Discrete Mathematics and Theoretical Computer Science **4**, 2001, 157–172

Minimum Eccentricity Multicast Trees

David W. Krumme[1] and Paraskevi Fragopoulou[2]

[1] *Electrical Engineering and Computer Science Department, Tufts University, Medford MA 02155, USA.*
Email: `dwk@eecs.tufts.edu`

[2] *Laboratoire PRiSM, Universite de Versailles-St. Quentin en Yvelines, 45 avenue des Etats-Unis, 78035 Versailles Cedex, France.*
Email: `vivi@prism.uvsq.fr`

received Dec 29, 2000, *accepted Sep 1, 2001*.

We consider the problem of constructing a multicast tree that connects a group of source nodes to a group of sink nodes (receivers) and minimizes the maximum end-to-end delay between any pair of source/sink nodes. This is known as the *minimum eccentricity multicast tree* problem, and is directly related to the quality of service requirements of real multipoint applications. We deal directly with the problem in its general form, meaning that the sets of source and sink nodes need not be overlapping nor disjoint. The main contribution of this work is a polynomial algorithm for this problem on general networks which is inspired by an innovative method that uses geometric relationships on the xy-plane.

Keywords: multicast, eccentricity, algorithm, communications, graph, network

1 Introduction

Multipoint applications are of growing importance in broadband communication networks and the Internet. In a multipoint application, a number of participants, remotely located, wish to exchange data for a duration of time. The simplest possible approach would be to have each dissemination take the form of a broadcast from the originator of the data to a set of receivers. This approach suffers from the overhead of setting up broadcast trees tailored to each use. An approach with less overhead is to construct a single tree to be used by varying sets of senders and receivers during the multipoint application. The price paid for this reduction in overhead is an increase in latencies and traffic concentration [2]. Shared trees have been built as either exact or approximate centered trees in which some distance measure from a single node is minimized [9, 1]. Generally, minimizing sums of distances leads to NP-complete problems such as the Steiner tree problem [10], while minimizing worst-case distances leads to tractable problems. In this paper, we address the problem of minimizing worst-case latencies in a general setting.

The network is modeled by a weighted undirected graph. Nodes represent locations of possible participants and the weights on the edges represent packet communication delays. We formulate the problem as follows: at a certain time we assume that a number of the network nodes, identified as the participants of a multipoint application, wish to establish a communication so that the maximum delay between any pair of participants is minimized. We further assume that the participants of the group are either sources of information, either sinks (receivers), or both, meaning that the groups of source and sink nodes are neither overlapping, nor disjoint. Only pairs of source and sink nodes contribute to the maximum end-to-end delay.

This problem can be solved in polynomial time. McMahan and Proskurowski [7] have independently established this via an $O(|V|^3 + |E||V|log|V|)$ algorithm. We present an algorithm with running time $O(|V|^3 + |E||V|)$ which provides a solution to the problem on general networks. Our algorithm returns a partial subtree of the network that includes at least all the participants of the multipoint group, such that the maximum distance between any source node and any sink node is the minimum possible.

The construction of spanning trees, or even partial spanning trees on graphs with respect to different constraints is an old problem and have been dealt with in different papers. Different measures of goodness have been studied under various restrictions for the network or the source and sink sets. In its general form, the construction of optimum communication spanning trees was initiated in [4]. In this paper, the problem was defined on the complete graph with a length and a requirement on the edges. The cost measure that had to be minimized was the sum of vertex distances weighted by the requirements between all vertices of the network. By setting the requirements equal to zero and parameterizing the number of source nodes, the problem was treated in [3] and was shown to be NP-complete. Some exact solutions were provided for the 2-source problem on restricted classes of graphs, such as unicycles and cactuses. Furthermore, the optimal solution to the 2-source problem on general graphs was approximated within a factor of 2. In an other recent paper [12] an approximation algorithm was provided for the all source problem. In [8] some heuristic algorithms were given for the minimum partial spanning trees taking into consideration the delay between a single source node and a group of sink (destination) nodes, while trying to bound the maximum difference in these delays.

The problem treated in this paper was proposed in [3] in the more restricted form, where the sink nodes were all the nodes of the network and by parameterizing the number of source nodes only. In [3], a pseudopolynomial algorithm was provided for this restricted version of the problem, and the weights on the edges had to be bound by the size of the graph. The 2-source problem on general graphs was efficiently approximated by a factor of 2. In this paper, we treat this problem in a much more general form, where the number of sink nodes is also parameterized. Moreover, the sets of sources and sinks are neither overlapping nor disjoint. We provided an innovative polynomial algorithm on general graphs. This problem is related to the selection of a subset of vertices in the graph to minimize a given cost function, or to the selection of p-centers [5].

The remaining of this paper is organized as follows: In section 2, we give a formal definition of the problem. In section 3, we show that an equivalent problem is to find a subdivision of a single edge that yields a spanning tree that minimizes distances to the point of the cut. This makes it easy to construct a minimum-eccentricity spanning tree once the correct edge has been identified. In section 4, we establish formulas and properties that the algorithm depends upon. In section 5 the actual algorithm is presented in detail and its complexity is analyzed. We conclude in section 6 with some suggestions for further research.

2 Network model and problem definition

In what follows the terms *vertex* and *node* are used interchangeably. Similarly for the terms *edge* and *link*.

A network is modeled as an undirected weighted graph $G = (V,E)$, where V is the set of network nodes and E is the set of edges. An edge connecting two nodes v_1 and v_2, denoted by (v_1, v_2), has an associated weight $\ell(v_1,v_2)$, which is a nonnegative real number, and represents the delay that data packets experience on this edge.

Consider a multipoint application that requires a set of source nodes S_1 to transmit data to a set of sink (destination) nodes S_2. In order for the communication to proceed, we need to establish paths between each pair of source/sink nodes. In general, and for efficiency reasons, multicast packets are routed through the links of a *multicast tree* that includes at least all nodes of $S_1 \cup S_2$. Nodes that do not belong in the multipoint application as sources or sinks can be used as relay nodes in the multicast tree. However, for obvious reasons, these nodes are never leaf nodes of the tree.

Let us suppose that T is the multicast tree of G used in a certain multipoint application with source nodes S_1 and sink nodes S_2. The total communication delay between a source node v_1 and a sink node v_2 using T is denoted by $d_T(v_1,v_2)$, which is the weighted distance between v_1 and v_2 in T. One objective that has to be taken into account during the construction of a multicast tree T is the maximum delay between any source and any sink node. This objective is directly related to the quality of service requirements of the multipoint application because packets delivered more than a certain number of time units later could be of no value to the receiver. The most desirable objective is to construct a multicast tree that minimizes the maximum source/sink delay. The maximum distance between any source and any sink node in a multicast tree is the *eccentricity* of this tree. More formally, we express the *Minimum Eccentricity Multicast Tree* problem as follows:

Problem: *Given a network $G = (V,E)$, a set of source nodes S_1, a set of sink nodes S_2, a link delay $\ell(v_1,v_2)$ for each edge (v_1,v_2). Assume that $\mathcal{T}$ is the set of all multicast trees of G that include at least the nodes $S_1 \cup S_2$. For each tree $T \in \mathcal{T}$ we define its eccentricity*

$$e(T) = \max_{s_1 \in S_1, s_2 \in S_2} d_T(s_1,s_2) \tag{1}$$

We wish to find the tree $T \in \mathcal{T}$ that has the minimum eccentricity

$$e(G) = \min_{T \in \mathcal{T}} e(T) \tag{2}$$

In this paper, we show that this problem is polynomial and we describe an efficient algorithm that constructs minimum eccentricity multicast trees on general networks.

Before we proceed to the description of the algorithms we need the following definitions and notation:

- For any subgraph H of G, $d_H(v_1,v_2)$ is the weighted distance between v_1 and v_2 in H.
- If P is a path between v_1 and v_2, then a vertex c of P is a *midpoint* if $D_P(v_1,c) = D_P(v_2,c)$.
- A minimum-distance spanning tree rooted at c is a spanning tree T in which $d_T(v,c) = d_G(v,c)$ for all $v \in V$.
- If T is a spanning tree, an S_i-critical path ($i = 1$ or $i = 2$) is a path in T between $v_1 \in S_i$ and $v_2 \in S_i$ such that $d_T(v_1,v_2) \geq d_T(v_1',v_2')$ for all $v_1' \in S_i$ and $v_2' \in S_i$.
- A *1-refinement* R of a weighted graph G is obtained by replacing an edge $E_R = (u,w)$ by two edges (u,c_R) and (w,c_R) where $c_R \notin V$ is a new vertex, with $\ell(u,c_R) + \ell(w,c_R) = \ell(u,w)$. The resulting graph is denoted G_R. If T is a spanning tree of G that contains E_R, then the 1-refinement can be viewed also as a 1-refinement T_R of T.

- $\mathcal{R}$ is the set of all 1-refinements of G.
- For any 1-refinement $R \in \mathcal{R}$,

$$d(R) = \max_{s \in S_1} d_{G_R}(s, c_R) + \max_{s \in S_2} d_{G_R}(s, c_R) \tag{3}$$

- $d(G) = \min_{R \in \mathcal{R}} d(R)$

3 Characterizing the solutions

In this section we show that a minimum eccentricity multicast tree is a partial shortest path tree rooted at some "point" in the network that includes at least all source and sink nodes. This point could be either a vertex, in the ideal case, or a point c_R lying on an edge, produced by an 1-refinement of the graph. This result will be used in the following section to derive a polynomial algorithm for this problem.

Lemma 1 *Let $R \in \mathcal{R}$. Let T be a minimum-distance spanning tree rooted at c_R. Then $e(T) \leq d(R)$.*

Proof

$$\begin{aligned} e(T) &= \max_{s_1 \in S_1, s_2 \in S_2} d_T(s_1, s_2) \\ &\leq \max_{s_1 \in S_1} d_T(s_1, c_R)) + \max_{s_2 \in S_2} d_T(s_2, c_R) \\ &\leq \max_{s_1 \in S_1} d_G(s_1, c_R)) + \max_{s_2 \in S_2} d_G(s_2, c_R) \\ &= d(R) \end{aligned}$$

□

Lemma 2 *Let $T \in \mathcal{T}$. Let R be a 1-refinement of T such that c_R is the midpoint of some S_i-critical path in T, for $i = 1$ or $i = 2$. Then $d(R) \leq e(T)$.*

Proof Assume w.l.o.g. that $i = 1$, and that the S_i critical path connects s with s'. Let $D = d_T(s, c_R) = d_T(s', c_R)$.

Claim (1): For any $s_1 \in S_1$, $d_G(s_1, c_R) \leq D$. If not, then

$$d_T(s_1, s') \geq d_T(s_1, c_R) + d_T(c_R, s') > 2D = d_T(s, s') \tag{4}$$

if the path in T from s_1 to s' passes through c_R or

$$d_T(s_1, v) \geq d_T(s_1, c_R) + d_T(c_R, s) > 2D = d_T(s, s') \tag{5}$$

if the path in T from s_1 to s passes through c_R. Either case contradicts the assumption that the (s, s')-path in T is an S_1-critical path.

Claim (2): For any $s_2 \in S_2$, $d_G(s_2, c_R) \le e(T) - D$. If not, then

$$d_T(s_2, s') \ge d_T(s_2, c_R) + d_T(c_R, s') > e(T) \tag{6}$$

if the path in T from s_2 to s' passes through c_R or

$$d_T(s_2, s) \ge d_T(s_2, c_R) + d_T(c_R, s) > e(T) \tag{7}$$

if the path in T from s_2 to s passes through c_R. Either of these contradicts the definition of $e(T)$.

Combining claims (1) and (2) yields $d(R) \le D + (e(T) - D) = e(T)$. □

Theorem 1 $d(G) = e(G)$.

Proof Let T be such that $e(T) = e(G)$ and let R' be a 1-refinement of T with $c_{R'}$ the midpoint of some S_i-critical path in T. By Lemma 2, $d(R') \le e(T)$. Thus $d(G) \le d(R') \le e(T) = e(G)$.
Let R be such that $d(R) = d(G)$ and let T' be a minimum-distance spanning tree rooted at c_R. By Lemma 1, $e(T') \le d(R)$. Thus $e(G) \le e(T') \le d(R) = d(G)$.
Combining these results yields $d(G) = d(R') = e(G) = e(T')$. □

Based on the previous theorem, we provide an upper bound for eccentricity.

Corollary 1 $e(G) \le 2\,\text{radius(G)}$, *where* $\text{radius}(G) = \min_{c \in V} \max_{v \in V} d_G(c, v)$.

Proof Let $R \in \mathcal{R}$ be determined by a vertex c_R for which $\max_{v \in V} d_G(c_R, v) = \text{radius}(G)$. Then $e(G) = d(G) \le d(R) \le 2\,\text{radius(G)}$. □

When an edge must be split into two nontrivial parts to obtain the minimum value of $d(R)$, $e(G)$ will be less than 2 radius(G). This is the case with the graph of Figure 2, where radius(G) = 25 (with w as center) and $e(G) = 42$ (with (u, w) split and $\ell(u, c_R) = 3$ and $\ell(w, c_R) = 9$).

The customary approach to forming a tree is to root it at a vertex of the given graph, rather than at a point created through the subdivision of an edge. The eccentricity of such a tree is bounded above by

$$\hat{d}(G) = \min_{v \in V} \left(\max_{s \in S_1} d_G(s, v) + \max_{s \in S_2} d_G(s, v) \right) \tag{8}$$

Theorem 1 allows us to determine in general that such a rooted tree is within one edge weight of being optimal:

Theorem 2 $e(G) \le \hat{d}(G) \le e(G) + \max_{(u,w) \in E} \ell(u, w)$.

Proof The first inequality is a consequence of the definitions. For the second, let c_R define a 1-refinement R such that $d(R) = d(G)$. Let the edge divided by c_R be (u, w) where $\ell(u, c_R) \le \ell(w, c_R)$.

Then by the triangle inequality, for all $v \in V$, $d_G(u, v) \le \ell(u, c_R) + d_{G_R}(v, c_R)$, and thus for any $s_1 \in S_1$ and $s_2 \in S_2$,

$$d_G(s_1, u) + d_G(s_2, u) \le d_{G_R}(s_1, c_R) + d_{G_R}(s_2, c_R) + 2\ell(u, c_R). \tag{9}$$

This implies $d_G(s_1, u) + d_G(s_2, u) \le d(G) + \ell(u, w)$. □

The upper bound is tight, as the graph in Figure 1 illustrates, where S_1 comprises the two degree-1 vertices and S_2 the two degree-2 vertices. There are only eight possible spanning trees, so it is not difficult to find the optimal trees by exhaustive checking. The optimal vertex-rooted tree is rooted at one of the degree-4 vertices, uses at least one edge weighted $L-\varepsilon$, and has eccentricity $3L-\varepsilon$. The optimal edge-rooted tree is based on bisecting the edge in the middle of the diagram, does not use either edge weighted $L-\varepsilon$, and has eccentricity $2L+2\varepsilon$.

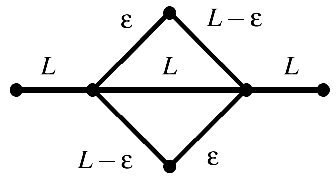

Fig. 1: The optimal edge-rooted tree has eccentricity $2L+2\varepsilon$. The optimal vertex-rooted tree has eccentricity $3L-\varepsilon$.

4 Basis for an algorithm

The algorithm focuses on identifying the appropriate edge (u,w) that can be cut to create a "center" c_R from which to construct a minimum-distance spanning tree. This edge is found by examining for each edge (u,w) the longest weighted distances from u and w to members of S_1 and S_2. The constructions in this section will be illustrated using the example in Figure 2.

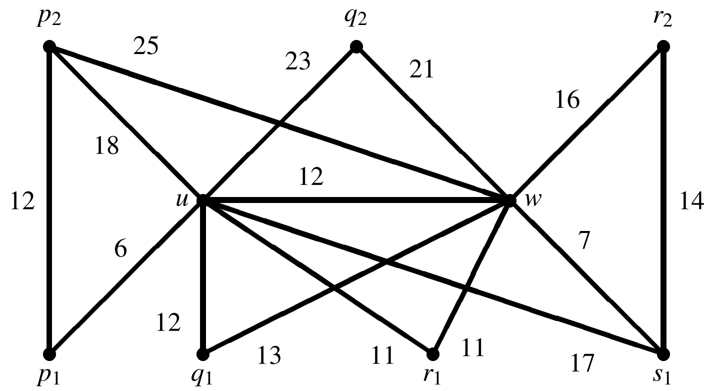

Fig. 2: An example problem graph.

The algorithm is mostly easily understand with reference to some geometric relationships in the xy-plane. If (a,b) is a point in the xy-plane, denote by $Q(a,b)$ the open quarter plane to the upper-right of (a,b); i.e., $Q(a,b)=\{(x,y)\mid x>a \text{ and } y>b\}$. Given a fixed reference point (a_0,b_0), denote by $L^{(a_0,b_0)}(a,b)$ the closed rectangle to the lower left of (a,b); i.e.,

$$L^{(a_0,b_0)}(a,b)=\{(x,y)\mid a_0\le x\le a \text{ and } b_0\le y\le b\}. \tag{10}$$

If $a_0>a$ or $b_0>b$, $L^{(a_0,b_0)}(a,b)$ is empty.

Given an edge (u,w), for any $v \in V$, the point $(d_G(v,u), d_G(v,w))$ lies in the first quadrant of the xy-plane. For $i = 1$ or $i = 2$, consider the sets of such points $\mathcal{D}_i^{uw} = \{(d_G(s,u), d_G(s,w)) \mid s \in S_i\}$. In the example,

$$\begin{aligned}\mathcal{D}_1^{uw} &= \{(6,18),(12,13),(11,11),(17,7)\}\\ \mathcal{D}_2^{uw} &= \{(18,25),(23,21),(28,16)\}.\end{aligned}$$

For a given edge (u,w) and set S_i, the fixed reference point $(a_{i0}^{uw}, b_{i0}^{uw})$ is used, where

$$a_{i0}^{uw} = \max_{s \in S_i} d_G(s,w) - \ell(u,w) \tag{11}$$

$$b_{i0}^{uw} = \max_{s \in S_i} d_G(s,u) - \ell(u,w) \tag{12}$$

In the example, $(a_{10}^{uw}, b_{10}^{uw}) = (6,5)$. Denote by L_i^{uw} the union of all the closed rectangles determined by points in $\mathcal{D}_i^{uw}$; that is,

$$L_i^{uw} = \bigcup_{(a,b) \in \mathcal{D}_i^{uw}} L^{(a_{i0}^{uw}, b_{i0}^{uw})}(a,b) \tag{13}$$

Denote by Q_i^{uw} the complement of L_i^{uw} with respect to the upper-right quarter plane $Q(a_{i0}^{uw}, b_{i0}^{uw})$, that is,

$$Q_i^{uw} = Q(a_{i0}^{uw}, b_{i0}^{uw}) \setminus L_i^{uw} \tag{14}$$

Figure 3 illustrates these constructions in the case of the graph from Figure 1, for edge (u,w) and S_1. The points of $\mathcal{D}_1^{uw}$ are marked as p_1, q_1, r_1, s_1. Although it does not happen in this example, in general a_{i0}^{uw} and/or b_{i0}^{uw} may be negative, and some rectangles in $\mathcal{D}_i^{uw}$ may be empty. The rectangles for vertices at maximum distance from u or w are always nonempty.

The essential property of Q_i^{uw} is described in the following lemma.

Lemma 3 *$Q(a,b) \subseteq Q_i^{uw}$ if and only if for every vertex $v \in S_i$, either $d_G(v,u) \le a$ or $d_G(v,w) \le b$.*

Proof If $d_G(v,u) > a$ and $d_G(v,w) > b$, then $(d_G(v,u), d_G(v,w)) \in Q(a,b)$. But from the definition, $(d_G(v,u), d_G(v,w)) \in L_i^{uw}$. This is impossible since $Q_i^{uw} \bigcap L_i^{uw} = \emptyset$. □

Since L_i^{uw} is the union of a finite number of closed rectangles all of which have $(a_{i0}^{uw}, b_{i0}^{uw})$ as lower left corner, the region Q_i^{uw} is the union of a finite number of upper-right quarter planes. Let $\mathcal{Q}_i^{uw}$ be the collection of quarter planes $Q(a,b)$ that are maximal in Q_i^{uw}. Clearly, $\mathcal{Q}_i^{uw}$ comprise a minimal finite set of upper-right quarter planes whose union is Q_i^{uw}, and if $Q(a,b) \subseteq Q_i^{uw}$, then $Q(a,b)$ is a subset of some member of $\mathcal{Q}_i^{uw}$. It is relatively easy to see from Figure 3 that for our example a minimal set of upper-right quarter planes whose union is Q_1^{uw} is the following

$$\mathcal{Q}_1^{uw} = \{Q(a_1,b_1), Q(a_2,b_2), Q(a_3,b_3)\}. \tag{15}$$

An essential property of $\mathcal{Q}_i^{uw}$ is the following:

Lemma 4 *Let $Q(a,b) \in \mathcal{Q}_i^{uw}$. If $a \ne a_{i0}^{uw}$ and $b \ne b_{i0}^{uw}$, then for some $u' \in V$, $d_G(u',u) = a$ and $d_G(u',w) > b$, and for some $w' \in V$, $d_G(w',w) = b$ and $d_G(w',u) > a$.*

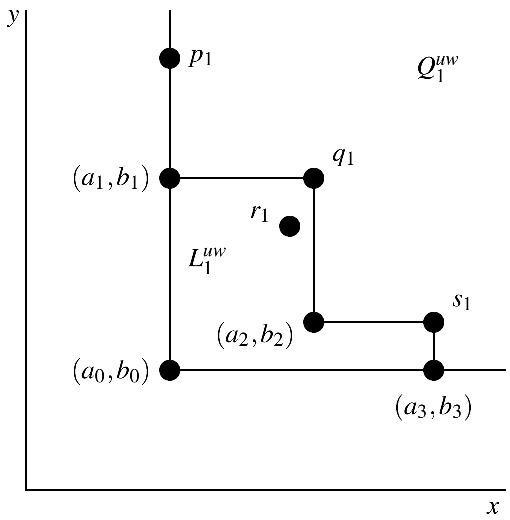

Fig. 3: The geometric interpretation for edge (u, w) and S_1.

Proof Consider all vertices u' for which $d_G(u', w) > b$. This set is nonempty because otherwise $Q(a_{i0}^{uw}, b) \subseteq Q_i^{uw}$ which is impossible because $Q(a, b)$ is maximal and $a \neq a_{i0}^{uw}$. For all such u', $d_G(u', u) \leq a$ by Lemma 3. Now choose u' to maximize $d_G(u', u)$. If then $d_G(u', u) = a' < a$, $Q(a', b) \subseteq Q_i^{uw}$ contradicting the maximality of $Q(a, b)$. Thus $d_G(u', u) = a$. The proof for w' is analogous. □

Let the members of Q_i^{uw} be

$$Q(a_{i1}^{uw}, b_{i1}^{uw}), Q(a_{i2}^{uw}, b_{i2}^{uw}), \ldots, Q(a_{iN_i^{uw}}^{uw}, b_{iN_i^{uw}}^{uw}) \tag{16}$$

arranged so that

$$a_{i1}^{uw} < a_{i2}^{uw} < \cdots < a_{iN_i^{uw}}^{uw}. \tag{17}$$

Since no member of Q_i^{uw} can contain any other, all the values a_{ik}^{uw} are distinct, all the values b_{ij}^{uw} are distinct, and it must be the case that

$$b_{i1}^{uw} > b_{i2}^{uw} > \cdots > b_{iN_i^{uw}}^{uw}. \tag{18}$$

Geometrically, these properties reflect that fact that the points $(a_{ik}^{uw}, b_{ik}^{uw})$ are the lower-left-hand corners that occur on the boundary between L_i^{uw} and Q_i^{uw}. In our example (see Figure 3),

$$Q_1^{uw} = \{Q(a_1, b_1), Q(a_2, b_2), Q(a_3, b_3)\} = \{Q(6, 13), Q(12, 7), Q(17, 5)\}. \tag{19}$$

The lower-left-hand corners that occur on the boundary between L_i^{uw} and Q_i^{uw} have the property described in lemma 4. Intuitively, this property can be explained as follows: for a certain edge (u, w) of the graph and for a subset of nodes S_1, a point $(a_{1k}^{uw}, b_{1k}^{uw})$ means that the nodes of S_1 are at maximum distance a_{1k}^{uw} from u or at maximum distance b_{1k}^{uw} from w. Furthermore, there is at least one node at distance a_{1k}^{uw} from u whose distance from w is at least b_{1k}^{uw}, and the other way around (there is a node at distance b_{1k}^{uw} from w whose distance from u is at least a_{1k}^{uw}).

The algorithm is based on a linear scan of the values $(a_{ik}^{uw}, b_{ik}^{uw})$. The following lemmas establish the properties that enable this.

Lemma 5

(a) $a_{i1}^{uw} = a_{i0}^{uw}$;

(b) $b_{i1}^{uw} \leq \max_{v \in S_i} d_G(v,w)$;

(c) $a_{iN_i^{uw}}^{uw} \leq \max_{v \in S_i} d_G(v,u)$;

(d) $b_{iN_i^{uw}}^{uw} = b_{i0}^{uw}$.

Proof

(a) Obviously $Q(a_{i0}^{uw}, \max_{v \in S_i} d_G(v,w)) \subseteq Q_i^{uw}$. Since $a_{i0}^{uw} \leq a_{i1}^{uw} < a_{i2}^{uw} < \cdots < a_{iN_i^{uw}}^{uw}$, the member of Q_i^{uw} that contains $Q(a_{i0}^{uw}, \max_{v \in S_i} d_G(v,w))$ must be $Q(a_{i1}^{uw}, b_{i1}^{uw})$ and this implies $a_{i1}^{uw} = a_{i0}^{uw}$.

(b) Let b' be the maximum of $d_G(v,w)$ for all vertices v such that $d_G(v,u) > a_{i0}^{uw}$. Then $Q(a_{i0}^{uw}, b') \subseteq Q_i^{uw}$ implies $Q(a_{i0}^{uw}, b') \subseteq Q(a_{i1}^{uw}, b_{i1}^{uw})$ and thus $b_{i1}^{uw} \leq b'$ which implies the result.

(c) and (d) are analogous to (a) and (b) respectively. □

Lemma 6 *For any* (u,w), $i \in \{1,2\}$, *and* $0 \leq k \leq N_i^{uw}$, $\left|a_{ik}^{uw} - b_{ik}^{uw}\right| \leq \ell(u,w)$.

Proof By symmetry, assume that $b_{ik}^{uw} \geq a_{ik}^{uw}$.

For $k = 0$, the result follows from the definitions and the fact that a path to u or w can be extended using the edge (u,w).

For $k = 1$, Lemma 5 implies

$$b_{i1}^{uw} - a_{i1}^{uw} \leq \max_{v \in S_i} d_G(v,w) - a_{i0}^{uw} = \ell(u,w). \tag{20}$$

For $k = N_i^{uw}$,

$$b_{iN_i^{uw}}^{uw} - a_{iN_i^{uw}}^{uw} \leq b_{i0}^{uw} - a_{i0}^{uw} \leq \ell(u,w), \tag{21}$$

by Lemma 5 and the fact that a_{ik}^{uw} is an increasing function of k.

For $1 < k < N_i^{uw}$, suppose by way of contradiction that $b_{ik}^{uw} - a_{ik}^{uw} > \ell(u,w)$. Let u' be as in Lemma 4. Then

$$d_G(u',w) - d_G(u',u) \geq b_{ik}^{uw} - a_{ik}^{uw} > \ell(u,w) \tag{22}$$

contradicting the existence of a path from u' through u to w of length $d_G(u',u) + \ell(u,w)$.

□

Lemma 7 *Let (u,w) be any edge and let $1 \le k \le N_1^{uw}$ and $1 \le j \le N_2^{uw}$. Then there exists a spanning tree T such that*

$$e(T) \le \max(a_{1k}^{uw} + \ell(u,w) + b_{2j}^{uw}, a_{2j}^{uw} + \ell(u,w) + b_{1k}^{uw}) \tag{23}$$

Proof Consider the line in the xy-plane through $(a_{1k}^{uw}, b_{1k}^{uw})$ with slope 1. It intersects the boundary of Q_2^{uw} in a point $(a,b) = (a_{1k}^{uw} + \lambda, b_{1k}^{uw} + \lambda)$. Since the line has slope 1, $b - a = b_{1k}^{uw} - a_{1k}^{uw}$. Since $Q(a,b) \subseteq Q_2^{uw}$, $d_G(v,u) \le a$ or $d_G(v,w) \le b$ for every $v \in S_2$. Let the 1-refinement R be determined by replacing (u,w) with (u,c_R) and (w,c_R), where

$$\begin{aligned} \ell(u,c_R) &= \frac{1}{2}\ell(u,w) - \frac{1}{2}(a_{1k}^{uw} - b_{1k}^{uw}) \\ \ell(w,c_R) &= \frac{1}{2}\ell(u,w) + \frac{1}{2}(a_{1k}^{uw} - b_{1k}^{uw}) \end{aligned} \tag{24}$$

Note that Lemma 6 ensures that $\ell(u,c_R)$ and $\ell(w,c_R)$ are nonnegative.

Let $s \in S_1$. If $d_G(s,u) \le a_{1k}^{uw}$, then

$$\begin{aligned} d_{G_R}(s,c_R) &\le d_G(s,u) + d_{G_R}(u,c_R) \\ &\le \frac{1}{2}\ell(u,w) + \frac{1}{2}a_{1k}^{uw} + \frac{1}{2}b_{1k}^{uw}. \end{aligned} \tag{25}$$

Otherwise, $d_G(s,w) \le b_{1k}^{uw}$ which yields

$$\begin{aligned} d_{G_R}(s,c_R) &\le d_G(s,w) + d_{G_R}(w,c_R) \\ &\le \frac{1}{2}\ell(u,w) + \frac{1}{2}a_{1k}^{uw} + \frac{1}{2}b_{1k}^{uw}. \end{aligned} \tag{26}$$

Similarly, for $v \in S_2$, if $d_G(v,u) \le a$, then

$$\begin{aligned} d_{G_R}(v,c_R) &\le d_G(v,u) + d_{G_R}(u,c_R) \\ &\le a + \frac{1}{2}\ell(u,w) + \frac{1}{2}(b_{1k}^{uw} - a_{1k}^{uw}) \\ &= a + \frac{1}{2}\ell(u,w) + \frac{1}{2}(b-a). \end{aligned} \tag{27}$$

Otherwise, $d_G(v,w) \le b$, so that

$$\begin{aligned} d_{G_R}(v,c_R) &\le d_G(v,w) + d_{G_R}(w,c_R) \\ &\le b + \frac{1}{2}\ell(u,w) - \frac{1}{2}(b_{1k}^{uw} - a_{1k}^{uw}) \\ &= b + \frac{1}{2}\ell(u,w) - \frac{1}{2}(b-a). \end{aligned} \tag{28}$$

Combining these results gives

$$\begin{aligned} d(R) &\le \frac{1}{2}(a_{1k}^{uw} + b_{1k}^{uw}) + \ell(u,w) + \frac{1}{2}(a+b) \\ &= \ell(u,w) + a_{1k}^{uw} + b_{1k}^{uw} + \lambda. \end{aligned} \tag{29}$$

The boundary of Q_2^{uw} is such that for (a,b) and $(a_{2j}^{uw}, b_{2j}^{uw})$, as for any two points on the boundary, if $a_{2j}^{uw} < a$, then $b_{2j}^{uw} \geq b$. If $a_{2j}^{uw} \geq a$, then $a_{1k}^{uw} + \lambda \leq a_{2j}^{uw}$ and $d(R) \leq \ell(u,w) + a_{2j}^{uw} + b_{1k}^{uw}$. Otherwise, $b_{2j}^{uw} \geq b$, so that $b_{1k}^{uw} + \lambda \leq b_{2j}^{uw}$ and $d(R) \leq \ell(u,w) + a_{1k}^{uw} + b_{2j}^{uw}$. Thus

$$d(R) \leq \max(a_{1k}^{uw} + \ell(u,w) + b_{2j}^{uw}, a_{2j}^{uw} + \ell(u,w) + b_{1k}^{uw}). \tag{30}$$

The result now follows from Lemma 1. □

Lemma 8 *Let T be a minimum-eccentricity spanning tree. Let u', w' be the endpoints of any S_i-critical path in T, and let (u,w) be an edge in that path such that $d_T(u',u) \leq \frac{1}{2} d_T(u',w')$ and $d_T(w,w') \leq \frac{1}{2} d_T(u',w')$. (There are either one or two such edges.) Then for some $Q(a_{1k}^{uw}, b_{1k}^{uw}) \in Q_1^{uw}$ and $Q(a_{2j}^{uw}, b_{2j}^{uw}) \in Q_2^{uw}$,*

$$e(T) \geq \max(a_{1k}^{uw} + \ell(u,w) + b_{2j}^{uw}, a_{2j}^{uw} + \ell(u,w) + b_{1k}^{uw}) \tag{31}$$

Proof Without loss of generality, assume $i = 1$. If the edge (u,w) is deleted from T, two trees are produced; denote by T_u the one containing u and by T_w the one containing w. Because (u,w) lies in an S_1-critical path, both $S_1 \bigcap T_u$ and $S_1 \bigcap T_w$ are nonempty. Let $u_1 \in S_1 \cap T_u$ be chosen to maximize the value $d_G(u_1,u)$. Let $w_1 \in S_1 \cap T_w$ be chosen to maximize the value $d_G(w_1,w)$. Then for every $v \in S_1$, either

$$d_G(v,u) \leq d_G(u_1,u) \leq d_T(u_1,u) \tag{32}$$

or

$$d_G(v,w) \leq d_G(w_1,w) \leq d_T(w_1,w). \tag{33}$$

This means that $Q(d_T(u_1,u), d_T(w_1,w)) \subseteq Q_1^{uw}$. Thus

$$Q(d_T(u_1,u), d_T(w_1,w)) \subseteq Q(a_{1k}^{uw}, b_{1k}^{uw}) \in Q_1^{uw} \tag{34}$$

for some k, which means $a_{1k}^{uw} \leq d_T(u_1,u)$ and $b_{1k}^{uw} \leq d_T(w_1,w)$.

Now suppose both $S_2 \cap T_u$ and $S_2 \cap T_w$ are nonempty. Proceeding in the same fashion, we select vertices $u_2 \in S_2 \cap T_u$ and $w_2 \in S_2 \cap T_w$ such that

$$Q(d_T(u_2,u), d_T(w_2,w)) \subseteq Q(a_{2j}^{uw}, b_{2j}^{uw}) \in Q_2^{uw} \tag{35}$$

for some j, so that $a_{2j}^{uw} \leq d_T(u_2,u)$ and $b_{2j}^{uw} \leq d_T(w_2,w)$. Then

$$\begin{aligned} e(T) &\geq d_T(u_1,w_2) = d_T(u_1,u) + \ell(u,w) + d_T(w,w_2) \\ &\geq a_{1k}^{uw} + \ell(u,w) + b_{2j}^{uw}. \end{aligned} \tag{36}$$

Similarly,

$$\begin{aligned} e(T) &\geq d_T(u_2,w_1) = d_T(u_2,u) + \ell(u,w) + d_T(w,w_1) \\ &\geq a_{2j}^{uw} + \ell(u,w) + b_{1k}^{uw}. \end{aligned} \tag{37}$$

This establishes the lemma in this case.

For the case when $S_2 \cap T_u = \emptyset$, select u_1, w_1, and w_2 and before, let $j = 1$, and note in this case that $d_G(w, w_2) = \max_{v \in S_2} d_G(w, v) \geq b_{21}^{uw}$. As before,

$$\begin{aligned} e(T) &\geq d_T(u_1, w_2) = d_T(u_1, u) + \ell(u, w) + d_T(w, w_2) \\ &\geq a_{1k}^{uw} + \ell(u, w) + b_{21}^{uw}. \end{aligned} \tag{38}$$

Now the hypothesis that $d_T(w, w_1) \leq \frac{1}{2} d_T(u_1, w_1)$ implies $d_T(u_1, w) \geq d_T(w, w_1)$. Thus

$$\begin{aligned} e(T) &\geq d_T(u_1, w_2) = d_T(u_1, w) + d_T(w, w_2) \\ &\geq d_T(w, w_1) + d_T(w, w_2) \\ &\geq b_{1k}^{uw} + d_G(w, w_2) \\ &= b_{1k}^{uw} + \ell(u, w) + a_{20}^{uw} \\ &= b_{1k}^{uw} + \ell(u, w) + a_{21}^{uw}. \end{aligned} \tag{39}$$

The case when $S_2 \cap T_w = \emptyset$ is similar, with $j = N_2^{uw}$, in light of the symmetrical relationship between a and b. □

Theorem 3

$$e(G) = \min_{(u,w),k,j} \max(a_{1k}^{uw} + \ell(u, w) + b_{2j}^{uw}, a_{2j}^{uw} + \ell(u, w) + b_{1k}^{uw}) \tag{40}$$

Proof The proof is immediate from Lemmas 7 and 8. □

5 The algorithm

Moving from Theorem 3 to an explicit algorithm is mostly straightforward. One fine point is that instead of evaluating $a_{1k}^{uw} + b_{2j}^{uw}$ and $a_{2j}^{uw} + b_{1k}^{uw}$ for all combinations of k and j, a single pass can be made from $k = 1$, $j = 1$ to $k = N_1^{uw}$, $j = N_2^{uw}$ since the coordinates are ordered. Calculating the values a_{ik}^{uw} and b_{ik}^{uw} is most easily understood in terms of finding the lower-left-hand corners in the boundary between L_i^{uw} and Q_i^{uw}, which is formed from horizontal and vertical line segments. The method used in Figure 5 is but one of several possible ways to solve this geometric problem.

The algorithm is shown in Figures 4 and 5. First, the weighted distances between all pairs of points are calculated. Second, for each vertex v, a list $H_0[v]$ is created of all vertices in order of increasing distance from v. Third, by selecting just members of S_i from $H_0[v]$, lists $H_1[v]$ and $H_2[v]$ are created in which the members of S_1 and S_2, respectively, are listed in order of increasing distance from v. Fourth, for each edge (u, w), arrays A_i and B_i are created such that $A_i[k] = a_{ik}^{uw}$ and $B_i[k] = b_{ik}^{uw}$ for $1 \leq k \leq N_i^{uw}$ and $i = 1, 2$. The sizes of these arrays are also recorded in N_i. Finally, the arrays A_i and B_i are scanned in one pass to seek the minimum described in Theorem 3.

The first two steps rely only on the edge weights in the graph, and not on S_1 and S_2. In practice, the network topology can be considered fixed, so the first two steps can be precomputed. These two offline steps can be accomplished in $O(|V|^3)$ steps using Dijkstra's algorithm [11]. The running time of step 3 is $O(|V|^2)$. The running time of steps 5.1 and 5.3 are each $O(|S_1 + S_2|)$, so the time for step 5 is

0. Initially, $L[x][y]$ is the weight of edge (x,y) and Z_i is the size of S_i.

1. Construct D so that $D[x][y] = d_G(x,y)$.

2. For each vertex v, construct $H_0[v]$ so that $H_0[v][k]$ for $k = 1,2,3,\ldots$ is a list of all vertices in order of increasing distance from v.

3. For each vertex v and for $i = 1,2$, construct $H_i[v]$ so that $H_i[v][k]$ for $k = 1,2,3,\ldots$ is a list of the members of S_i in order of increasing distance from v.

4. $e \leftarrow \infty$.

5. For each edge (u,w):

- **5.1.** Calculate A_1, B_1, A_2, and B_2. (See Figure 5.)
- **5.2.** $k \leftarrow 1$; $j \leftarrow 1$;
- **5.3.** While ($k \leq N_1$ **and** $j \leq N_2$):
  - **5.3.1.** $c \leftarrow A_1[k] + L[u][w] + B_2[j]$;
  - **5.3.2.** $d \leftarrow A_2[j] + L[u][w] + B_1[k]$;
  - **5.3.3.** If $(c > d)$:
    - **5.3.3.1.** If $(c < e)$ then $e \leftarrow c$;
    - **5.3.3.2.** $j \leftarrow j+1$;
  - **5.3.4.** Else:
    - **5.3.4.1.** If $(d < e)$ then $e \leftarrow d$;
    - **5.3.4.2.** $k \leftarrow k+1$;
    - **5.3.4.3.** If $(c = d)$ then $j \leftarrow j+1$;

Fig. 4: The algorithm.

$O(|E|\,|S_1 + S_2|)$. Thus the offline portion of the algorithm has time complexity $O(|V|^3)$ and the online portion $O(|E|\,|S_1+S_2|) \leq O(|V|\,|E|) \leq O(|V|^3)$.

Except for steps 5.1 and 5.3, the correctness of the algorithm should be patently clear. It is a relatively straightforward problem to scan the sorted arrays A_i and B_i in parallel seeking the minimum value of $\max(A_1[k]+B_2[j], A_2[j]+B_1[k])$; a detailed argument is presented below that step 5.3 correctly finds this minimum. It is also a relatively straightforward problem to find the lower-left-hand corners of a nonincreasing curve in the xy-plane composed of horizontal and vertical line segments; a detailed argument is presented below that step 5.1 correctly finds these corners.

Step 5.3 makes a single pass in parallel through A_i and B_i, rather than checking all possible pairs. This is possible because a_{ik}^{uw} is an increasing function of k and b_{ij}^{uw} is a decreasing function of j. It is straightforward to verify the loop invariant property that at statement 5.3, $e \leq \min_{j'<j \text{ or } k'<k} \max(A_1[k']+B_2[j'], A_2[j']+B_1[k']$.

Step 5.1 is the heart of the algorithm. It searches the boundary between L_i^{uw} and Q_i^{uw} working from upper left to lower right. Step 5.1.2 stores a large number in $B_i[0]$ just to ensure that the condition is true

5.1.1. $A_i[0] \leftarrow D[H_i[w][Z_i]][w] - L[u][w]$;

5.1.2. $B_i[0] \leftarrow D[H_i[w][Z_i]][w] + 1$;

5.1.3. $k \leftarrow 0$;

5.1.4. $A_i[1] \leftarrow A_i[0]$;

5.1.5. $p \leftarrow Z_i$;

5.1.6. While $(p \geq 1)$:

- **5.1.6.1.** $v \leftarrow H_i[w][p]$;
- **5.1.6.2.** If $(D[v][u] > A_i[k+1])$:
  - **5.1.6.2.1.** If $(D[v][w] < B_i[k])$ then $\{k \leftarrow k+1; B_i[k] \leftarrow D[v][w]\}$
  - **5.1.6.2.2.** $A_i[k+1] \leftarrow D[v][u]$;
- **5.1.6.3.** $p \leftarrow p - 1$;

5.1.7. $B_i[0] \leftarrow D[H_i[u][Z_i]][u] - L[u][w]$;

5.1.8. If $(k = 0$ **or** $B_i[k] > B_i[0])$ then $\{k \leftarrow k+1; B_i[k] \leftarrow B_i[0]\}$

5.1.9. $N_i \leftarrow k$;

Fig. 5: Calculating A_i and B_i.

in the first execution of 5.1.6.2.1, thus simplifying the organization of the loop. Final values are assigned to $A_i[0]$, $A_i[1]$, and $B_i[0]$, in steps 5.1.1, 5.1.4, and 5.1.7, respectively.

Loop 5.1.6 works through the members of S_i in order of decreasing distance from w. Within the loop, $A_i[k]$ and $B_i[k]$ (except for $B_i[0]$) have been given their final values and $A_i[k+1]$ and $B_i[k+1]$ are being updated. Since the vertices are visited in order of decreasing distance from w, and since they are all within L_i^{uw}, vertex v satisfies one of three conditions: (i) v lies on the horizontal line through $(a_{ik}^{uw}, b_{ik}^{uw})$; (ii) otherwise, v lies on or to the left of the vertical line through $(a_{i,k+1}^{uw}, b_{i,k+1}^{uw})$; (iii) otherwise, v lies on the horizontal line segment to the right of $(a_{i,k+1}^{uw}, b_{i,k+1}^{uw})$. In the first case, step 5.1.6.2.2 responds to the fact that $a_{i,k+1}^{uw} \geq d_G(v,u)$. In the third case, steps 5.1.6.2.1 and 5.1.6.2.2 finalize the value of $b_{i,k+1}^{uw}$, initialize $a_{i,k+2}^{uw}$, and move k forward. It is easy to see that for every lower-left corner point $(a_{ik}^{uw}, b_{ik}^{uw})$, condition (iii) will occur exactly once and (i) and (iii) together will occur once for each vertex on the line segment $b_{ik}^{uw} = y, a_{ik}^{uw} \leq x$.

Thus the coordinates of all lower left corners will be left in A_i and B_i by the time the loop terminates, with the possible exception of $k = 1$ and $k = N_i^{uw}$ which depend on the entry and exit conditions of the loop. It is easy to see that $k = 1$ is handled properly and that $k = N_i^{uw}$ is handled properly when there is a vertex v with $d_G(v,w) = b_{i0}^{uw}$. Step 5.1.8 tests for this last situation, adding the final lower-left corner when necessary.

6 Conclusions

We presented an algorithm for the construction of a minimum eccentricity multicast tree on general networks. Further work would naturally include a distributed version of the algorithm. Furthermore, it would be interesting to incorporate other measures of goodness such as the minimization of the total cost of the multicast tree. This would turn our problem into an NP-complete problem that requires special treatment though heuristic/approximation algorithms.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT), an Architecture for Scalable Inter-domain Multicast Routing," *Proceedings of the ACM Conference on Communications Architectures, Protocols and Aplications (SIGCOMM'93)*, (1993), 85–94.

[2] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An Architecture for Wide-area Multicast Routing," *Proceedings of the ACM Conference on Communications Architectures, Protocols and Aplications (SIGCOMM'94)*, (1994), 126–135.

[3] A.M. Farley, P. Fragopoulou, D. Krumme, A. Proskurowski, and D. Richards, "Multi-source Spanning Tree Problems", *Journal of Interconnection Networks*, 1 (2000) 61-71.

[4] T.C. Hu, "Optimum Communication Spanning Trees", *SIAM Journal on Computing*, 3 (1974), 188-195.

[5] O. Kariv and S.L. Hakimi, "An algorithmic Approach to network location problems, I: the *p*-centers", *SIAM Journal on Applied Mathematics*, 37 (1979), 513.

[6] D.W. Krumme, *A Program that Finds a Minimum Eccentricity Multicast Tree*, Tufts Univ. EECS Dept. Technical Report 99-2, Nov. 1999, http://www.cs.tufts.edu/TR/.

[7] B. McMahan and A. Proskurowski, "Multi-source spanning trees: Algorithms for minimizing source eccentricities", submitted for publication.

[8] G.N. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints", *IEEE Journal on Selected Areas in Communications*, 15(3) (1997), 346-356.

[9] D.W. Wall, *Mechanisms for Broadcast and Selective Broadcast*, Ph.D. thesis, Stanford University, June 1980.

[10] B.M. Waxman, "Routing of Multipoint Connections", *IEEE Journal on Selected Areas in Communications* 6 (1988), 1617–1622.

[11] D.B. West, *Introduction to Graph Theory*. Prentice Hall (1996).

[12] 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", *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, (1998), 21-32.