Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 143 (2004) 342–350
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Notes

An improved algorithm for the k-source maximum eccentricity spanning trees

Bang Ye Wu

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

Received 29 January 2003; received in revised form 14 December 2003; accepted 25 December 2003

Abstract

Let $G=(V,E,w)$ be an undirected graph with positive edge lengths and $S \subset V$ a set of k specified sources. The k-source maximum eccentricity spanning tree is a spanning tree T minimizing the maximum distance from a source to a vertex, i.e., $\max_{s\in S}\{\max_{v\in V}\{d_T(s,v)\}\}$, where $d_T(s,v)$ is the length of the path between s and v in T. In this paper, we propose an $O(|V|^2\log|V|+|V||E|)$ time algorithm, which improves the previous result on the problem.

Keywords: Algorithms; Network design; Spanning trees; Optimization problems

1. Introduction

An important family of problems in network design is to find spanning trees with small source-to-destination distances. Two natural objective functions are used to measure the goodness of spanning trees: *min–sum* and *min–max*. Let $G=(V,E,w)$ be an undirected graph with positive length function w on edges and $S \subset V$ a set of k specified sources, $1 \leqslant k \leqslant |V|$. The *k-source minimum routing cost spanning tree* (*k-MRCT*), or *k-source shortest path spanning tree* (*k-SPST*), is the spanning tree T minimizing $\sum_{s\in S}\sum_{v\in V} d_T(s,v)$, where $d_T(s,v)$ is the length of the path between s and v in T. While the k-MRCT problem defined by a min–sum objective function, the *k-source maximum eccentricity spanning tree* (*k-MEST*) problem is defined by a min–max objective function, and the goal is to find a spanning tree T minimizing $\max_{s\in S}\{\max_{v\in V}\{d_T(s,v)\}\}$.

If there is only one source, both the two problems can be solved by finding the *shortest-path tree*, a spanning tree in which 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 [3] finds a shortest path tree for a graph with non-negative 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 [2]. For undirected graph with positive integer weights, the most efficient algorithm runs in $O(|E|)$ time [10].

When there are more than one sources, the k-MRCT and k-MEST problems are of different complexities. The k-MRCT problem is a generalization of the MRCT problem (also called the *shortest total path length spanning tree* problem, [ND3] in [5]), in which all vertices are sources. The MRCT problem is NP-hard [5,7] and admits a *polynomial time approximation scheme* (PTAS) [13,16]. The k-MRCT problem for general k is obviously NP-hard since it includes the MRCT problem as a special case. But the NP-hardness of MRCT does not imply the complexity of the k-MRCT problem for a fixed constant k. The 2-MRCT problem has been shown to be NP-hard in [4,11]. Recently, the k-MRCT problem for any constant k has been shown to be also NP-hard [12].

E-mail address: bangye@mail.stu.edu.tw (B. Ye Wu).

0166-218X/$ - see front matter
doi:10.1016/j.dam.2003.12.004

The *optimum communication spanning tree* (OCT) problem [6] ([ND7] in [5]) 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. Approximation algorithms for OCT problem with some restricted requirements were studied [14,15]. One of the restricted OCT problem is the *optimal sum-requirement communication spanning tree* (SROCT) problem, in which each vertex has a non-negative weight and the requirement between two vertices is the sum of their weights. A 2-approximation algorithm for the SROCT problem was shown [14]. The k-MRCT problem can be thought of as a special case of the SROCT problem by setting each source of weight one and all the other vertices of weight zero. Therefore the 2-approximation algorithm for the SROCT problem ensures the same approximation ratio of the k-MRCT. For $k=2$, Farley et al. showed another 2-approximation algorithm [4], and the algorithm was independently discovered and generalized to a PTAS [11]. For any $\varepsilon > 0$, the PTAS finds a solution with approximation ratio $1+\varepsilon$ in $O(n^{\lceil 1/\varepsilon+1 \rceil})$ time.

In this paper, we focus on the k-MEST problem. It was shown that the problem can be solved in pseudo-polynomial time [4]. Although the algorithm is exponential for general cases, an important observation about the structure of an optimal tree was pointed out in their paper. Recently, McMahan and Proskurowski [9] presented an $O(|V|^3 + |E|\,|V| \log |V|)$ algorithm based on that observation. In [8], Krumme and Fragopoulou presented another algorithm which runs in $O(|V|^3 + |E|\,|V|)$ time and works for arbitrarily given source and destination sets. In this paper, we propose an $O(|V|^2 \log |V| + |E|\,|V|)$ algorithm for the problem. Although the method is similar to that in [8], our algorithm was developed independently and seems easier to understand.

The remaining sections are organized as follows: in Section 2, we define some notations and show some results about the central edge, which plays an important role in our algorithm. First we present an algorithm for the k-MEST of a metric graph in Section 3, and then generalize the algorithm to general graphs in Section 4, where a metric graph is a complete graph with triangle inequality. Finally concluding remarks are in Section 5.

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 positive edge length function w. For any graph G, $V(G)$ denotes its vertex set and $E(G)$ denotes its edge set. For two vertices u and v, the shortest path length in the input graph is denoted by $d(u,v)$. For a tree T, $d_T(u,v)$ is the length of the unique simple path between the two vertices. A vertex in a tree is a *leaf* if it is incident to only one edge, and a non-leaf vertex is called as an *internal node* (*vertex*). For $v \in V$ and $U \subset V$, define $D(v,U) = \max_{u \in U}\{d(v,u)\}$ the maximum of shortest path length from vertex v to any vertex in U. For a spanning tree T, the definition of $D_T(v,U)$ is similar except that the distance is on the tree T. The set of source vertices is denoted by S, and we assume $|S| = k > 1$. For $V_i \subset V$, $1 \leqslant i \leqslant 2$, we use S_i to denote $S \cap V_i$ the set of sources in V_i.

We now define the eccentricity and the k-MEST formally.

Definition 1. Let T be a tree and $S \subset V(T)$ the set of sources. The *maximum eccentricity* of T, denoted by $c(T)$, is the maximum distance from a source to a vertex, i.e., $c(T) = \max_{s \in S}\{\max_{v \in V}\{d_T(s,v)\}\}$, or $c(T) = \max_{v \in V}\{D_T(v,S)\}$ equivalently. Given a graph $G=(V,E,w)$ and a set of k sources $S \subset V$, the k-MEST problem is to find a spanning tree T with minimum $c(T)$ among all possible spanning trees.

A crucial observation is the existence of a *central edge* defined below:

Definition 2. Let T be a tree and $q = \max_{s_1,s_2 \in S}\{d_T(s_1,s_2)\}$ the maximum of intra-source distances on T. An edge $(m_1,m_2) \in E(T)$ is a central edge if $\min\{d_T(s,m_1), d_T(s,m_2)\} \leqslant q/2$ for any $s \in S$ and (m_1,m_2) is contained in a longest intra-source path.

The central edge is not unique if and only if there exists a vertex which is the midpoint of all longest intra-source paths.

Lemma 1. *Let T be a spanning tree of $G=(V,E,w)$ and $(m_1,m_2) \in E(T)$. Let T_1 and T_2 be the two trees obtained by removing (m_1,m_2) from T and $m_1 \in V(T_1)$. Let $S_i = V(T_i) \cap S$ for $i = 1,2$. The edge (m_1,m_2) is a central edge of T if and only if both S_1 and S_2 are not empty and*

$$|D_T(m_1,S_1) - D_T(m_2,S_2)| \leqslant w(m_1,m_2).$$

Proof. Let q be the maximum of intra-source distance in T, and let $D_i = D_T(m_i, S_i)$ for $i = 1, 2$. Suppose that (m_1, m_2) is a central edge. By definition, both S_1 and S_2 are not empty and $D_i \leqslant q/2$ for $i = 1, 2$. Since $D_1 + D_2 + w(m_1, m_2) = q$, we have

$$D_1 + w(m_1, m_2) \geqslant q/2 \geqslant D_2,$$

and

$$D_2 + w(m_1, m_2) \geqslant q/2 \geqslant D_1.$$

Consequently we obtain

$$|D_1 - D_2| \leqslant w(m_1, m_2).$$

Conversely suppose that both S_1 and S_2 are not empty and $|D_1 - D_2| \leqslant w(m_1, m_2)$. Without loss of the generality, we assume that $D_1 \geqslant D_2$. Since $D_1 \leqslant D_2 + w(m_1, m_2)$ and $D_1 + D_2 + w(m_1, m_2) \leqslant q$, we obtain $D_1 \leqslant q/2$. By definition, $D_T(m_i, S_i)$ is the maximum distance from any source in S_i to m_i. We have that $\min\{d_T(s, m_1), d_T(s, m_2)\} \leqslant q/2$ for any $s \in S$. To conclude that (m_1, m_2) is a central edge, we shall show that $D_1 + D_2 + w(m_1, m_2) = q$. By the triangle inequality, $d_T(s_1, m_1) + d_T(s_2, m_1) \geqslant d_T(s_1, s_2)$ for any $s_1, s_2 \in S_1$. We have $2D_1 \geqslant \max_{s_1, s_2 \in S_1}\{d_T(s_1, s_2)\}$. Since $D_1 \leqslant D_2 + w(m_1, m_2)$,

$$D_1 + D_2 + w(m_1, m_2) \geqslant \max_{s_1, s_2 \in S_1}\{d_T(s_1, s_2)\}.$$

Similarly we can show that

$$D_1 + D_2 + w(m_1, m_2) \geqslant \max_{s_1, s_2 \in S_2}\{d_T(s_1, s_2)\}.$$

By definition, $D_1 + D_2 + w(m_1, m_2)$ is the maximum distance between a source in S_1 and a source in S_2. Therefore

$$D_1 + D_2 + w(m_1, m_2) \geqslant \max_{s_1, s_2 \in S}\{d_T(s_1, s_2)\} = q. \quad \square$$

Lemma 2. *If* (m_1, m_2) *is a central edge of* T,

$$c(T) = w(m_1, m_2) + \max\{D_T(m_1, S_1) + D_T(m_2, V_2), D_T(m_2, S_2) + D_T(m_1, V_1)\}$$

in which V_1 *and* V_2 *are the vertex sets of* T_1 *and* T_2 (*defined in Lemma* 1), *respectively*.

Proof. By Lemma 1, $D_T(m_1, S_1) \leqslant D_T(m_2, S_2) + w(m_1, m_2)$, which implies that the furthest source to m_1 is in S_2. Consequently, for any vertex $v \in V_1$,

$$\max_{s \in S}\{d_T(v, s)\} = \max_{s \in S_2}\{d_T(v, s)\}$$
$$= d_T(v, m_1) + w(m_1, m_2) + D_T(m_2, S_2).$$

Taking the maximum over all vertices in V_1, we have

$$\max_{v \in V_1}\left\{\max_{s \in S}\{d_T(v, s)\}\right\} = D_T(m_1, V_1) + w(m_1, m_2) + D_T(m_2, S_2).$$

Similarly,

$$\max_{v \in V_2}\left\{\max_{s \in S}\{d_T(v, s)\}\right\} = D_T(m_2, V_2) + w(m_1, m_2) + D_T(m_1, S_1).$$

The result follows the definition of $c(T)$. $\square$

3. On metric graphs

In this section, we consider the k-MEST problem on metric graphs. A metric graph $G = (V, E, w)$ is a complete graph with $w(u, v) = d(u, v)$ for any $u, v \in V$, i.e., any edge is a shortest path between its two endpoints.

Definition 3. A *2-star* is a tree with at most two internal nodes. In the case of two internal nodes, the edge between the two internal nodes is the unique central edge of the tree.

Lemma 3. *For a metric graph, there exists a k-MEST* T *which is a* 2-*star*.

Proof. Let Y be a k-MEST of a metric graph G and (m_1, m_2) be a central edge of Y. Let V_1 and V_2 be the vertex sets of the two components resulted by removing (m_1, m_2) from Y and $m_1 \in V_1$. By Lemma 2,

$$c(Y) = w(m_1, m_2) + \max\{D_Y(m_1, S_1) + D_Y(m_2, V_2), D_Y(m_2, S_2) + D_Y(m_1, V_1)\}.$$

We consider three cases.

Case 1: $D(m_1, S_1) + w(m_1, m_2) < D(m_2, S_2)$. We construct T by connecting all vertices to m_2, i.e., T is the star centered at m_2. Since

$$D(m_2, S_1) \leqslant D(m_1, S_1) + w(m_1, m_2) < D(m_2, S_2),$$

we have $D(m_2, S) = D(m_2, S_2)$. Let $s \in S_2$ the furthest source to m_2, i.e., $d(s, m_2) = D(m_2, S)$. For any $v \in V_1$,

$$d_T(v, s) = d(v, m_2) + d(m_2, s) \leqslant d_Y(v, s) \leqslant c(Y).$$

For any vertex $v \in V_2$, since (m_1, m_2) is a central edge of Y, $d(m_2, s) \leqslant d_Y(m_2, s) \leqslant D_Y(m_2, S_1)$. We have

$$\begin{aligned} d_T(v, s) &= d(v, m_2) + d(m_2, s) \\ &\leqslant d(v, m_2) + D_Y(m_2, S_1) \\ &\leqslant D_Y(v, S_1) \leqslant c(Y). \end{aligned}$$

Since T is a star,

$$c(T) = d(m_2, s) + \max\{D(m_2, V_1), D(m_2, V_2)\} \leqslant c(Y).$$

By definition, a star is also a 2-star, and therefore T is the desired k-MEST and (m_2, s) is its central edge.

Case 2: $D(m_2, S_2) + w(m_1, m_2) < D(m_1, S_1)$. The case is similar, and the desired tree is the star centered at m_1.

Case 3: $|D(m_1, S_1) - D(m_2, S_2)| \leqslant w(m_1, m_2)$. We construct a spanning tree T as follows. For all vertices in V_1, we construct a star centered at m_1, and similarly construct a star centered at m_2 for all vertices in V_2. Then we connect the two stars into a tree T by inserting edge (m_1, m_2). By Lemma 1, the edge (m_1, m_2) is also a central edge of T. Since G is a metric graph, by Lemma 2,

$$\begin{aligned} c(T) &= w(m_1, m_2) + \max\{D(m_1, S_1) + D(m_2, V_2), D(m_2, S_2) + D(m_1, V_1)\} \\ &\leqslant w(m_1, m_2) + \max\{D_Y(m_1, S_1) + D_Y(m_2, V_2), D_Y(m_2, S_2) + D_Y(m_1, V_1)\} \\ &= c(Y). \quad \square \end{aligned}$$

In order to find a k-MEST, our algorithm tries all the edges. For each edge (m_1, m_2), we find the best 2-star with (m_1, m_2) as its central edge. For specified central edge, a 2-star is determined by the vertex partition (V_1, V_2), in which V_1 and V_2 are the vertex sets of the two components obtained by removing the central edge. We shall focus on how to find the best bipartition. Define a cost function on any partition (V_1, V_2) of V

$$C(V_1, V_2) = \max\{D(m_1, S_1) + D(m_2, V_2), D(m_2, S_2) + D(m_1, V_1)\}.$$

By Lemmas 2 and 3, it is sufficient to find (V_1, V_2) minimizing $C(V_1, V_2)$ subject to $|D(m_1, S_1) - D(m_2, S_2))| \leqslant w(m_1, m_2)$ and $S_i \neq \emptyset$ for $i = 1, 2$.

First we focus on the partitions of S. Define the xy-pair of a bipartition (S_1, S_2) of S to be the ordered pair (x, y) in which $x = D(m_1, S_1)$ and $y = D(m_2, S_2)$. Since any source is in either S_1 or S_2, the next fact is trivial.

Fact 4. If (x, y) is a xy-pair, there is no source s with $d(s, m_1) > x$ and $d(s, m_2) > y$ simultaneously.

Let $\mathscr{P} = \{(d(s_1, m_1), d(s_2, m_2)) | s_1, s_2 \in S\}$. There are at most k^2 different xy-pairs among the 2^k bipartitions. However, to minimize the cost, not all possible pairs need to be considered.

Definition 4. An xy-pair (x, y) is *minimal* if

(1) for any other $(x_1, y_1) \in \mathscr{P}$, $x_1 > x$ or $y_1 > y$; and
(2) there exists a source s with $d(s, m_1) = x$ and $d(s, m_2) > y$.

Obviously, to minimize the cost, a pair (x, y) is useless if there exists another pair (x_1, y_1) such that $x_1 \leqslant x$ and $y_1 \leqslant y$. Let (V_1, V_2) be a bipartition of V. If there exists a source $s \in V_1$ with $d(s, m_2) \leqslant D(m_2, S_2)$, we can move s from S_1 to S_2 without increasing the maximum eccentricity of the tree. Therefore, we only need to consider the minimal xy-pairs. Clearly there are at most $(k - 1)$ minimal xy-pairs since there are at most k different x-values.

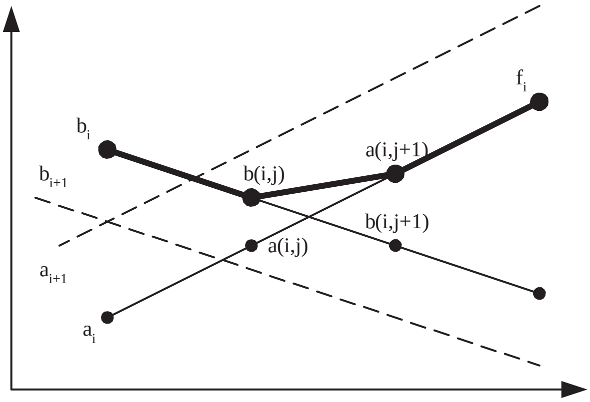

Fig. 1. $a(i,j)$, $b(i,j)$, and $f(i,j)$ (bold line).

Let L_s be the list of all minimal pairs (x_i, y_i) satisfying $|x_i - y_i| \leqslant w(m_1, m_2)$, where all x_i are in increasing order and all y_i are in decreasing order. The next lemma shows that L_s can be constructed efficiently.

Lemma 5. *Given the sorted distances from sources to m_1, the list L_s can be constructed in* $O(k)$ *time.*

Proof. Let $S = \{s_i | 1 \leqslant i \leqslant k\}$ and $d(m_1, s_i) \leqslant d(m_1, s_{i+1})$ for $1 \leqslant i < k$. Let L be the list of (x_i, y_i), in which $x_i = d(m_1, s_i)$ and $y_i = \max_{j>i}\{d(s_j, m_2)\}$ for $1 \leqslant i < k$. By definition and Fact 4, L_s is a subsequence of L. Since

$$y_i = \max\{y_{i+1}, d(s_{i+1}, m_2)\},$$

starting with $y_{k-1} = d(s_k, m_2)$, the list L can be constructed in linear time. Then the desired list L_s can be obtained by removing the non-minimal pairs and the pairs with $|x_i - y_i| > w(m_1, m_2)$. The time complexity is $O(k)$. □

Let $U = V \setminus S$ and (U_1, U_2) be the projection of a partition (V_1, V_2) of V onto U, i.e., $U_1 = U \cap V_1$ and $U_2 = U \cap V_2$. Define the pq-pair of (U_1, U_2) to be the ordered pair (p, q) in which $p = D(m_1, U_1)$ and $q = D(m_2, U_2)$. Similar to Lemma 5, the sorted list L_u of all minimal pq-pairs (p_i, q_i) can be constructed in $O(|U|)$ time. For convenience, we reverse the list so that all q_i's are in increasing order and all p_i's are in decreasing order.

Define $a(i,j) = x_i + q_j$, $b(i,j) = y_i + p_j$ and $f(i,j) = \max\{a(i,j), b(i,j)\}$. The goal is to find the minimum of $f'(i,j) = \max\{f(i,j), x_i + y_i\}$, which is exactly the minimum of $C(V_1, V_2)$. Define $a_i(j) = a(i,j)$, $b_i(j) = b(i,j)$, and $f_i(j) = f(i,j)$ for some fixed i. Our algorithm computes the minimum of $f_i(j)$ for each i, and the minimum of $f'(i,j)$ can be obtained directly. We observe the following:

Fact 6. The function $a(i,j)$ is monotonically increasing for both i and j, and $b(i,j)$ is monotonically decreasing for both i and j.

The function a_i is monotonically increasing and b_i is monotonically decreasing. As a consequence, f_i is bitonic: monotonically decreasing and then monotonically increasing (Fig. 1). Suppose that f_i achieves its minimum at j. It is easy to show that a_i dominates b_i at the right side of j and b_i dominates a_i at the left side. For the completeness, we give a formal proof.

Fact 7. If f_i achieves its minimum at j, $b_i(j+1) < a_i(j+1)$ and $a_i(j-1) < b_i(j-1)$.

Proof. Since f_i achieves its minimum at j, we have

$$f_i(j) \leqslant f_i(j+1)$$

$$\max\{a_i(j), b_i(j)\} \leqslant \max\{a_i(j+1), b_i(j+1)\}$$

$$b_i(j) \leqslant \max\{a_i(j+1), b_i(j+1)\}.$$

However, $b_i(j+1) < b_i(j)$ since b_i is monotonically decreasing, and we have

$$b_i(j+1) < a_i(j+1)$$

The other part can be shown similarly. □

By the above fact, the minimum of f_i can be found by a method similar to the binary search. The time complexity for finding the minimum of $f(i,j)$ is $O(|V| + k\log|V|)$. However, the complexity is $O(|V|\log|V|)$ for large k. In the following, we shall show how to reduce the complexity to $O(|V|)$. The key point is stated in the next lemma.

Lemma 8. *If f_i and f_{i+1} achieve their minima at j and j', respectively, then $j' \leqslant j$.*

Proof. Since f_i achieve its minimum at j, $f_i(j) \leqslant f_i(j+1)$. By definition and Fact 7,

$$\max\{a(i,j), b(i,j)\} \leqslant \max\{a(i,j+1), b(i,j+1)\} = a(i,j+1).$$

We have $b(i,j) \leqslant a(i,j+1)$. Since $b(i+1,j) < b(i,j)$ and $a(i,j+1) < a(i+1,j+1)$, we obtain

$$b(i+1,j) < a(i+1,j+1).$$

Since $a(i+1,j) < a(i+1,j+1)$,

$$\begin{aligned} f_{i+1}(j) &= \max\{a(i+1,j), b(i+1,j)\} \\ &< a(i+1,j+1) \\ &\leqslant \max\{a(i+1,j+1), b(i+1,j+1)\} = f_{i+1}(j+1). \end{aligned}$$

We have shown that $f_{i+1}(j) < f_{i+1}(j+1)$. Since f_{i+1} is bitonic, the result follows. □

Lemma 9. *Given L_s and L_u, the minimum of $f'(i,j)$ can be computed in $O(|V|)$ time.*

Proof. Let f_i achieve its minimum at j_i. By Lemma 8, we have $j_i \geqslant j_{i+1}$ for each i. Starting at $j_0 = |L_u|$, we compute $f_1(j_0), f_1(j_0-1), \ldots$ until we find $f_1(j-1) \geqslant f_1(j)$ for some j. Since f_1 is bitonic, $j_1 = j$. Once j_1 is found, j_2 can be determined similarly but we start at $j = j_1$ instead of j_0. Obviously all the minima can be found in totally $O(|V|)$ time. Finally the minimum of $f'(i,j)$ is given by $\min_i\{\max\{f_i(j_i), x_i + y_i\}\}$ and can be also obtained in $O(|V|)$ time. □

Our algorithm for k-MEST of a metric graph is stated below:

Algorithm A1.

Input: A metric graph $G = (V,E,w)$ and $S \subset V$.
Output: A spanning tree T of G.
1. For each vertex v, sort the distances from v to all others.
2. For each edge (m_1, m_2) do
2.1. Compute the sorted list L_s of the minimal xy-pairs.
2.2. Compute the sorted list L_u of the minimal pq-pairs.
2.3. Find i^* and j^* minimizing $f'(i,j)$.
2.4. Let $cost(m_1,m_2) = w(m_1,m_2) + f'(i^*, j^*)$.
3. Let (m_1,m_2) and i and j minimize the cost at Step 2.
3.1. Construct $V_1 = \{s | s \in S, d(s,m_1) \leqslant x_i\} \cup \{v | v \notin S, d(v,m_1) \leqslant p_j\}$.
3.2. Construct $V_2 = V \setminus V_1$.
3.3. $E(T) = \{(m_1,m_2)\} \cup \{(v,m_1) | v \in V_1\} \cup \{(v,m_2) | v \in V_2\}$.

Theorem 10. *The k-MEST problem for a metric graph $G = (V,E,w)$ can be solved in $O(|V|^2\log|V| + |V|\,|E|)$ time.*

Proof. The correctness of the algorithm follows Lemma 3. The first step takes $O(|V|^2\log|V|)$ time. For each edge $(m_1,m_2) \in E(G)$, by Lemmas 5 and 9, Steps 2.1–2.4 take $O(|V|)$ time. Constructing the tree at Step 3 takes $O(|V|)$ time. Therefore the time complexity is $O(|V|^2\log|V| + |V|\,|E|)$. □

4. On general graphs

In this section, we generalize the algorithm to the case of general graphs. For a given general graph G, we first compute the all-pair shortest path lengths, and then find the k-MEST of the metric closure (defined later) of G. Finally we construct a spanning tree of G with the same eccentricity.

Definition 5. Let $G=(V,E,w)$ be a graph. The *metric closure*, denoted by $\bar{G}=(V,\bar{E},\bar{w})$, is a complete graph in which the length of an edge is the shortest path length of the two endpoints on G, i.e., $\bar{w}(u,v)=d(u,v)$ for each $u,v\in V$.

The next corollary is derived from Lemma 3.

Corollary 11. *Let $\bar{G}$ be the metric closure of a graph $G=(V,E,w)$. There exists a k-MEST T of $\bar{G}$ such that T is a 2-star and T has a central edge (m_1,m_2) with $w(m_1,m_2)=d(m_1,m_2)$.*

Proof. By Lemma 3, there exists such a k-MEST T with central edge (m_1,m_2) except that the central edge may be not in E or $w(m_1,m_2)>d(m_1,m_2)$. For both cases, we transform T into Y by replacing the central edge with a shortest path P between the two endpoints. Clearly $c(Y)=C(T)$ since the eccentricity does not increase and T is optimal. Furthermore there exists a longest intra-source path containing P and its length remains the same. Otherwise T is not optimal. Therefore we can find an edge of P which is a central edge of Y. Since P is a shortest path on G, $w(u,v)=d(u,v)$ for each edge $(u,v)\in E(P)$. Using the same procedure in the proof of Lemma 3, we can construct the desired k-MEST. □

Our algorithm is in the following.

Algorithm A2.

Input: A graph $G=(V,E,w)$ and $S\subset V$.
Output: A spanning tree T of G.
1. Construct the metric closure $\bar{G}=(V,\bar{E},\bar{w})$ of G.
2. For each vertex v, sort the distances from v to all others.
3. For each edge $(m_1,m_2)\in E$ do
3.1. Find i^* and j^* minimizing $f'(i,j)$.
3.2. Compute $cost(m_1,m_2)=w(m_1,m_2)+f'(i^*,j^*)$.
4. Let (m_1,m_2) and i and j minimize the cost at Step 3.
4.1. Partition V into V_1 and V_2 such that V_1 contains m_1 and all vertices v with $d(v,m_1)-d(v,m_2)\leqslant x_i-y_i$.
4.2. On G, construct a shortest path tree T_1 rooted at m_1 and spanning V_1.
4.3. On G, construct a shortest path tree T_2 rooted at m_2 and spanning V_2.
4.4. $E(T)=E(T_1)\cup E(T_2)\cup\{(m_1,m_2)\}$.

First we show the validity of the tree T_1 and T_2 constructed at Steps 4.2 and 4.3. The next two lemmas are sufficient.

Lemma 12. *At Step 4, $m_1\in V_1$, and $V_2=\emptyset$ if $m_2\in V_1$.*

Proof. Since $(x_i,y_i)\in L_s$, $|x_i-y_i|\leqslant w(m_1,m_2)$. We have

$$x_i-y_i\geqslant -w(m_1,m_2)=d(m_1,m_1)-d(m_1,m_2),$$

and therefore $m_1\in V_1$.

Suppose that $m_2\in V_1$. By the definition of V_1, we have

$$d(m_2,m_1)-d(m_2,m_2)=w(m_1,m_2)\leqslant x_i-y_i.$$

However, $x_i-y_i\leqslant w(m_1,m_2)$, and consequently $x_i-y_i=w(m_1,m_2)$. By triangle inequality, $d(v,m_1)\leqslant d(v,m_2)+w(m_1,m_2)$ for any vertex v, and we obtain

$$d(v,m_1)-d(v,m_2)\leqslant w(m_1,m_2)=x_i-y_i.$$

Therefore all vertices are in V_1 and $V_2=\emptyset$. □

Lemma 13. *Any shortest path P from a vertex $v \in V_1$ to m_1 contains no vertex in V_2. Similarly, any shortest path from a vertex in V_2 to m_2 contains no vertex in V_1.*

Proof. We show the first part, and the other one can be shown similarly. Suppose that there exists a vertex $u \in V_2 \cap V(P)$. By definition, $d(u,m_1) - d(u,m_2) > x_i - y_i$. However, since $u \in V(P)$ and P is a shortest path, $d(v,m_1) = d(v,u) + d(u,m_1)$. We have

$$(d(u,m_1) + d(u,v)) - (d(u,m_2) + d(u,v)) > x_i - y_i,$$

$$d(v,m_1) - (d(u,m_2) + d(u,v)) > x_i - y_i,$$

$$d(v,m_1) - d(v,m_2) > x_i - y_i,$$

since $d(v,m_2) \leqslant d(u,m_2) + d(u,v)$. It implies that v is in V_2, a contradiction. □

Now we show that T is optimal.

Lemma 14. $c(T) \leqslant cost(m_1,m_2)$.

Proof. In the case that $V_2 = \emptyset$, m_1 is the midpoint of the longest intra-source path, and T is a shortest path tree rooted at m_1. It is easy to show that $c(T) \leqslant cost(m_1,m_2)$ since, for any vertex, the distance to the furthest source does not increase. In the following, we assume that $V_2 \neq \emptyset$.

Since (x_i, y_i) is a minimal pair in L_s, there exists a source s with $d(s,m_1) = x_i$ and $d(s,m_2) > y_i$. By the definition of V_1, we have that $s \in V_1$, which implies that $D(m_1,S_1) \geqslant x_i$. Suppose that there is a source $s_1 \in V_1$ with $d(s_1,m_1) > x_i$. Since $d(s_1,m_1) - d(s_1,m_2) \leqslant x_i - y_i$, we have

$$d(s_1,m_2) \geqslant y_i + (d(s_1,m_1) - x_i) > y_i,$$

a contradiction to Fact 4. Consequently $D(m_1,S_1) = x_i$. Similarly we can show that $D(m_2,S_2) = y_i$.

By the definition of L_s, $|x_i - y_i| \leqslant \bar{w}(m_1,m_2) = w(m_1,m_2)$. Since T_1 and T_2 are shortest path trees, we have $|D_T(m_1,S_1) - D_T(m_2,S_2)| \leqslant w(m_1,m_2)$ and conclude that (m_1,m_2) is a central edge of T by Lemma 1.

By definition, $cost(m_1,m_2) = w(m_1,m_2) + \max\{f(i,j), x_i + y_i\}$. We consider two cases.

Case 1: $x_i + y_i \geqslant f(i,j)$. It implies that there is no vertex v with $d(v,m_1) > x_i$ and $d(v,m_2) > y_i$ simultaneously. For $v \in V_1$, if $d(v,m_1) > x_i$, by the definition of V_1,

$$d(v,m_1) - d(v,m_2) \leqslant x_i - y_i,$$

$$d(v,m_2) \geqslant d(v,m_1) - x_i + y_i > y_i.$$

It is a contradiction, and therefore $D(m_1,V_1) = x_i$. Similarly $D(m_2,V_2) = y_i$, and we have $c(T) = cost(m_1,m_2)$.

Case 2: $cost(m_1,m_2) = w(m_1,m_2) + f(i,j)$. Recall that $f(i,j) = \max\{x_i + q_j, y_i + p_j\}$. Let $U = V \setminus S$ and $U_i = V_i \setminus S_i$ for $i = 1,2$. Since (p_j,q_j) is a minimal pair in L_u, there is no vertex $u \in U$ with $d(u,m_1) > p_j$ and $d(u,m_2) > q_j$ simultaneously. For any $u \in U_1$, if $d(u,m_1) \leqslant p_j$, we have

$$d(u,m_1) + y_i \leqslant p_j + y_i \leqslant f(i,j).$$

Otherwise, $d(u,m_2) \leqslant q_j$. Since $u \in U_1$,

$$\begin{aligned} d(u,m_1) + y_i &\leqslant x_i + d(u,m_2) \\ &\leqslant x_i + q_j \leqslant f(i,j). \end{aligned}$$

Consequently, $d(u,m_1) + y_i \leqslant f(i,j)$ for any $u \in U_1$, and therefore $D_T(U_1,m_1) + y_i \leqslant f(i,j)$. Similarly we can show that $D_T(U_2,m_2) + x_i \leqslant f(i,j)$. Since (m_1,m_2) is a central edge of T,

$$\begin{aligned} c(T) &= w(m_1,m_2) + \max\{D_T(m_1,U_1) + y_i, D_T(m_2,U_2) + x_i\} \\ &\leqslant w(m_1,m_2) + f(i,j) \\ &= cost(m_1,m_2). \quad \square \end{aligned}$$

By definition, any spanning tree of a graph is also a spanning tree of its metric closure. Since $cost(m_1,m_2)$ is the minimum of the maximum eccentricity of any spanning tree of $\bar{G}$, we have $cost(m_1,m_2) \leqslant c(T)$, and the next corollary follows Lemma 14.

Corollary 15. $c(T) = cost(m_1, m_2)$.

Corollary 16. *Let T be a k-MEST of a graph G. If Y is a k-MEST of the metric closure $\bar{G}$, $c(Y) = c(T)$.*

Theorem 17. *The k-MEST problem for a graph $G = (V, E, w)$ can be solved in* $O(|V|^2 \log |V| + |V|\,|E|)$ *time.*

Proof. The correctness of the algorithm follows Lemmas 13, 14, and Corollary 16. To compute the all-pair shortest path lengths, it takes $O(|V|^2 \log |V| + |V|\,|E|)$ time for Step 1 [2]. Step 2 takes $O(|V|^2 \log |V|)$ time for sorting. Step 3 takes $O(|V|)$ time for each edge of G, and therefore $O(|V|\,|E|)$ time in total. Finally constructing the tree can be done in $O(|V| \log |V| + |E|)$ time. In summary, the time complexity is $O(|V|^2 \log |V| + |V|\,|E|)$. □

5. Concluding remarks

In this paper, we present an algorithm for the k-MEST with time complexity $O(|V|^2 \log |V| + |V|\,|E|)$, which is more efficient than the previous algorithms for general graphs. Besides the k-MRCT and the k-MEST problems, some other cost metrics for the multi-source spanning trees were discussed recently [1]. Efficient algorithms or approximation algorithms for those problems may be an interesting future work.

References

[1] H.S. Connamacher, A. Proskurowski, The complexity of minimizing certain cost metrics for k-source spanning trees, Discrete Appl. Math. 131 (2003) 113–127.
[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1994.
[3] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[4] A.M. Farley, P. Fragopoulou, D.W. Krumme, A. Proskurowski, D. Richards, Multi-source spanning tree problems, J. Interconnect. Networks 1 (2000) 61–71.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[6] T.C. Hu, Optimum communication spanning trees, SIAM J. Comput. 3 (1974) 188–195.
[7] D.S. Johnson, J.K. Lenstra, A.H.G. Rinnooy Kan, The complexity of the network design problem, Networks 8 (1978) 279–285.
[8] D.W. Krumme, P. Fragopoulou, Minimum eccentricity multicast trees, Discrete Math. Theoret. Comput. Sci. 4 (2001) 157–172.
[9] H.B. McMahan, A. Proskurowski, Multi-source spanning trees: algorithms for minimizing source eccentricities, Discrete Appl. Math. 137 (2004) 213–222.
[10] M. Thorup, Undirected single source shortest paths with positive integer weights in linear time, J. ACM 46 (1999) 362–394.
[11] B.Y. Wu, A polynomial time approximation scheme for the two-source minimum routing cost spanning trees, J. Algorithm. 44 (2002) 359–378.
[12] B.Y. Wu, Approximation algorithms for the optimal p-source communication spanning tree, Discrete Appl. Math., to appear.
[13] 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.
[14] 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.
[15] B.Y. Wu, K.-M. Chao, C.Y. Tang, A polynomial time approximation scheme for optimal product-requirement communication spanning trees, J. Algorithm. 36 (2000) 182–204.
[16] 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.