

New algorithms for computing Steiner trees for a fixed number of terminals

Benny Kimelfeld and Yehoshua Sagiv

*The Selim and Rachel Benin School of Engineering and Computer Science
Edmond J. Safra Campus
The Hebrew University
Jerusalem 91904, ISRAEL*

Abstract

New algorithms for computing Steiner trees for a fixed number of terminals are presented. For the undirected Steiner-tree problem, an improvement of the algorithm of Dreyfus and Wagner is presented. The improved algorithm avoids the computation of all shortest paths between pairs of nodes and, hence, reduces the running time. Thus, the running time of our algorithm is better than that of existing algorithms in the literature. In addition, an adaptation of this algorithm to the directed and the group variants of the Steiner-tree problem is presented.

1 Preliminaries

A *graph* G consists of a set $\mathcal{V}(G)$ of *nodes* and a set $\mathcal{E}(G)$ of *edges*. Unless explicitly stated otherwise, edges are directed, i.e., an edge is a pair (n_1, n_2) of nodes. The *weight* function w_G assigns a positive weight $w_G(e)$ to every edge $e \in \mathcal{E}(G)$. The weight of the data graph G , denoted by $w(G)$, is the sum of the weights of all its edges, i.e., $w(G) = \sum_{e \in \mathcal{E}(G)} w_G(e)$. In the examples of this paper, we assume that all edges have an equal weight and, hence, that weight is not shown explicitly.

An *undirected subtree* of a graph G is a connected subgraph without cycles, assuming that edges are undirected. A *directed subtree* T of G has a node r , called the *root*, such that every other node of T is reachable from r through a unique directed path.

Let G be a directed graph and U be a subset of the k nodes of G . A *directed Steiner-tree* of U in G is a directed subtree T of G , such that T contains U and the weight of T is minimal, i.e., the weight of every directed subtree that

<p>Algorithm: DIRECTEDSTEINERWEIGHT</p> <p>Input: A directed graph G, a set U of nodes in G</p> <p>Returns: Array \mathcal{M}</p> <p>1: for all nodes $v \in \mathcal{V}(G)$ do</p> <p>2: for all subsets $W \subseteq U$ do</p> <p>3: $\mathcal{M}(v, W) := \infty$</p> <p>4: $\mathcal{M}(v, \emptyset) := 0$</p> <p>5: for all nodes $u \in U$ do</p> <p>6: $\mathcal{M}(u, \{u\}) := 0$</p> <p>7: for all $W \subseteq U$ in increasing size do</p> <p> /* — determine $\mathcal{M}(v, W)$ for all $v \in \mathcal{V}(G)$ — */</p> <p>8: let \mathcal{Q}_W be an empty queue of nodes v with priority $\mathcal{M}(v, W)$</p> <p>9: for all nodes $v \in \mathcal{V}(G)$ do</p> <p>10: $\mathcal{M}(v, W) := \min\{\mathcal{M}(v, Y) + \mathcal{M}(v, W \setminus Y) \mid Y \subseteq W\}$</p> <p>11: $\mathcal{Q}_W.\text{insert}(v)$</p> <p>12: while $\mathcal{Q}_W \neq \emptyset$ do</p> <p>13: $v_{\min} := \mathcal{Q}_W.\text{removeTop}()$</p> <p>14: for all edges $e = (v, v_{\min}) \in \mathcal{E}(G)$ do</p> <p>15: if $\mathcal{M}(v, W) > \mathcal{M}(v_{\min}, W) + w_G(e)$ then</p> <p>16: $\mathcal{M}(v, W) := \mathcal{M}(v_{\min}, W) + w_G(e)$</p> <p>17: return \mathcal{M}</p>
--

Fig. 1. Finding a directed Steiner tree of U in G

contains U is at least as that of T . We similarly define an *undirected Steiner tree* as a minimal-weight subtree among the undirected subtrees that contain U . Consider k subsets U_1, \dots, U_k of the nodes of G . A *group Steiner tree* of U_1, \dots, U_k in G is an undirected subtree T of G , such that T contains at least one node from each U_i and the weight of T is minimal.

We consider the problem of finding Steiner trees. Thus, there are three optimization problems that correspond to the three types of Steiner trees. Under each variant, however, it is intractable to find Steiner trees. In particular, given a graph G , a set U of k nodes of G (or k sets of nodes of G) and an integer j , it is NP-complete to decide whether G contains a Steiner tree that has at most j nodes. However, all three variants of Steiner trees can be found efficiently under the assumption that k is fixed.

2 Finding Steiner Trees

In this section, we present an algorithm for finding directed Steiner trees. Our algorithm is basically an adaptation of the algorithm of Dreyfus and Wagner [1] for finding undirected Steiner trees to the directed case. In addition, we improve the running time by avoiding the computation of all shortest paths between pairs of nodes. Thus, the running time is better than that of existing algorithms in the literature, e.g., [2].

The input of the algorithm DIRECTEDSTEINERWEIGHT of Figure 1 consists of a graph G and a set U of nodes in G . This algorithm returns an array \mathcal{M} of weights. An index of \mathcal{M} consists of a node v of G and a subset W of U . $\mathcal{M}(v, W)$ stores the weight of the minimal directed subtree T , such that (1) T contains all nodes of W and (2) T is rooted at v . If no such subtree T exists, then $\mathcal{M}(v, W)$ is ∞ . Throughout this section, we use $w_{\min}^{v,w}$ to denote this minimal weight. In particular, the weight of a directed Steiner tree of U in G is the minimal value $w_{\min}^{v,U}$ among all nodes u . As we later show, the array \mathcal{M} can easily be used for constructing a directed Steiner tree of U .

The array \mathcal{M} is initialized in Line 1–6 as follows. $\mathcal{M}(v, W) = 0$ if either $W = \emptyset$ or $W = \{v\}$ (hence $v \in U$); otherwise, $\mathcal{M}(v, W) = \infty$. Lines 8–16 are executed for every subset W of U . During this part of the computation, the correct value of $\mathcal{M}(v, W)$ is determined for all nodes v . We use a priority queue \mathcal{Q}_W that stores nodes of G . The top element in this queue is a node v for which $\mathcal{M}(v, W)$ is minimal. Lines 8–16 consist of two main parts. In Lines 9–11, each node v in G is inserted into \mathcal{Q}_W and the value $\mathcal{M}(v, W)$ is determined as the minimal value among $\mathcal{M}(v, Y) + \mathcal{M}(v, W \setminus Y)$, where Y is some subset of W . In Lines 12–16, every node v_{\min} in \mathcal{Q}_W is removed. When v_{\min} is removed, all edges $e = (v, v_{\min})$ incoming v_{\min} are considered. $\mathcal{M}(v, W)$ is then set to be the minimal between $\mathcal{M}(v, W)$ and $\mathcal{M}(v_{\min}, W) + w_G(e)$. Finally, the array \mathcal{M} is returned in Line 17.

Next, we prove the correctness of this algorithm. The following proposition can be easily proved by induction on the number of step in which $\mathcal{M}(v, W)$ is updated.

Proposition 2.1 *Throughout the execution of DIRECTEDSTEINERWEIGHT(G, U), the value of $\mathcal{M}(v, W)$ is not increasing. Furthermore, $\mathcal{M}(v, W) \geq w_{\min}^{v,W}$ always holds.*

Correctness of the algorithm follows from the next lemma.

Lemma 2.2 *When v is removed from \mathcal{Q}_W , $\mathcal{M}(v, W) = w_{\min}^{v,W}$ holds.*

Proof. If $w_{\min}^{v,W} = \infty$, then correctness of the lemma follows directly from

Proposition 2.1. So suppose that $w_{\min}^{v,W} < \infty$ and let $T_{\min}^{v,W}$ be a directed subtree of G that is rooted at v , weights $w_{\min}^{v,W}$ and has a minimal number of nodes. We will prove the correctness of the lemma by induction, first on the size of W and then on the number of nodes in $T_{\min}^{v,W}$. For the basis of the induction, suppose that $T_{\min}^{v,W}$ contains no edges at all. Then either $W = \{v\}$ or $W = \emptyset$. In this case, $\mathcal{M}(v, W) = w_{\min}^{v,W}$ clearly holds due to the initialization of Lines 1–6 and Proposition 2.1.

Now suppose that $T_{\min}^{v,W}$ has least one edge. We consider two cases. In the first case, $T_{\min}^{v,W}$ has a proper subtree T' , such that T' contains all the nodes of W . Since $T_{\min}^{v,W}$ is minimal, v has exactly one child in $T_{\min}^{v,W}$. Let u be that child. Let T_u be the subtree of $T_{\min}^{v,W}$ that is obtained by removing the edge from v to u . Clearly, T_u is a minimal rooted subtree of G that contains W and is rooted at u . Since T_u contains fewer nodes than $T_{\min}^{v,W}$, the induction hypothesis implies that, when u is removed from the \mathcal{Q}_W , $\mathcal{M}(u, W) = w_{\min}^{u,W}$ holds. It is easy to see that elements of \mathcal{Q}_W are removed in increasing order of $\mathcal{M}(\cdot, W)$. Therefore, when u is removed from \mathcal{Q}_W and Lines 14–16 are executed, the value $\mathcal{M}(v, W)$ is at most $\mathcal{M}(u, W) + w_G((v, u)) = w_{\min}^{v,W}$.

In the second case, no proper subtree of $T_{\min}^{v,W}$ contains W . In this case, it can easily be shown that there are two subtrees T_1 and T_2 of $T_{\min}^{v,W}$ that satisfy the following. First, $T_1 \cup T_2 = T_{\min}^{v,W}$. Second, v is the only node that is in both T_1 and T_2 . Third, both T_1 and T_2 contain nonempty subsets W_1 and W_2 of W (and possibly other nodes of W), respectively, such that $W_1 = W \setminus W_2$. It follows from the induction hypothesis that, when v is removed from \mathcal{Q}_{W_i} , $\mathcal{M}(v, W_i) = w_{\min}^{v,W_i}$ holds. Since the order of the iteration of Line 7 is by increasing size of W , both $\mathcal{M}(v, W_1) = w_{\min}^{v,W_1}$ and $\mathcal{M}(v, W_2) = w_{\min}^{v,W_2}$ hold when $\mathcal{M}(v, W)$ is considered. We conclude that, after Line 10 is executed for W and v , $\mathcal{M}(v, W) \leq \mathcal{M}(v, W_1) + \mathcal{M}(v, W_2) (\leq w(T_{\min}^{v,W}))$ holds, hence $\mathcal{M}(v, W) = w_{\min}^{v,W}$, as claimed. \square

Next, we analyze the running time of DIRECTEDSTEINERWEIGHT(G, U). We use n and e to denote the number of nodes and edges of G , respectively. We assume that G is a connected graph (otherwise, we consider only the connected component that contains U). Therefore, $e \geq n - 1$. We use k to denote the number of nodes in U .

Lemma 2.3 DIRECTEDSTEINERWEIGHT(G, U) *terminates in* $\mathcal{O}(3^k n + 2^k e \log n)$ *time.*

Proof. We first analyze the execution cost of Lines 8–16 for a set W of size j . The execution cost of Line 10 is $\mathcal{O}(2^j)$. The execution cost of Line 11 is $\mathcal{O}(\log n)$. Therefore, Lines 9–11 cost $\mathcal{O}(2^j n + n \log n)$ time. In Lines 12–16 we remove n nodes from \mathcal{Q}_W and update \mathcal{Q}_W (in Line 16) e times. Therefore, the execution cost of Lines 12–16 is $\mathcal{O}(e \log n)$. We conclude Lines 8–16 cost

$\mathcal{O}(2^j n + e \log n)$.

Clearly, the running time is dominated by the execution cost of Lines 7–16. Let $f(G, U)$ denote this cost. Then,

$$\begin{aligned} f(G, U) &= \mathcal{O} \left(\sum_{j=0}^k \binom{k}{j} (2^j n + e \log n) \right) = \\ &= \mathcal{O} \left(n \sum_{j=0}^k \binom{k}{j} 2^j + e \log n \sum_{j=0}^k \binom{k}{j} \right) = \mathcal{O} (3^k n + 2^k e \log n), \end{aligned}$$

as claimed. \square

Next, we describe how we can obtain from \mathcal{M} a minimal subtree $T_{\min}^{v,W}$ of G that contains a set $W \subseteq U$ and is rooted at a node v . For each outgoing edge $e = (v, u)$, we test whether $\mathcal{M}(v, W) = w_G(e) + \mathcal{M}(u, W)$. If so, we recursively construct the tree $T_{\min}^{u,W}$ and add the edge e . Otherwise, we find a subset $\emptyset \neq Y \subsetneq W$, such that $\mathcal{M}(v, W) = \mathcal{M}(v, Y) + \mathcal{M}(v, W \setminus Y)$. We then recursively construct $T_{\min}^{v,Y}$ and $T_{\min}^{v,W \setminus Y}$ and concatenate the two subtrees. Finally, to obtain a directed Steiner tree of U in G , we choose the node v for which $\mathcal{M}(v, W)$ is minimal and construct the subtree $T_{\min}^{v,U}$. Note that an alternative approach to the above tree construction is to store, during the execution of DIRECTEDSTEINERWEIGHT, information about the decisions made when updating \mathcal{M} .

Using the algorithm DIRECTEDSTEINERWEIGHT, we can find undirected and group Steiner trees by reducing these two problems to that of finding directed Steiner trees. For both problems, we add a back edge (u, v) for every edge (v, u) in the graph (the weight of (u, v) is that of (v, u)). Then, in order to find an undirected Steiner tree of U , we find a directed Steiner tree of U in the new graph. For finding a group Steiner tree, we also add a node u_i for every set U_i and a zero-weight edge from every $v \in U_i$ to the node u_i . We then find a directed Steiner tree of $\{u_1, \dots, u_k\}$ in the new graph. It can also be shown that the algorithm DIRECTEDSTEINERWEIGHT can be adapted to find undirected and group Steiner trees without adding new edges and nodes. This way, Steiner trees are found more efficiently than the above reductions. The following theorem summarizes all the above.

Theorem 2.4 *Let G be a directed graph, U be a set of k nodes of G and U_1, \dots, U_k be k sets of nodes of G . The following can be found in $\mathcal{O}(3^k n + 2^k e \log n)$ time.*

- A directed Steiner tree of U in G .
- An undirected Steiner tree of U in G .
- A group Steiner tree of U_1, \dots, U_k in G .

References

- [1] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.
- [2] J. Feldman and M. Ruhl. The directed steiner network problem is tractable for a constant number of terminals. In *FOCS*, pages 299–308, 1999.