Multi-source Trees: Algorithms for Minimizing Eccentricity Cost Metrics

Paraskevi Fragopoulou[1], Stavros D. Nikolopoulos[2], and Leonidas Palios[2]

[1] Department of Applied Informatics and Multimedia,
Technological Educational Institute of Crete,
P.O.Box 1939, GR-71004 Heraklion-Crete, Greece
fragopou@epp.teiher.gr

[2] Department of Computer Science, University of Ioannina,
GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr

Abstract. We consider generalizations of the k-source sum of vertex eccentricity problem (k-SVET) and the k-source sum of source eccentricity problem (k-SSET) [1], which we call SDET and SSET, respectively, and provide efficient algorithms for their solution. The SDET (SSET, resp.) problem is defined as follows: given a weighted graph G and sets S of source nodes and D of destination nodes, which are subsets of the vertex set of G, construct a tree-subgraph T of G which connects all sources and destinations and minimizes the SDET cost function $\sum_{d \in D} max_{s \in S}\, d_T(s,d)$ (the SSET cost function $\sum_{s \in S} \max_{d \in D}\, d_T(s,d)$, respectively). We describe an $O(nm \log n)$-time algorithm for the SDET problem and thus, by symmetry, to the SSET problem, where n and m are the numbers of vertices and edges in G. The algorithm introduces efficient ways to identify candidates for the sought tree and to narrow down their number to $O(m)$. Our algorithm readily implies $O(nm \log n)$-time algorithms for the k-SVET and k-SSET problems as well.

Keywords: Multi-source trees, eccentricity, weighted graphs, networks, communication, algorithms, complexity.

1 Introduction

The work in this paper is motivated by problems in collective communication on networks modeled by graphs. In the general case, a group of network nodes, defined as the sources, wish to consistently transmit information to another group of network nodes, the destinations. This type of collective communication is served by the establishment of a tree T connecting the sources to the destinations which minimizes certain criteria in order to guarantee efficient communication. The criteria to optimize are diverse and various cases have been considered in the literature [1, 3, 6, 7, 9]. The main reason is that different applications pose different cost requirements; note also that some of these cost requirements lead to intractable problems in general graphs. The problem of collective communication from a single source node is a well studied problem [4, 5, 8, 10]. Multiple sources have also been considered [3, 6], although to a lesser degree.

In its general form, the construction of optimum communication spanning trees was initiated in [4] where the problem was defined on the complete graph with a length and a requirement on its edges; the cost measure that had to be minimized was the sum of vertex distances weighted by the requirements between all vertices of the graph (network). By setting the requirement equal to 0 and parameterizing the number of source nodes, we obtain the k-SPST problem or *sum of distances from every source to every destination*, which was studied in [3] and was shown to be NP-complete. Some exact solutions were provided for the 2-source SPST problem on restricted classes of graphs, such as unicycles and cactuses [3]. Furthermore, approximation algorithms for both the 2-source SPST problem on general graphs [3] and the all source SPST problem [10] are available. In [9], heuristic algorithms were given for the minimum partial spanning trees taking into consideration the delay between a single source node and a group of destination nodes, while trying to bound the maximum difference in these delays. Another problem, the k-MEST problem, is defined in terms of the *maximum distance from a source to a destination*, known as the *maximum eccentricity.* This cost function was studied in [3] for some special types of graphs; in [6], the problem in its general form (arbitrary sets of source and destination nodes) was shown to be tractable and an efficient polynomial algorithm which for a graph on n vertices runs in $O(n^3)$ time was given. The same was independently established in [7] considering all graph vertices as destinations via an $O(n^3 + nm \log n)$-time algorithm, where m is the number of edges of the graph.

The k-SPST problem takes into consideration all source-destination distances but defines an intractable problem, whereas the k-MEST problem takes into consideration only the maximum source-destination distance and thus does not give any indication for the distances for the rest of the source-destination pairs. Two cost functions that fill the gap between these two extreme cases were introduced in [1]: the k-SVET cost function or *sum of vertex eccentricities spanning tree* is defined as the sum of distances from each destination to its most distant graph vertex in the constructed spanning tree, and the k-SSET cost function or *sum of source eccentricities spanning tree* is defined as the sum of distances from each source to its most distant vertex in the constructed tree.

In this paper, we consider the following generalizations of the k-SVET and the k-SSET problems: for a set of source nodes and a set of destination nodes, which are arbitrary subsets of the vertex set of a graph, we are interested in minimizing the sum of distances from each destination (source, respectively) to its most distant source (destination, respectively) node in the constructed tree. Under this generalization, the two problems, the generalized k-SVET and k-SSET, become symmetric (simply exchange S and D) and thus the results derived for one of them apply to the other in a straightforward manner. We call the generalized k-SVET problem *SDET* and the generalized k-SSET problem *SSET*. We derive an $O(nm \log n)$-time algorithm for the SDET problem and thus, by symmetry, to the SSET problem, where n and m are the numbers of vertices and edges of the given graph (network). The algorithm introduces efficient ways to identify candidates for the sought tree and to narrow down their

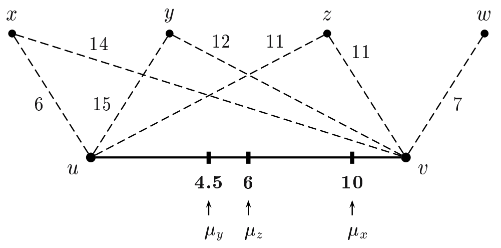

Fig. 1. The midpoints of the vertices x, y, z on an edge uv of length 12

number to $O(m)$. Our algorithm readily implies $O(nm \log n)$-time algorithms for the k-SVET and k-SSET problems as well.

2 Theoretical Framework

Let G be a simple weighted graph which models a network and has vertex set $V(G)$, edge set $E(G)$, and non-negative symmetric weights on the edges, and let $S, D \subseteq V(G)$ be the set of source nodes and the set of destination nodes, respectively; the sets S and D do not need to be disjoint. By $d(u, v)$ we denote the length (weight) of the shortest path from vertex u to vertex v in the graph G, and by $d_T(u, v)$ the length (weight) of the path from node u to node v in a spanning tree T of G.

Let uv be an edge of G. A *point* on uv is either a vertex-endpoint or a location on uv. For any two points α, β on uv, we denote by $\ell(\alpha, \beta)$ the length from α to β along uv. Thus, the length (weight) of the edge uv is denoted by $\ell(u, v)$. (Note that $d(u, v) \leq \ell(u, v)$, although $\ell(u, v), d(u, v)$ are not necessarily equal.) With respect to the edge uv, we partition the vertex set $V(G)$ as follows:

$$\begin{aligned} V_u &= \{u\} \cup \{\, w \mid \text{the shortest paths from } v \text{ to } w \text{ all go along the edge } vu \,\} \\ V_v &= \{v\} \cup \{\, w \mid \text{the shortest paths from } u \text{ to } w \text{ all go along the edge } uv \,\} \\ V_{uv} &= V(G) - (V_u \cup V_v). \end{aligned}$$

(For example, in Figure 1, $V_u = \{u\}$, $V_v = \{v, w\}$, and $V_{uv} = \{x, y, z\}$.) For each vertex $x \in V_{uv}$, we define the midpoint μ_x of x with respect to uv as the point on uv such that $\ell(v, \mu_x) = 1/2 \cdot \big(d(u, x) - d(v, x) + \ell(u, v)\big)$; note that for each such vertex x, any shortest path from μ_x to x through u and any shortest path from μ_x to x through v are of equal length (see Figure 1).

Let T be a spanning tree of the graph G and let u, v be any two distinct vertices of G. The removal of the path ρ connecting u, v in T produces two subtrees of T, containing u and v, respectively. Then, we say that a vertex x that does not belong to the path ρ is *connected to* u (v, resp.) in T if x and u (v, resp.) belong to the same subtree. Then, the definitions of V_u, V_v, V_{uv}, and of the midpoint of a vertex imply the following observation:

Observation 1. *Let p be a point on an edge uv and T_p a shortest paths tree rooted at p. Then, for any vertex $x \in V(G)$, we have:*

(i) If $x \in V_u$ then vertex x is connected to u in T_p; if $x \in V_v$ then vertex x is connected to v in T_p.
(ii) If $x \in V_{uv}$ then: if the midpoint μ_x of x with respect to uv is located in the interval $[u, p)$ then vertex x is connected to v in T_p; if μ_x is located in the interval $(p, v]$ then x is connected to u in T_p; if μ_x coincides with p, then x may be connected either to u or to v in T_p.

Observation 1 implies the following corollary.

Corollary 1. *If a vertex x is connected to u in a shortest paths tree T_p rooted at a point p of an edge uv, then x is connected to u in any shortest paths tree rooted at any point in the interval $[u, p]$; if x is connected to v in T_p, then x is connected to v in any shortest paths tree rooted at any point in the interval $[p, v]$.*

Given a spanning tree T of the graph G and a vertex v, a *critical source* for vertex v in T is an element of the set S of source nodes at maximum distance from v in T. For the tree T, two sources at maximum intrasource distance in T (i.e., their distance in T is no less than the distance of any other pair of sources) form a *pair of critical sources.* It is important to note that a tree T may have more than one pair of critical sources; if this is the case, then we can show the following:

Lemma 1. *Let T be a spanning tree of a graph, a set S of source nodes, and suppose that T has more than one pair of critical sources. Then:*

(i) The midpoints of all the paths in T connecting pairs of critical sources coincide.
(ii) Let a, b and c, d be two pairs of critical sources. Then, a, c and b, d or a, d and b, c are also pairs of critical sources.

The following lemma, established in [1], gives properties of trees which are important both for the k-SVET and the SDET problems.

Lemma 2. [1] *Let s_1 and s_2 be two sources with maximum intrasource distance in a tree T. For any vertex $d \in V(T)$ and any source $s_i \in S - \{s_1, s_2\}$, either $d_T(d, s_i) \leq d_T(d, s_1)$ or $d_T(d, s_i) \leq d_T(d, s_2)$.*

In other words, for each vertex v in a tree T with a pair s_1, s_2 of critical sources, s_1 or s_2 is a critical source for v in T. Based on this and other results, Connamacher and Proskurowski [1] showed the following theorem (the same technique had been used to show that the maximum eccentricity problem is polynomial [6]):

Theorem 1. [1] *Given a weighted graph G, there exists a point χ such that any shortest paths tree rooted at χ is an optimal tree for the k-SVET problem.*

Theorem 1 shows that the tree minimizing the k-SVET cost function is a shortest paths tree rooted at a vertex or at a point on an edge of G. In a similar fashion, we can show the following:

Theorem 2. *Given a weighted graph G, a set S of source nodes, and a set D of destination nodes, there exists an optimal tree for the SDET problem which is a shortest paths tree rooted at a point on an edge of the graph G.*

3 The Algorithm

Our algorithm for the SDET problem relies on Theorem 2. It receives as input a weighted undirected graph G with non-negative symmetric weights and outputs a shortest paths tree that minimizes the SDET cost function. In high level, it works as follows:

Algorithm SDET

```
1. for each vertex $v$ of the graph $G$ do
   1.1 compute the distances $d(v, x)$ in $G$ from $v$ to every other vertex $x \in V(G)$;
   1.2 associate each pair $v, x$ with either a neighbor $u$ of $v$ if all the shortest
       paths from $v$ to $x$ go along the edge $uv$, or with $v$ otherwise;
2. $mincost \leftarrow +\infty$;
   for each edge $uv$ of $G$ do
   2.1 determine the description of a shortest paths tree $T$ of $G$ rooted at a
       point on the edge $uv$ which minimizes the SDET cost function over all
       shortest paths trees rooted at points of $uv$ and let cost($T$) be the value
       of the SDET cost function for $T$;
   2.2 if cost($T$) < $mincost$
       then save the description of $T$ as it gives the currently optimal tree $T_{opt}$;
            $mincost \leftarrow$ cost($T$);
3. Construct the tree $T_{opt}$ from its description and clip it by repeatedly remo-
   ving leaves that do not belong to $S \cup D$.
```

Clearly, the correctness of Step 2.1 implies the correctness of the entire algorithm. For the execution of Step 2.1 on an edge uv, we first compute a list L_{uv} of shortest paths trees rooted at points on uv, which includes a tree exhibiting the minimum of the SDET cost function over all shortest paths trees rooted on uv, and then, among the trees in L_{uv}, we select one with the minimum value of the SDET cost; these are discussed in the following subsections.

3.1 Finding Candidates (for the Optimal Tree) Rooted on an Edge

Let us consider an edge uv of the input graph G. In order to guarantee the correctness of our algorithm, we should consider all structurally different shortest paths trees of G rooted at points on uv. Observation 1 narrows down the possibilities and determines which vertices are connected to u and which to v in a tree: it implies that if we walk along the edge uv from u to v and compute the shortest paths tree rooted at the current point of uv, the tree is unique and remains the same for as long as we do not cross any midpoint; when we cross the midpoint μ_x of a vertex x, then x, which has been connected to u in the shortest paths trees considered so far, gets now connected to v. Since the vertices in V_{uv} are those contributing midpoints on the edge uv, we consider the partition of the set S of source nodes into the following three sets:

$$S_u = S \cap V_u \qquad S_v = S \cap V_v \qquad S_{uv} = S \cap V_{uv}$$

(for example, if $S = \{x, y, z, w\}$ in Figure 1, then $S_u = \emptyset$, $S_v = \{w\}$, and $S_{uv} = \{x, y, z\}$).

The following lemma, which we establish, helps us avoid additional unnecessary work as well.

Lemma 3. *Let G be a graph, and $S, D \subseteq V(G)$ be sets of source and destination nodes, respectively. For the computation of an optimal solution for the SDET problem for G, S, D, it suffices that we consider only shortest paths trees of G rooted at points r satisfying both following conditions:*

(i) the root r of the tree lies on the path connecting a pair of critical sources in the tree;
(ii) the root r and the (common) midpoint of all the paths connecting pairs of critical sources in the tree are located on (the closure of) an edge of G.

Lemma 3 has the following very important implications:

Corollary 2. *Let G, S, D be as described in Lemma 3. For the computation of an optimal solution for the SDET problem for G, S, D, it suffices that we consider only shortest paths trees T of G such that if σ_u (σ_v, resp.) is a source connected to u (v, resp.) in T at maximum distance from u (v, resp.), the midpoint of the path connecting σ_u and σ_v in T belongs to the edge uv. Then, in any such tree:*

(i) the sources σ_u, σ_v form a pair of critical sources in T;
(ii) the source σ_u (σ_v, resp.) is critical for every vertex x connected to v (u, resp.) in T.

Finally, if the midpoints of k source nodes with respect to the edge uv coincide at a point p, then there are 2^k structurally different shortest paths trees rooted at p (and in fact, even more if midpoints of other vertices also coincide with p). Yet, in such a case, we can show the following:

Lemma 4. *Let G be a graph, $S, D \subseteq V(G)$ be sets of source and destination nodes, respectively, and $A \subseteq S$ be a set of source nodes whose midpoints all fall at a point r on an edge uv of G. Then, for the computation of an optimal solution for the SDET problem for G, S, D, among all the shortest paths trees of G rooted at r, it suffices that we consider only those in which the sources in A are either all connected to u or all connected to v.*

The details of the processing of an edge uv of the input graph G are given in the Algorithm TREES_ROOTED_ON_EDGE presented below. For an edge uv of G, the algorithm produces a list L_{uv} of shortest paths trees rooted at points on uv, which is guaranteed to include a tree minimizing the SDET cost function over all such trees with their roots on uv; the trees are listed in L_{uv} in the order their roots are met along uv from u to v, and each such tree T is represented by its root r, the distance $\delta_u(r)$ of u to the critical sources in T connected to u, the distance $\delta_v(r)$ of v to the critical sources in T connected to v, and a number $k_S(r)$ such that the sources stored in a subarray $\Sigma[1..k_S(r)]$ of an array Σ of size $|S|$

are those connected to v in T whereas the remaining ones are those connected to u. We also note that in the case that the midpoints of several sources coincide, Algorithm TREES_ROOTED_ON_EDGE considers more possibilities than the two specified in Lemma 4; nevertheless, it certainly considers those two.

Algorithm TREES_ROOTED_ON_EDGE

```
1. Compute the sets $S_u$, $S_v$, and $S_{uv}$, and the midpoints of the source nodes in
   $S_{uv}$ as well as their distances from vertex $u$ on the edge $uv$;
2. Construct a sorted list $(s_1, s_2, \ldots, s_t)$ of the source nodes in $S_{uv}$ in order of
   non-decreasing distance of their midpoints (on the edge $uv$) from $u$;
   Construct a sorted array $\Sigma$ of the source nodes which stores the elements
   of $S_v$, followed by the sources $s_1, s_2, \ldots, s_t$ in that order, followed by the
   elements in $S_u$;
   $s_0 \leftarrow$ source node in $S_v$ (if any) at maximum distance from $v$;
   $s_{t+1} \leftarrow$ source node in $S_u$ (if any) at maximum distance from $u$;
3. Construct two arrays: $A_u = [d(u, s_1), d(u, s_2), \ldots, d(u, s_t), d(u, s_{t+1})]$ and
   $A_v = [d(v, s_0), d(v, s_1), d(v, s_2), \ldots, d(v, s_t)]$;
4. Compute the suffix-maxima $[a_1, a_2, \ldots, a_{t+1}]$ on the array $A_u$ and the prefix-
   maxima $[b_0, b_1, \ldots, b_t]$ on the array $A_v$, i.e., $a_i = \max\{d(u, s_i), d(u, s_{i+1}), \ldots,$
   $d(u, s_{t+1})\}$ and $b_i = \max\{d(v, s_0), d(v, s_1), \ldots, d(v, s_i)\}$;
5. $L_{uv} \leftarrow$ an empty list;
   for each $i = 0, 1, 2, \ldots, t$ do
       {consider the shortest paths tree rooted at $u$ if $i = 0$ or at $\mu_{s_i}$ otherwise,
        in which the sources in $S_v \cup \{s_1, s_2, \ldots, s_i\}$ are connected to $v$ and
        the sources in $\{s_{i+1}, s_{i+2}, \ldots, s_{t+1}\} \cup S_u$ are connected to $u$}
       if ($i = 0$ and $S_v = \emptyset$) or ($i = t$ and $S_u = \emptyset$)
       then do nothing;        {$b_0$ or $a_{t+1}$ are not well defined}
       else if $| a_{i+1} - b_i | \leq \ell(u, v)$
             then   {the midpoint of the paths connecting pairs of critical
                       sources belongs to uv}
                   if $i = 0$ then $r \leftarrow u$;
                              else $r \leftarrow$ midpoint $\mu_{s_i}$ of $s_i$ on the edge $uv$;
                   $\delta_u(r) \leftarrow a_{i+1}$;
                   $\delta_v(r) \leftarrow b_i$;
                   $k_S(r) \leftarrow |S_v| + i$;
                   insert at the end of $L_{uv}$ a record for a shortest paths tree
                   represented by its root $r$, $\delta_u(r)$, $\delta_v(r)$, and $k_S(r)$;
6. Return the list $L_{uv}$ of shortest paths trees and the array $\Sigma$;
```

The correctness of the algorithm follows from Observation 1, Theorem 2, Corollary 2, and Lemma 4. It is not difficult to see that the algorithm runs in $O(|S| \log |S|)$ time. Thus, we have:

Lemma 5. *Given a weighted graph G, a set of source nodes $S \subseteq V(G)$, and an edge uv of G, Algorithm* TREES_ROOTED_ON_EDGE *computes in $O(|S| \log |S|)$ time a collection of shortest paths trees rooted at points on uv among which there is one that minimizes the SDET cost function over all shortest paths trees rooted on uv.*

3.2 Selecting a Shortest Paths Tree of Minimum Cost Among Those Rooted at Points of an Edge

Let uv be an edge of the given graph G. In light of Lemma 5, finding a shortest paths tree of minimum SDET cost among those rooted at points on uv reduces to finding a tree of minimum SDET cost among those computed by Algorithm Trees_Rooted_on_Edge (see Section 3.1). We process these trees in the order their roots are met on the edge uv from u to v; similarly, we process the destination nodes in the order their midpoints are met on the edge uv from u to v. If we consider the partition of the set D of destination nodes into

$$D_u = D \cap V_u, \qquad D_v = D \cap V_v, \qquad D_{uv} = D \cap V_{uv},$$

then, for any shortest paths tree T rooted at a point r of uv and any destination node d, Observation 1 specifies whether d is connected to u or to v in T. Additionally, the critical sources connected to u (v, resp.) are critical for each destination node connected to v (u, resp.); see Corollary 2. Finally, Lemma 6 (similar to Lemma 4) addresses the case of destination nodes whose midpoints coincide with the root of a shortest paths tree.

Lemma 6. *Let G be a graph, $D \subseteq V(G)$ the set of destination nodes, $B \subseteq D$ be a set of destination nodes whose midpoints all fall at a point r on an edge uv of G, and let σ_u, σ_v be sources connected to u and v, repsectively, forming a critical pair in a shortest paths tree rooted at r. Then, for the computation of an optimal solution for the SDET problem for G, S, D, among all the shortest paths trees of G rooted at r, it suffices that we consider only the one in which the destinations in B are either all connected to v if $d(u,\sigma_u) + \ell(u,r) < \ell(r,v) + d(v,\sigma_v)$, or all connected to u otherwise.*

The details of the computation are given below:

Algorithm Edge_Min_Cost

```
1. Compute the sets $D_u$, $D_v$, and $D_{uv}$, and the midpoints of the destination
   nodes in $D_{uv}$ as well as their distances from vertex $u$ on the edge $uv$;
2. Construct a sorted array $\Delta$ of the destination nodes which stores the elements
   of $D_v$, followed by the elements of $D_{uv}$ in order of non-decreasing distance
   of their midpoints (on the edge $uv$) from $u$, followed by the elements in $D_u$;
3. Execute Algorithm Trees_Rooted_on_Edge on the edge $uv$: in addition to
   an ordered array $\Sigma$ of the sources, the algorithm returns a list $L_{uv}$ of shortest
   paths trees rooted at points on $uv$ in the order their roots $r_1, r_2, \ldots, r_{t'}$
   appear along $uv$ from $u$ to $v$; the tree rooted at $r_i$ is also associated with the
   distances $\delta_u(r_i)$ and $\delta_v(r_i)$, and the number $k_S(r_i)$ (see Section 3.1);
4. $c_u \leftarrow \sum_{x \in D - D_v} d(u,x)$    and    $c_v \leftarrow \sum_{x \in D_v} d(v,x)$;
   $j \leftarrow |D_v| + 1$    and    $mincost \leftarrow +\infty$;
   for each candidate root location $r_i \in L_{uv}$, $1 \le i \le t'$, do
   4.1 while $j \le |D| - |D_u|$ and $\mu_{\Delta[j]}$ is to the left of $r_i$ on the edge $uv$ do
           subtract the value $d(u, \Delta[j])$ from $c_u$;
           add the value $d(v, \Delta[j])$ to $c_v$;
           $j \leftarrow j + 1$;
```

```
    4.2 if $\delta_u(r_i) + \ell(u, r_i) < \ell(r_i, v) + \delta_v(r_i)$        {*apply Lemma 6*}
        then while $j \leq |D| - |D_u|$ and $\mu_{\Delta[j]}$ coincides with $r_i$ do
                  subtract the value $d(u, \Delta[j])$ from $c_u$;
                  add the value $d(v, \Delta[j])$ to $c_v$;
                  $j \leftarrow j + 1$;
        $k_D(r_i) \leftarrow j - 1$;       {*number of destination nodes connected to v*}
    4.3 $cost \leftarrow c_v + k_D(r_i) \cdot \big(\ell(u,v) + \delta_u(r_i)\big) +$
                 $c_u + \big(|D| - k_D(r_i)\big) \cdot \big(\ell(u,v) + \delta_v(r_i)\big)$;
        if $cost < mincost$
        then $\widehat{r}_{uv} \leftarrow r_i$;
             $mincost \leftarrow cost$;
 5. return a description of the computed shortest paths tree consisting of its
    root $\widehat{r}_{uv}$, its cost $mincost$, the ordered pair $(u, v)$, and the source nodes stored
    in $\Sigma[1..k_S(\widehat{r}_{uv})]$ and the destination nodes stored in $\Delta[1..k_D(\widehat{r}_{uv})]$ which are
    to be connected to $v$ whereas the remaining source and destination nodes
    are to be connected to $u$;
```

The correctness of the algorithm follows from the discussion preceding the description of the algorithm. Regarding the complexity of the algorithm, we have: Step 1 requires $O(|D|)$ time, Step 2 $O(|D| \log |D|)$ time, Step 3 $O(|S| \log |S|)$ time (Lemma 5), Step 4 $O(|S| + |D|)$ time (for each $d_i \in D$, the distances $d(u, d_i)$ and $d(v, d_i)$ are available in constant time thanks to Step 1 of Algorithm SDET and we spend $O(1)$ time for each candidate root r_i and each destination node), and Step 5 $O(|S|+|D|)$ time. In total, the algorithm runs in $O(|S| \log |S|+|D| \log |D|)$ time. Thus, we have the following result.

Lemma 7. *Given a weighted graph G, a set of source nodes $S \subseteq V(G)$, a set of destination nodes $D \subseteq V(G)$, and an edge uv of G, then Algorithm* EDGE_MIN_COST *runs in $O(|S| \log |S| + |D| \log |D|)$ time and produces the description of a shortest paths tree of G rooted at a point of uv which minimizes the SDET cost function over all shortest paths trees rooted at points on uv.*

3.3 Time Complexity of Algorithm SDET

We assume that the input graph G has n vertices and m edges and is given in adjacency list representation.

Step 1: The distances $d(v, x)$ from v to all other vertices x of the input graph G can be computed using the well known Dijkstra's algorithm in $O((n + m) \log n)$ time [2]. Since $|S| + |D| = O(n)$, we have that Step 1.1 is executed in $O(n(n + m) \log n)$ time. Finding whether the shortest paths from vertex v to any source or destination node x all go along the same edge incident on v or not can be easily carried out by executing a slightly modified version of Dijkstra's algorithm for v which maintains this information; the modified version runs in $O(n(n + m) \log n)$ time for all pairs of vertices of G as well. In total, Step 1 requires $O(n(n + m) \log n)$ time.

Step 2: Since Algorithm EDGE_MIN_COST runs in $O(|S| \log |S| + |D| \log |D|)$ time for each edge of G (Lemma 7), the step is executed in $O(nm \log n)$ time.

Step 3: The shortest paths tree T_{opt} can be constructed from its description in $O((n+m)\log n)$ time by using Dijkstra's algorithm to determine the desired shortest paths, while the clipping of unnecessary leaves takes $O(n)$ time.

Therefore, we obtain the following result.

Theorem 3. *The SDET problem on a weighted graph G on n vertices and m edges is solved in $O(nm\log n)$ time.*

4 Concluding Remarks

We described an algorithm for the SDET problem, which runs in $O(nm\log n)$ time and, to the best of our knowledge, is the first one for the problem in question. The algorithm also provides $O(nm\log n)$-time algorithms for the k-SVET and k-SSET problems for which it has only been proven that they are polynomial. The obvious open question is whether a faster algorithm can be obtained for the SDET problem; a potential improvement would arise if the set of edges contributing candidate roots for the sought shortest paths tree could be narrowed to only $o(m)$ edges.

References

1. H.S. Connamacher and A. Proskurowski, "The complexity of minimizing certain cost metrics for k-source spanning trees", *Discrete Applied Mathematics* **131** (2003) 113–127.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.
3. A.M. Farley, P. Fragopoulou, D.W. 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. D.S. Johnson, J.K. Lenstra, and A.H.G. Rinnoy Kan, "The complexity of the network design problem", *Networks* **8** (1978) 279–285.
6. D.W. Krumme and P. Fragopoulou, "Minimum eccentricity multicast trees", *Discrete Mathematics and Theoretical Computer Science* **4** (2001) 157–172.
7. B. McMahan and A. Proskurowski, "Multi-source spanning trees: algorithms for minimizing source eccentricities", *Discrete Applied Mathematics* **137** (2004) 213–222.
8. R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi, "Spanning trees - short or small", *SIAM Journal of Discrete Mathematics* **9** (1996) 178–200.
9. 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** (1997) 346–356.
10. 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", *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms – SODA'98* (1998) 21–32.