_{v\in U_i}(\lambda d_T(v,s_1) + d_T(v,s_2)) \leqslant \sum_{v\in U_i}(\lambda d_Y(v,s_1) + d_Y(v,s_2)) + \frac{2n}{k+1}w(P_i)$.

Proof. As in the proof of Lemma 5.3, $d_T(m_i,s_1) \leqslant d_Y(m_i,s_1)$ and $d_T(m_i,s_2) \leqslant d_Y(m_i,s_2)$ for any $0 \leqslant i \leqslant k$.

For any $v \in U_i$, by definition, it is connected to some vertex of P_i on the optimal tree Y. If v is connected to m_i on T, the distance from v to s_1 is no more than that on Y, and the distance from v to s_2 is increased by at most $2w(P_i)$ because, by the triangle inequality,

$$\begin{aligned}
d_T(v,s_2) - d_Y(v,s_2) &= w(v,m_i) + w(P_i) - d_Y(v,m_{j+1})\\
&\leqslant d_Y(v,m_i) - d_Y(v,m_{i+1}) + w(P_i) \leqslant 2w(P_i).
\end{aligned}$$

That is,

$$\begin{aligned}
&(\lambda+1)w(v,m_i) + \lambda d_T(m_i,s_1) + d_T(m_i,s_2)\\
&\quad \leqslant \lambda d_Y(v,s_1) + d_Y(v,s_2) + 2w(P_i).
\end{aligned}$$

Since v is connected to some m_j to make the cost as small as possible, the cost on T is no more than the one in the above case. Therefore

$$\lambda d_T(v,s_1)+d_T(v,s_2)\leqslant \lambda d_Y(v,s_1)+d_Y(v,s_2)+2w(P_i).$$

Summing over all vertices in U_i, since $|U_i|\leqslant n/(k+1)$, we have

$$\begin{aligned}&\sum_{v\in U_i}\big(\lambda d_T(v,s_1)+d_T(v,s_2)\big)-\sum_{v\in U_i}\big(\lambda d_Y(v,s_1)+d_Y(v,s_2)\big)\\&\quad\leqslant\sum_{v\in U_i}2w(P_i)\leqslant\frac{2n}{k+1}w(P_i).\quad\square\end{aligned}\tag{15}$$

Theorem 5.2. *The weighted 2-MRCT problem on metric graphs admits a PTAS. For any $\varepsilon>0$, a $(1+\varepsilon)$-approximation of the optimal can be found in $O(n^{\lceil 2/\varepsilon\rceil})$ time.*

Proof. By Lemmas 5.3 and 5.4 and that $c(Y,s_1,s_2,\lambda)\geqslant n\times w(P)$, we have

$$\begin{aligned}&c(T,s_1,s_2,\lambda)-c(Y,s_1,s_2,\lambda)\\&\quad=\sum_{v\in U}\big(\lambda d_T(v,s_1)+d_T(v,s_2)-\big(\lambda d_Y(v,s_1)+d_Y(v,s_2)\big)\big)\\&\qquad+\sum_{0\leqslant i\leqslant k}\sum_{v\in U_i}\big(\lambda d_T(v,s_1)+d_T(v,s_2)-\big(\lambda d_Y(v,s_1)+d_Y(v,s_2)\big)\big)\\&\quad\leqslant\sum_{0\leqslant i\leqslant k}\frac{2n}{k+1}w(P_i)=\frac{2n}{k+1}w(P)\leqslant\frac{2}{k+1}c(Y,s_1,s_2,\lambda).\end{aligned}$$

Therefore T is a $(\frac{k+3}{k+1})$-approximation of the weighted 2-MRCT. The number of all possible k-tuple is $O(n^k)$ for constant k. Apparently each iteration takes $O(kn)$ time. The time complexity of the algorithm is $O(n^{k+1})$. For any $\varepsilon>0$, we choose $k=\lceil\frac{2}{\varepsilon}-1\rceil$. The approximation ratio is $1+\varepsilon$ and the time complexity is $O(n^{\lceil 2/\varepsilon\rceil})$. $\square$

6. Concluding remarks

An interesting problem is the computational complexity of the k-source MRCT for any fixed k. By the NP-hardness of 2-MRCT, we may easily show that the k-MRCT problem is also NP-hard for any fixed even number $k=2q$. For an instance of the 2-MRCT problem, we duplicate q copies of the two sources. If there is a polynomial time algorithm for the k-MRCT, it can also solve the 2-MRCT problem in polynomial time. However, we did not find any obvious reduction for fixed odd k.

The main question left in the paper is how to improve the approximation ratio for the weighted 2-MRCT of a general graph. By the previous result for the SROCT problem, the k-MRCT admits a 2-approximation algorithm for arbitrary k and for both weighted and unweighted cases. The 2-approximation algorithm in Section 5 only improves the time complexity. Although there is a PTAS for metric graphs, we did not find a similar result for general graphs.

Acknowledgment

We thank the anonymous referees for their useful comments.

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 1994.
[2] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[4] T.C. Hu, Optimum communication spanning trees, SIAM J. Comput. 3 (1974) 188–195.
[5] D.S. Johnson, J.K. Lenstra, A.H.G. Rinnooy Kan, The complexity of the network design problem, Networks 8 (1978) 279–285.
[6] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[7] M. Thorup, Undirected single source shortest paths with positive integer weights in linear time, J. ACM 46 (1999) 362–394.
[8] R. Wong, Worst-case analysis of network design problem heuristics, SIAM J. Algebraic Discrete Math. 1 (1980) 51–63.
[9] B.Y. Wu, K.M. Chao, C.Y. Tang, Approximation algorithms for the shortest total path length spanning tree problem, Discrete Appl. Math. 105 (2000) 273–289.
[10] B.Y. Wu, G. Lancia, V. Bafna, K.M. Chao, R. Ravi, C.Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees, SIAM J. Comput. 29 (2000) 761–778.
[11] B.Y. Wu, K.M. Chao, C.Y. Tang, Approximation algorithms for some optimum communication spanning tree problems, Discrete Appl. Math. 102 (2000) 245–266.
[12] B.Y. Wu, K.M. Chao, C.Y. Tang, A polynomial time approximation scheme for optimal product-requirement communication spanning trees, J. Algorithms 36 (2000) 182–204.