Journal of Interconnection Networks

MULTI-SOURCE SPANNING TREE PROBLEMS

ARTHUR M. FARLEY
Computer and Information Science Department, University of Oregon, Eugene, OR 97403 USA

PARASKEVI FRAGOPOULOU
Laboratoire de Recherche en Informatique Université Paris Sud *

DAVID KRUMME
Computer Science Department, Tufts University Medford, MA 02155

ANDRZEJ PROSKUROWSKI
Computer and Information Science Department, University of Oregon Eugene, OR 97403 USA

DANA RICHARDS
Computer Science Department, George Mason University 400 University Drive, Fairfax, VA 22030-4444

Received (received date)
Revised (revised date)

We consider a model of multicast communication in a network whereby multiple sources have messages to disseminate among all sites of a network. We propose that the messages from all sources are disseminated along the same spanning tree of the network and consider the problem of constructing an optimal such tree. One measure for suitability of the construction is the sum of distances from all sources to all other vertices. We show that finding the exact solution in this case is $\mathcal{NP}$-hard (in the strong sense). We then investigate solutions for some restricted classes of graphs and give efficient algorithms for those. We also consider an alternative measure of goodness for the spanning tree, being the maximum eccentricity of a source. We show that the problem of finding such a a minimum eccentricity spanning tree is somewhat easier to solve and give a pseudo-polynomial solution algorithm.

Keywords: communication models, multicast, spanning tree, optimization problems.

1. Introduction

In computer networks, including the Internet, an increasingly prominent communication paradigm is *multicast*, whereby any of a set of network vertices called

*Present address: Laboratoire de recherche en informatique PRISM, Université de Versailles St-Quentin en Yvelines, 45 Avenue des Etas-Unis, Versaille CEDEX, France.

senders broadcasts its message to a set of multicast group *members*. In current Internet practice, the multicast protocol may require building an "optimal" routing tree from each of the senders to all group members; this protocol suffers an explosive growth of resource usage with the network size. We will discuss possible modifications to the above multicast protocol, whereby all senders utilize the same "shared" routing tree.

We model a network as a simple, undirected graph without loops, $G = \langle V, E\rangle$, where V is the set of *vertices* and $E \subseteq V \times V$ is the set of *edges*. There is a *length function* defined for the edges, $\ell : E \to Z^+$. We will refer to the number $N = |V|$ of vertices as the *size* of the graph G. We consider classes of multicast applications that restrict the sets of senders and receivers. For the purpose of our discussion, we will assume that the group membership includes all N vertices of the network (modeled as a simple undirected graph) and that some k of those ($1 \le k \le N$) are designated as sources. We will also assume that all group communications take place along edges of a *spanning tree* of the network, *i.e.*, a connected acyclic subgraph of the network graph that includes all vertices of the network.

There are several extant measures of goodness of a spanning tree, all with respect to a possible length function on the edges of the graph. One measure involves a simple sum of lengths of all edges of the spanning tree, resulting in the so called *minimum spanning tree*. Another takes into consideration the *distance* between two vertices of the graph, i.e., the sum of the lengths of edges along the path between the two vertices in a spanning tree. For a single source, this defines a *single-source shortest-paths tree*. Because of the attributes and requirements of multicast applications, several other measures come to mind. We consider two cost measures: *sum-cost*, being the sum of distances from all sources to all vertices, and *max-cost*, being the maximum distance from a source to another vertex. We will discuss the complexity of the problems as parameterized by the number k of sources ($1 \le k \le N$).

2. Problem definitions

We first define a family of decision problems called *k-Source Shortest-Paths Spanning Tree (k-SPST)*, parameterized by a positive integer k.

k-SPST:

Instance: A graph $G = \langle V, E\rangle$ with length function ℓ, k sources $s_1, \ldots, s_k \in V$, and positive integer K.

Question: Is there a spanning tree T of G whose sum-cost does not exceed K?

For $k = 1$, this problem becomes the single-source shortest-path problem, with well-known, efficient solution algorithms (for instance, the Dijkstra-Prim algorithm). For the set of sources identical with the set of vertices of the instance graph, we have a problem whose uniform edge lengths version has been defined before as *Shortest Total Path Length Spanning Tree* (*cf.* [ND3] in [3]) and shown to be $\mathcal{NP}$-complete. In a recent paper [6], Wu et al. propose an efficient approximation algorithm for the all sources problem.

A very general problem of *Optimum Communication Spanning Tree* (*cf.* [ND7] in [3]) has been defined by Hu [4], based on a complete graph G with length $\ell(u,v)$ and requirement $r(u,v)$ for each edge (u,v). In this problem, the cost measure of a spanning tree T of G is defined as the sum of vertex distances in T weighted by the corresponding requirements: $\sum_{u,v \in V} d_T(u,v) r(u,v)$. By restricting the requirements $r(u,v) = 0$ for all vertices u except the sources s_i, and allowing "very large" edge lengths for $(u,v) \notin E$, the problem reduces to k-SPST.

We also consider a multi-source spanning tree problem with the alternative cost measure max-cost. This measure corresponds to the maximum source eccentricity (i.e., maximum distance from a source to any other vertex). Thus we call the problem the *k-Source Maximum-Eccentricity Spanning Tree*, or k-MEST, problem.

k-MEST:

Instance: A graph $G = \langle V, E \rangle$ with length function ℓ, k sources $s_1, \ldots, s_k \in V$, and positive integer K.

Question: Is there a spanning tree T of G whose max-cost does not exceed K?

The two problems above are instances of what could be termed *spanning substructure* problems. Another example would be selection of an optimal k-spanner, being a minimum-cost subset of edges of a graph G such that the distance between any two vertices in the resulting graph is not more than k times the distance in G [2]. These problems can be distinguished from *network location* problems, which attempt to select a subset of vertices in a graph to minimize a given cost function. These problems include p-center problems and various domination problems [5]. We previously considered a location problem inversely related to our present spanning substructure problems: given a tree T, locate k sources to minimize broadcast time in T from those sources [1].

3. The k-SPST Problem

Before presenting an $\mathcal{NP}$-Completeness reduction, we state an observation about the structure of our measure of goodness for a solution of the 2-SPST problem.

Observation 3.1 *The sum-cost of a spanning tree T of a graph G with N vertices, two of which (s_1 and s_2) are sources, is equal to $Nd(p) + 2\sum_{v \in V} d(v,p)$, where p is the unique path between the sources, $d(p)$ its length, and for a vertex v, $d(v,p)$ is the shortest distance of v to a vertex of p.*

Proof. For any vertex v of G, the shortest paths to the two sources include twice the path between v and the nearest vertex w of p and the two subpaths of

$p : (w, s_1)$ and (w, s_2), whose lengths add to $d(p)$. Summing over all vertices of G gives the above expression. $\square$

We will use the 3-SAT decision problem to reduce to the 2-SPST problem.

3-SAT ([LO2] in [3]):

Instance: A set of disjunctive clauses, $C_i,\ 1 \leq i \leq m$ involving literals of boolean variables $x_1, \ldots, x_n$.

Question: Is there a truth value assignment to the variables that satisfies all the clauses?

Theorem 3.1 *2-SPST is $\mathcal{NP}$-complete when all edge lengths are equal.*

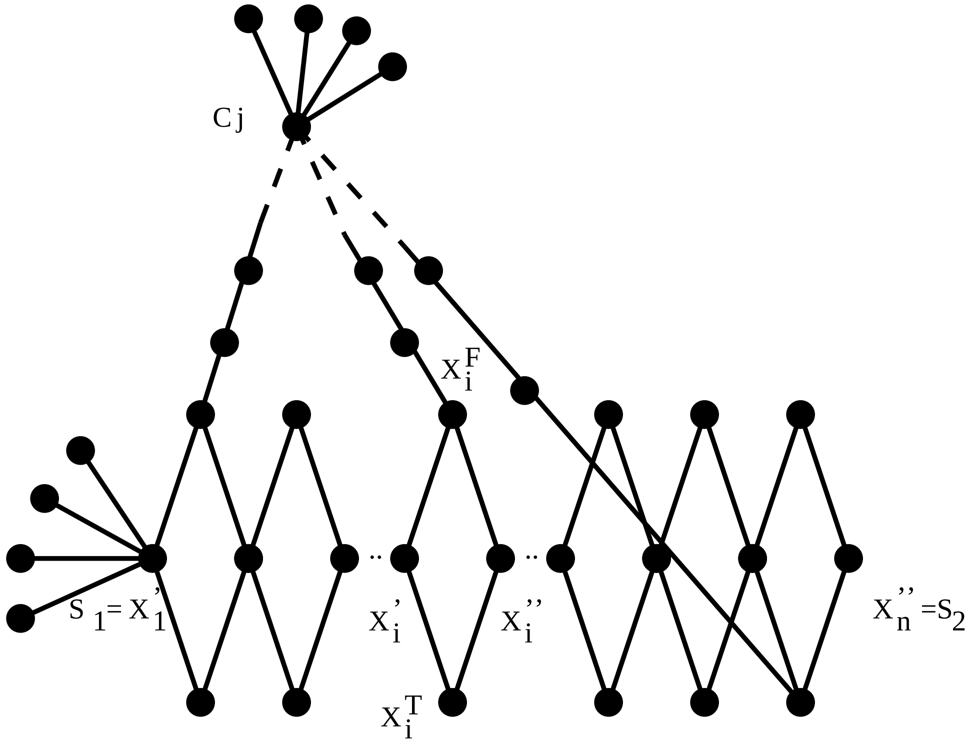

Fig. 1. The construction; a clause $c_j = \bar{x_1} \vee \bar{x_i} \vee x_n$.

Proof. A reduction from 3-SAT problem. Given an instance $C_1, \ldots, C_m$ of 3-SAT, construct an instance of 2-SPST by combining the following "gadgets" (illustrated by the example in Figure 1):

A gadget corresponding to the variable x_i consists of a four-cycle with vertices labeled, in order, $x_i', x_i^T, x_i'', x_i^F$. For each $i,\ 1 \leq i < n$, identify vertices x_i'' and x_{i+1}'.

A gadget corresponding to a clause $C_j = (y_{j_1}, y_{j_2}, y_{j_3})$ consists of a vertex labeled c_j, adjacent to $L = 2nm$ degree-one vertices; this makes it costly to not satisfy the clause (see below). Additionally, the clause vertex c_j is connected to the variable gadget vertices that correspond to the literals y_{j_1}, y_{j_2} and y_{j_3} through paths of $n-1$ vertices each. That is, for each j, $1 \leq j \leq m$, identify the vertex representing literal y_{j_i} of C_j with the F or T vertex of the corresponding variable gadget, depending on whether y_{j_i} is satisfied by assigning `FALSE` or `TRUE` to the variable.

The two source vertices are $s_1 = x'_1$ and $s_2 = x''_n$. This ensures that in any spanning tree, the path between s_1 and s_2 is of length at least $2n$ (and equal to $2n$ only if the path does not pass through a clause vertex). Finally, one of the sources, say s_1, is adjacent to $M = 4m(n+1)(L+1)$ degree-one vertices; this makes it costly to have a long path from s_1 to s_2 in the spanning tree. This graph has $N = M+3n+1+Lm+3nm-2m$ vertices. Completing the construction, the constant K of the instance of 2-SPST is defined to be $2nN+2M+2n+m(n+1)(2L+3n)-4m$.

Let us assume that the instance of 3-SAT is satisfiable. We construct a spanning tree by connecting s_1 and s_2 through the variable gadgets, for each i including the vertex x_i^T or x_i^F (call those "essential variable vertices") according to the satisfying truth assignment. The length of this path p is thus $2n$. We complete the tree by removing edges between a clause vertex and two of its degree-2 neighbors, leaving a connection to an essential variable vertex, for each clause vertex.

There are $3m$ paths in the graph connecting clause vertices to the (s_1, s_2)-path. We designate each one as "short" or "long" according to whether it attaches to an essential or inessential variable vertex. Let P denote the number of long paths. Since each clause has at least one short path, $P \leq 2m$. By Observation 1, the sum-cost of the tree is $2nN + 2\sum_{v \in V} d(v,p)$, where the component $2\sum_v d(v,p)$ equals

0	for vertices v on the s_1, s_2-path
$2M$	for M degree-one vertices v adjacent to s_1
$2n$	for n non-essential variable vertices v
$2nm$	for m clause vertices v
$2Lm(n+1)$	for mL degree-one vertices v adjacent to clause vertices
$P(n(n+1)-2)$	for vertices v in long paths
$(3m-P)(n-1)n$	for vertices v in short paths

This totals $2nN + 2M + 2n + 2Lm(n+1) + 3n^2m - nm + 2P(n-1) \leq K$.

We call a spanning tree T *legal* if it corresponds to a truth assignment (i.e., if no clause vertex lies on the path from s_1 to s_2 in T). If the instance of 3-SAT is not satisfiable, then the sum-cost of a legal tree is higher than K, since the degree-one neighbors of an unsatisfied clause are at distance $n+2$ from the path between s_1 and s_2; they would contribute an extra cost of $2L$, which would make the cost exceed K even if P were 0. P must be at least 3 because of the unsatisfied clause.

If T is not legal, the path from s_1 to s_2 includes a clause vertex and has length at least $2n+2$. The M degree-one neighbors of s_1 contribute an extra $2M$ to the cost of T. The cost of this tree is therefore more than:

$$\begin{aligned} N(2n+2)+2M &> 2Nn+4M \\ &\geq 2Nn+2M+8m(n+1)(L+1) \\ &\geq 2Nn+2M+2Lm(n+1)+6Lm(n+1)+8m(n+1) \\ &\geq 2Nn+2M+2Lm(n+1)+3nm(n+1)+2n. \end{aligned}$$

The cost of this tree exceeds K, implying that only a legal tree corresponding to a satisfying truth assignment for the given 3-SAT problem can be a solution to the derived 2-SPST problem. □

Two corollaries follow regarding the $\mathcal{NP}$-completeness of generalizations of the problem considered above.

Corollary 3.1 *The 2-SPST problem is $\mathcal{NP}$-complete.*

Corollary 3.2 *The k-SPST problem is $\mathcal{NP}$-complete for any $k > 1$.*

In light of the difficulty of solving the 2-SPST problem for general graphs, we will be satisfied with determining efficient solution algorithms for some restricted classes of graphs. We consider first unicyclic graphs consisting of cycle vertices x_i, $0 \leq i < n$ with trees T_i rooted in cycle vertices (*cf.* Figure 2). We use x_n as a synonym for x_0.

We first assume that, in the given input graph G with $N = \sum_{0 \leq i < n} |T_i|$ vertices, the sources are the roots x_0 and x_j of trees T_0 and T_j, $j > 0$ (for $j = 0$, an optimal spanning tree T of G is identical with the single-source shortest-paths spanning tree of G.) As we noted before, the cost of T consists of the length of the shortest path between the sources taken N times plus twice the distances of all vertices to that path. For each tree T_i, we denote by N_i the number of vertices in T_i and by S_i the sum of their distances to the root x_i.

Let us define function $L(x_i, x_k)$ to denote the "clockwise" distance from x_i to x_k; for instance, $L(x_i, x_k) = \sum_{i \leq m < k} l(x_m, x_{m+1})$ for $0 < i \leq k$ and similarly for other cases of i and k. Removing the edge (x_k, x_{k+1}) from the only cycle of G results in a tree, say $T^{(k)}$. The cost of $T^{(k)}$ is equal to, according to Observation 1, N times the sum of the distance between the sources, $L(x_0, x_j)$ when $j \leq k < n$, or $L(x_j, x_0)$ when $0 \leq k < j$, plus twice the distance sums S_i in T_i and twice the following sums:

$$\sum_{j < i \leq k} N_i L(x_j, x_i) + \sum_{k < i < n} N_i L(x_i, x_n) \qquad \text{for } j \leq k < n \tag{3.1}$$

or

$$\sum_{0 < i \leq k} N_i L(x_0, x_i) + \sum_{k < i < j} N_i L(x_i, x_j) \qquad \text{for } 0 \leq k < j. \tag{3.2}$$

Finding a k to minimize the sum-cost of $T^{(k)}$ can be done in linear time. To compute distances from the two sources in an initial tree formed by removing an arbitrary edge of the cycle can be done in linear time. Then, by going around the cycle, removing neighboring edges in turn, we can design an incremental update

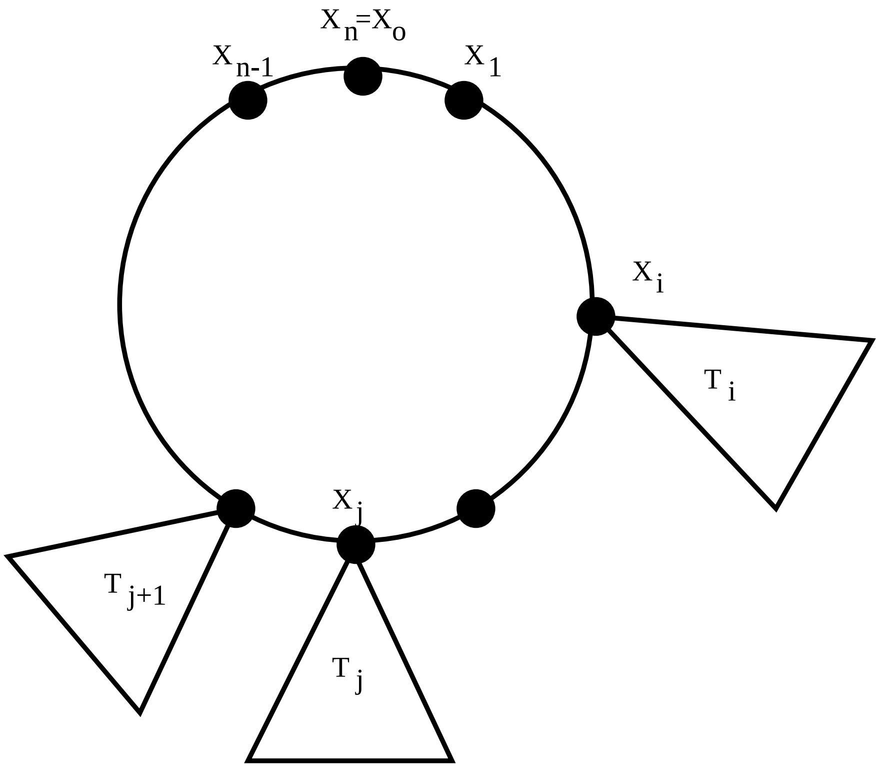

Fig. 2. A unicyclic graph.

algorithm such that the difference between costs of $T^{(k)}$ for two consecutive values of k can be computed in constant time.

If, in the given input graph G with $N = \sum_{0 \leq i < n} |T_i|$ vertices, the two sources are not the roots x_0 and x_j of trees T_0 and T_j, $j > 0$, we merely add their distances from the roots of their respective subtrees to our distance expressions. Then, the same, linear-time algorithm can be applied.

A generalization of unicyclic graphs may involve multiple cycles C_i connected in a tree-like manner through common articulation points. Such graphs are called *cacti*. Let G be a cactus with sources s_1 and s_2. There exists a path of cycles $C_1, \ldots, C_n$ between s_1 (defined to be $x_{1,0}$) and s_2 (defined to be x_{n,j_n}) , such that all (s_1, s_2)-paths pass through articulation vertices $x_{1,j_1} = x_{2,0}, \ldots, x_{n-1,j_{n-1}} = x_{n,0}$ of the consecutive intermediate cycles $C_2, \ldots, C_{n-1}$.

We describe an algorithm to eliminate an edge from each cycle in order to obtain a minimum cost spanning tree T of G. First we observe that for a subgraph separated from the sources by a single vertex $x_{i,j}$ of a cycle C_i, $0 \leq i \leq n$, the single-source shortest-paths spanning tree (with the articulation vertex being the source for the subproblem) provides a subtree of the optimal tree T for the input cactus

G. The number of vertices in such a subtree becomes an attribute $w(x_{i,j})$ of the articulation cycle vertex $x_{i,j}$ (analogous to N_i in the unicyclic case). For each cycle C_i with vertices $x_{i,j}$ weighted by $w(x_{i,j})$, we have a separate optimization problem of choosing an edge $(x_{i,k}x_{i,k+1})$ to remove, according to the formulae (3.1) and (3.2) above. This algorithm computes the optimal spanning tree in time linear as a function of the size of G. This discussion implies the following result.

Theorem 3.2 *The optimal solution to the* 2*-SPST problem for cacti can be obtained in time proportional to the number of vertices in the instance graph.*

We observe further that this algorithm applies to any instance of the problem in which the subgraph induced by all (s_1, s_2)-paths is a cactus. Any biconnected component of the instance graph sharing an articulation vertex s with such a cactus can be optimally spanned by a shortest-paths tree with the single-source s.

Finally, we consider an approximation algorithm for the 2-SPST problem on general graphs that constructs an approximate solution with cost at most twice the optimal. Given a graph G with two sources s_1 and s_2, the algorithm constructs a shortest (s_1, s_2)-path p, contracts its edges collapsing the path's vertices into one "super vertex" s, and then constructs a single-source s shortest-paths tree T' spanning the derived graph G'. The spanning tree T of G consisting of the path p and the edges of G corresponding to edges of T' has cost at most twice that of an optimal spanning tree T^* of G.

Theorem 3.3 *The optimal solution to the* 2*-SPST problem can be efficiently approximated within a factor of 2.*

Proof. Assume that the (s_1, s_2)-path p^* in T^* has length M. Any vertex v of G is at most $M/2$ further away from p than from p^* (this because a vertex of p^*, namely a source, can be reached from v in T within this extra distance, in addition to the distance from v to p in T). Thus, in T we over-estimate the cost of T^* by at most $2n(M/2) = Mn$, which is a lower bound on the cost of T^*. □

4. The k-MEST Problem

Now, we consider the alternative cost measure for an optimal spanning tree, namely max-cost (or the maximum eccentricity of a source). We will show an efficient solution algorithm for unicyclic graphs, leading to a pseudo-polynomial algorithm (of complexity polynomial as a function of graph size and edge weights) for cacti. Subsequently, we show that the general problem is solvable in pseudo-polynomial time.

For a cycle vertex x_i, $0 \le i < n$, of the unicyclic graph G, let $d(x_i)$ be the depth of the tree T_i. Let the function $L(x_i, x_k)$ be defined as before to denote the clockwise distance around the cycle. The maximum eccentricity of a source is minimized with the following expressions (when choosing the removed edge (x_k, x_{k+1})):

$$L(x_j, x_n) + \max\{\max_{0 \le i \le k}(L(x_0, x_i) + d(x_i)),\ \max_{k < i \le j}(L(x_i, x_j) + d(x_i))\} \quad \text{for } 0 \le k < j \quad (4.3)$$

$$L(x_0, x_j) + \max\{\max_{k<i<n}(L(x_i, x_n) + d(x_i)), \max_{j \le i \le k}(L(x_j, x_i) + d(x_i))\} \text{ for } j \le k < n \quad (4.4)$$

The expressions (4.3) and (4.4) can be evaluated in time proportional to n by separately computing values of L for x_0 and x_j (as one of the arguments) and then incrementally updating them with $d(x_i)$, to obtain the relevant maximum values. A final scan of those values will yield the minimum maximum source eccentricity. If the sources are located not on the cycle but in the trees, T_0 and T_j, then the distances of the sources to the cycle, $D_1 = d(s_1, x_0)$ and $D_2 = d(s_2, x_j)$, respectively, would modify the formulae (4.3) and (4.4) in the obvious manner.

Theorem 4.4 *The 2-MEST problem can be solved in linear time on unicyclic graphs.*

This theorem suggests a pseudo-polynomial time solution algorithm for the 2-MEST problem on cacti having uniform edge lengths. Namely, for each cycle C_i, as above, solve the problem for the *unicyclic* case for *all* possible values of D_1 and D_2 and then use dynamic programming to find an optimal spanning tree T.

Theorem 4.5 *The 2-MEST problem on cacti graphs can be solved in polynomial time when edge lengths are all equal or have magnitude polynomially bounded by the size of the graph.*

There is a pseudo-polynomial time algorithm for the k-MEST problem on general graphs having uniform edge lengths. A solution can be obtained by a low-order polynomial time algorithm based on computing single-source shortest-paths trees. Unfortunately, in the general case the shortest-paths algorithm has to be executed a number of times that is proportional not only to the number of edges but also to their lengths. The latter may not be polynomial in the size of the problem instance.

We will show that for an instance graph G with sources $S = \{s_1, \ldots, s_k\}$ and uniform edge lengths, there exists either a vertex x, or an edge (y, z), such that the shortest-paths tree (from either x or the "midpoint" of (y, z), respectively) in G minimizes the maximum source eccentricity.

We first formulate some facts relating source eccentricities in an optimal tree T^* of G to those in a shortest-paths tree T_x, for a certain vertex x. Let q be the maximum intra-source distance in T^* and let M^* be the maximum source eccentricity in T^*. Let P be a path of length q between a pair of sources, s_1 and s_2, in T^*. Let x be the midpoint (equidistant from s_1 and s_2) on P. If q is odd, x will be an auxiliary vertex subdividing the middle edge (y, z) of P; we define the lengths of (x, y) and (x, z) to be $\frac{1}{2}$. We observe that this modification does not change the solution to the original problem. The following facts are implied by our definitions.

Fact 4.1 *In T^*, no source $s \in S$ is more than $q/2$ farther away from x, i.e., $d_{T^*}(x, s) \le q/2$.*

Fact 4.2 *For any vertex v in G, the distance in T^* to x, $d_{T^*}(x, v) \le M^* - q/2$.*

Now, let T_x be a shortest-paths tree of G (possibly modified with the auxiliary vertex x, as per above) with the single source x. The construction implies the following fact.

Fact 4.3 *For any vertex v in G, the distance from v to x in T^* is not smaller than the distance from v to x in T_x.*

From these facts, it follows that the maximum source eccentricity in T_x does not exceed M^*.

Theorem 4.6 *For G, $S = \{s_1, \dots, s_k\}$, M^*, q and x, as above,*

$$\max_{v \in V, s \in S} d_{T_x}(v, s) = M^*.$$

Proof. By the definition of M^*, no spanning tree of G has a smaller max-cost, i.e., $\max_{v \in V, s \in S} d_{T_x}(v, s) \geq M^*$. The distance $d_{T_x}(v, s)$ is bounded by the sum of distances from x to v and s which is bounded by M^*, as per the above Facts:

$$d_{T_x}(v, x) + d_{T_x}(x, s) \leq d_{T^*}(v, x) + d_{T^*}(x, s) \leq (M^* - q/2) + q/2 = M^*.$$

□

The above theorem implies a brute-force algorithm that requires only polynomial time as a function of the instance size: compute a single-source shortest-paths tree for each vertex x and for the midpoint of each edge (y, z) in G. A tree that minimizes source eccentricity is the solution to the given instance of the k-MEST problem with uniform edge length.

A reduction from a general instance of k-MEST (with integral, non-uniform edge lengths) involves the construction of a derived graph, G', with edges of uniform length. G' is obtained from G by replacing each edge (y, z) with a path of $\ell(y, z)$ edges (with all new internal vertices of the path). It is easy to see that the minimum source eccentricity in the optimal spanning tree T' of G' is the minimum source eccentricity of the spanning tree of G uniquely determined by T'. This implies the following result.

Theorem 4.7 *The k-MEST problem can be solved exactly by a pseudopolynomial time algorithm.*

The reduction is polynomial-time only when edge lengths are bounded by a polynomial in the size of G. This remark limits the usefulness of our algorithm in the general case of k-MEST. In the general case, we can again specify an approximation algorithm. The algorithm that produces an approximation to the 2-SPST problem also gives an approximate solution to the 2-MEST problem.

Theorem 4.8 *The optimal solution to the 2-MEST problem can be efficiently approximated within a factor of 2.*

Proof. Let the graph G' be obtained from the instance graph G by collapsing vertices of a shortest path between the sources s_1 and s_2 into vertex s. Let T' be a single-source shortest-paths tree from s in G'. We obtain a spanning tree T of G by adding to the original edges of T' the contracted edges of the (s_1, s_2)-path. This can obviously be done in the time polynomial as a function of the graph size. The maximum source eccentricity, M, of a vertex in T is not larger than the distance between the sources, m, plus the maximum distance d from any vertex to the single source s of the derived graph G'. Being shortest path distances in G, both d and m are lower bounds on the maximum source eccentricity M^* in the optimal spanning tree T^* of G. Thus we have $M \leq 2M^*$. $\square$

5. Conclusion

In this paper we consider two multi-source spanning tree problems based on different measures of fitness. One evaluates a spanning tree in terms of the sum of lengths of paths from the sources to all other vertices. The other evaluates the tree in terms of the maximum eccentricity of any source. We show the first problem to be $\mathcal{NP}$-hard in the strong sense. We provide exact algorithms on a restricted class of graphs and an approximation algorithm for the case where there are two senders. Design of efficient algorithms for other restricted graph classes and giving an approximation ratio less than 2 remain open. The second problem is somewhat easier; we are able to provide an efficient exact solution for all cases where there is an given upper-bound on edge lengths. We also provide a general approximation algorithm for the case of two senders.

Acknowledgment

We thank Mikael Goldman for his modifications to our original $\mathcal{NP}$-completeness reduction, yielding a proof for the uniform edge lengths case of 2-SPST.

AMF and AP were supported in part by the grant NSF-ANI-9977524. PF was supported in part by the Marie-Curie postdoctoral fellowship.

References

1. A.M.Farley and A. Proskurowski, “Broadcasting in trees with multiple originators,” *SIAM Journal of Algebraic and Discrete Methods* **2**(4)(1981), 381–386;.
2. A.M. Farley, A. Proskurowski and K. Windish, “Spanners and Message Distribution in Networks,” UO-TR-99-03, University of Oregon (1999).
3. M.R. Garey and D.S. Johnson, *Computers and Intractability*, (Freeman 1971).
4. T.C. Hu, “Optimum communication spanning trees,” *SIAM J. Computing* **3** (1974), 188–195.
5. O. Kariv and S.L. Hakimi, “An algorithmic approach to network location problems, I: the p-centers,” *SIAM J. Appl. Math.* **37** (1979), 513.
6. 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,” in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, (1998), 21–32.