Steiner Problems on Directed Acyclic Graphs

Tsan-sheng Hsu[†], *Kuo-Hui Tsai*[†‡], *Da-Wei Wang*[†‡] and *D. T. Lee*[⋆]

November 3, 1995

Abstract

In this paper, we consider two variations of the minimum-cost Steiner problem on a directed acyclic graph $G(V, E)$ with a non-negative weight on each edge of E. The *minimum directed Steiner network* problem is defined as follows. Given a set of starting vertices $S \subset V$ and a set of terminating vertices $T \subset V$, find a subgraph with the minimum total edge weight such that for each starting vertex s there exists a path from s to a terminating vertex, and for each terminating vertex t, there exists a path from a starting vertex to t. The minimum union paths problem is similar to the minimum directed Steiner network problem except that we are given a set of hitting vertices $H \subset V$, in addition to the sets of starting and terminating vertices and that we want to find a subgraph with the minimum total edge weight such that the subgraph satisfies not only the conditions above, but also that every hitting vertex is on a path from a starting vertex to a terminating vertex.

We present algorithms for finding *optimal* solutions to these two problems in time, respectively, $O(n \cdot m + 2^{|S|+|T|} \cdot \alpha^3 \cdot (n^{\alpha-2} + n^{\beta-1}))$ and $O(k! \cdot (8 \cdot k)^k \cdot k^3 \cdot n^{k-1} + n \cdot m)$, where n and m denote the number of vertices and edges of the graph, $\alpha = \max\{|S|, |T|\}$, $\beta = \min\{|S|, |T|\}$, and $k = |S| + |T| + |H|$. The algorithms can also enumerate all possible optimal solutions for both problems. Our algorithm for the *minimum directed Steiner network* problem can also be used on undirected graphs.

We also give linear time algorithms for some special cases.

[†]Institute of Information Science, Academia Sinica, Nankang 11529, Taipei, Taiwan, ROC. E-mail: {tshsu, tsaikh, wdw}@iis.sinica.edu.tw.

[‡]Supported in part by National Science Council, Taiwan, ROC, under the Grant No. NSC-83-0408-E-001-021.

[⋆]Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA. E-mail: dtlee@dtlee.eecs.nwu.edu. Supported in part by the National Science Foundation under the Grants CCR-9309743 and INT-9207212, and by the Office of Naval Research under the Grant No. N00014-93-1-0272.

1 Introduction

Given a graph $G = (V, E)$, a non-negative cost on each edge in E, and a set of vertices $Z \subset V$, the *minimum Steiner problem* is to find a minimum cost subgraph with a given property which spans Z. The vertices besides Z in the subgraph are *Steiner vertices*. For example, the *minimum Steiner tree problem* is one in which the subgraph is a *tree*. The minimum Steiner problem has many important applications and has been extensively studied [2, 7, 8, 14, 15]. In one of the latest survey papers [7], more than 300 references are listed. Let S and T be two subsets of vertices of V. A subgraph G' of G is said to be *S,T-connected* if for each vertex $s \in S$ there exists a path[1] in G' from s to a vertex $t \in T$, and for each vertex $t \in T$ there exists a path in G' from a vertex $s \in S$ to t.

We consider the following two variations of the minimum Steiner problem on directed acyclic graphs $G(V, E)$ with a non-negative weight on each edge of E. The *minimum directed Steiner network* problem is defined as follows. Given a set of starting vertices S and a set of terminating vertices T, we want to find a subgraph G^* with the minimum total edge weight such that G^* is S,T-connected. The *minimum union paths* problem is a more general version of the minimum directed Steiner network problem. An additional set of *hitting* vertices $H \subset V$ is specified and the subgraph G^* is required to contain H. Note that the minimum directed Steiner network problem is a special case of the minimum union paths problem, in which the hitting set H is empty. It can be easily shown that the above two minimum Steiner problems on directed acyclic graphs are NP-hard by a polynomial transformation from the exact cover by 3-sets problems (see e.g., the discussion in Chapter 1 of [8]).

Several important applications for solving the minimum Steiner problem on a directed acyclic graph are mentioned in [11] for the case of finding a minimum Steiner arborescence. The two problems defined here can be used to solve generalized versions of these problems. For example, finding a minimum directed Steiner network can be used to design an optimal drainage system for a building, where starting vertices are places to dump wastes and terminating vertices are sewers. Finding a minimum union paths can be used to design an optimal system layout for an irrigation and drainage system where starting vertices are water sources, hitting vertices are places that need water, and terminating vertices are drains.

There are several types of algorithms that can be used to find an exact solution of the minimum Steiner problem which include dynamic programming [3], 0-1 linear programming [1], and exhaustive enumeration [9]. Though most of the discussions are on undirected graphs, they can be easily extended to directed graphs. For example, a typical exhaustive enumeration algorithm for the minimum Steiner tree problem makes use of the fact that

[1] In this paper, we refer a *path* as a sequence of vertices $a_1, a_2, \ldots, a_k$ such that (a_i, a_{i+1}), $1 \leq i < k$, is a directed edge pointed from a_i to a_{i+1} and $a_i \neq a_j$ for all $i \neq j$, i.e., every path is a *simple path*.

a minimum Steiner tree is a minimum spanning tree on the set of vertices in the solution. Thus by properly choosing the set of Steiner vertices, we can easily find the solution by computing a minimum spanning tree. Thus fining an accurate upper bound on the number of Steiner vertices is crucial in estimating the running time of this type of algorithms. Let the upper bound be k. Then the time complexity of the algorithm is $O(n^k \cdot MST(G))$, where $MST(G)$ denotes the time complexity of finding a minimum spanning tree of G. A similar algorithm can be devised to solve the minimum directed Steiner network problem and the minimum union paths problem in time exponential in k. We shall show that by exploiting some properties of the solutions to the two problems, we can solve them in time exponential roughly in $\frac{k}{2}$.

Some related work for the minimum Steiner problem on directed graphs can be found in [6, 10, 11, 12, 16]. In particular, Nastansky *et al.* [11] presented an efficient heuristic algorithm for solving the minimum Steiner arborescence problem on a directed acyclic graph by exhaustive enumeration. However, they do not provide any theoretical analysis on the time complexity of their algorithm, which we believe is exponential in k. Rao *et al.* [12] described an approximation algorithm for finding a minimum Steiner arborescence in a rectilinear plane rooted at the origin with the additional constraint that all paths from the root to the leaves must be shortest.

2 Minimum Directed Steiner Network

Given a directed acyclic graph $G = (V, E)$ and two non-empty disjoint subsets of vertices S and T, a *directed Steiner network* of G is a subgraph G^* of G such that G^* is S,T-connected. The vertices in S are called *starting vertices* and those in T are *terminating vertices*. Any path in a Steiner network starting from a vertex in S and ending in a vertex in T is called an *S-T path*. If the edges in G are associated with non-negative weights, then a directed *minimal* Steiner network with a minimum total edge weight is called a *minimum directed Steiner network*[2] and denoted as MSN(G, S, T). Note that in a minimum (directed) Steiner network[3], an *S-T path*, except the two extreme vertices, may contain vertices in S or T.

[2]Here we consider *the* Steiner network with the least number of edges among those with a minimum total edge weight.

[3]From now on we will omit the word "directed" without any confusion, and minimality of the network is assumed.

2.1 Preliminaries

Before we describe our algorithm for finding a minimum Steiner network, we examine its properties. Through these properties, we give a structural description of minimum Steiner networks. Then we obtain a minimum Steiner network by an efficient enumerative algorithm.

Let $S \dot{\cup} T \dot{\cup} Z$ be the set of vertices in MSN(G, S, T), where $\dot{\cup}$ is the disjoint set union operator. The vertices in Z, which contain no vertices in S and T, are *Steiner vertices*. Let $D(G)$ be the distance network for a directed acyclic graph G with non-negative edge weights, i.e., each edge (u, v) in $D(G)$ is associated with a weight equal to the total weight of a shortest path from u to v.

Lemma 2.1 *Given S and T, the cost of MSN(G, S, T) is equal to the cost of MSN$(D(G), S, T)$. Furthermore, given MSN$(D(G), S, T)$, MSN(G, S, T) can be constructed in time linear in the size of G.* □

The proof of Lemma 2.1 is quite straightforward. Let $P(u, v)$ denote a maximal path in MSN(G, S, T) connecting vertices u and v for some $u, v \in V$ such that the intermediate nodes in the path have in-degree and out-degrees equal to 1. The above lemma simply states that if we replace every such path $P(u, v)$ in MSN(G, S, T) with an edge (u, v) whose weight is equal to the total edge weight of all the edges in $P(u, v)$, then the resulting network has a cost equal to MSN$(D(G), S, T)$. Note that the set of Steiner vertices Z in MSN(G, S, T) can be partitioned into two sets $Z = Z_1 \dot{\cup} Z_2$, where Z_2 contains the intermediate vertices of all such $P(u, v)$ defined above and Z_1 will be the set of Steiner vertices in MSN$(D(G), S, T)$.

From Lemma 2.1, we know that it suffices to compute a minimum Steiner network from the distance network of the graph. It is well-known that the computation time of any algorithm to find a Steiner-tree-like solution is exponential in the number of Steiner vertices in a general graph by enumerating all possible candidates of Steiner vertices. We will focus our discussion on finding a minimum Steiner network in a distance network with an aim to show that in an acyclic graph, the possible candidates for Steiner vertices can be enumerated more efficiently.

A Steiner vertex in MSN$(D(G), S, T)$ is *convergent* if its in-degree is greater than 1, and is *divergent* if its out-degree is greater than 1. From Lemma 2.1, any Steiner vertex in a minimum Steiner network of a distance network must be either convergent or divergent. Note that a vertex may be both convergent and divergent if its in-degree and out-degree are greater than 1.

Lemma 2.2 *Given a directed acyclic graph G with non-negative edge weights and two disjoint subsets of vertices S and T, let $s \in S$, $t \in T$, and P be a path from s to t in*

MSN$(D(G), S, T)$ *whose length (number of edges) is greater than 1. Let* $P = [s, w_1, w_2, \ldots, w_q, t]$, $q \geq 1$. *Then there exists* b, $0 \leq b \leq q$, *such that exactly one of the following two conditions is true: (1)* $w_i, 1 \leq i \leq b$, *is convergent and* $w_i, b \leq i \leq q$, *is divergent; (2)* $w_i, 1 \leq i \leq b$, *is convergent and* $w_i, b+1 \leq i \leq q$, *is divergent.*

Proof: We will prove it by contradiction. Recall that MSN$(D(G), S, T)$ has a minimal number of edges. Let b be the largest index such that $w_1, \ldots, w_{b-1}$, and w_b are convergent in P. Assume that $w_{b+1} \neq t$ is *strictly* divergent, i.e., not convergent, and that w_a is convergent, where $a > b+1$. Let H be the resulting graph obtained by removing from P, and hence from MSN$(D(G), S, T)$, the edge from w_{a-1} to w_a. Note that H remains a Steiner network. Since the weight of each edge is non-negative, either the cost of H is less than that of MSN$(D(G), S, T)$ or H has fewer edges. Thus MSN$(D(G), S, T)$ is not optimal, which is a contradiction. □

The vertex w_b in the path P as specified in Lemma 2.2 is referred to as a *neutral* vertex. Note that if a vertex is neutral in an S-T path, then it is neutral in any other S-T path.

Corollary 2.3 *There exists at most one neutral vertex in the path* P *as specified in Lemma 2.2.* □

A directed graph is an *incoming arborescence* if it is an isolated vertex or there exists exactly one *root vertex* u such that there is exactly one path from any other vertex to u. A collection of incoming arborescences is an *incoming forest*. A directed graph is an *outgoing arborescence* if it is an isolated vertex or there exists exactly one root vertex u such that there is exactly one path from u to any other vertex in the graph. A collection of outgoing arborescences is an *outgoing forest*. A vertex is a *leaf* in an incoming (outgoing) forest if its in-degree (out-degree) is zero. A vertex in a forest that is not a leaf is an *internal vertex*. Given a minimum Steiner network MSN$(D(G), S, T)$, let its *neutral split network* be defined as follows — for each neutral vertex u in MSN$(D(G), S, T)$, we replace u with two vertices u_1 and u_2 such that u_1 inherits all incoming edges of u and u_2 inherits all outgoing edges of u. The resulting network is called a neutral split network. It is easy to see that the cost of the neutral split network is the same as the cost of the original minimum Steiner network. Figure 1 illustrates a minimum Steiner network, and its neutral split network. The set of neutral vertices are shown in solid circles, and the others shown in hollow circles.

Lemma 2.4 *There are at most* $|S| + |T| - 2$ *Steiner vertices in MSN*$(D(G), S, T)$, *and the bound is tight.*

Proof: We prove this lemma by induction on $|S| + |T|$. This lemma is obviously true when $|S| = 1$ and $|T| = 1$. For the induction step, observing that adding a vertex to S or T

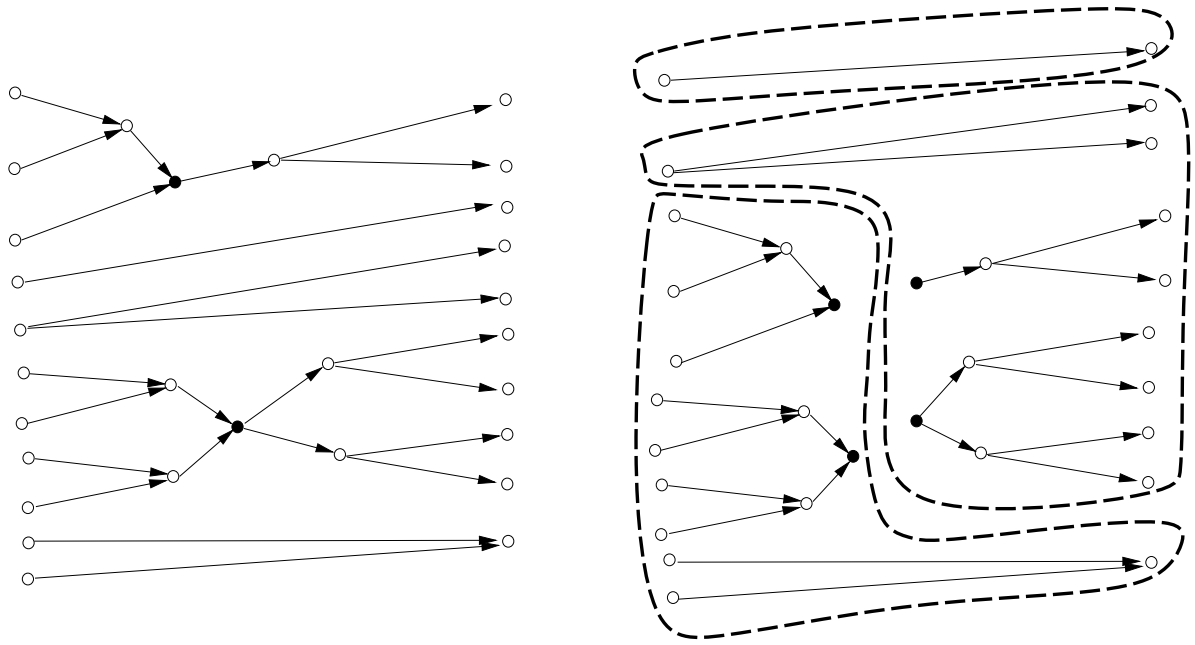

Figure 1: A minimum Steiner network, the set of neutral vertices shown in solid circles, and its neutral split network.

creates at most one Steiner vertex in $D(G)$, by converting a vertex in $\mathrm{MSN}(G, S, T)$ which is in Z_2 into one which is in Z_1. To prove that the bound is tight let us construct, as described below, a graph G such that $\mathrm{MSN}(D(G), S, T)$ has exactly $|S| + |T| - 2$ Steiner vertices.

Let B_S be an incoming arborescence such that the root is u, the set of leaves is S and the in-degree of each node except the leaves (each of whose in-degree is 0) is 2 and out-degree is 1, except the root (whose out-degree is 0). For convenience B_S is called an incoming binary tree. Let B_T be similarly defined except that it is an outgoing arborescence (binary tree) and whose root is v and the set of leaves is T. Then $G = B_S \cup B_T \cup \{(u, v)\}$ is a graph whose $\mathrm{MSN}(D(G), S, T)$ contains $|S| + |T| - 2$ Steiner vertices. □

Lemma 2.5 *The neutral split network of a minimum Steiner network $MSN(D(G), S, T)$ for an acyclic graph G with non-negative edge weights can be partitioned into three disjoint subgraphs N_u, N_{in}, and N_{out} where (1) N_u consists of edges from a vertex in S to a vertex in T; (2) N_{in} is an incoming forest where each leaf is a vertex in S, each non-root internal vertex is a Steiner vertex, and each root is either a Steiner vertex or a vertex in T; (3) N_{out} is an outgoing forest where each leaf is a vertex in T, each non-root internal vertex is a Steiner vertex, and each root is either a Steiner vertex or a vertex in S. There are no isolated vertices in N_{in} and N_{out}.*

Proof: Since G is acyclic, $\mathrm{MSN}(D(G), S, T)$ is also acyclic. Every vertex in $\mathrm{MSN}(D(G), S, T)$ is in an S-T path in $\mathrm{MSN}(G, S, T)$. Thus this lemma follows from Lemma 2.2 and Corollary 2.3. □

Corollary 2.6 *The number of arborescences in N_{in} (N_{out}) is no more than half of the number of leaves in N_{in} (respectively, N_{out}).*

Proof: There is no isolated vertex in N_{in} or N_{out}. In each arborescence, there are at least two leaves. Hence the corollary holds. □

Now we examine properties of the three disjoint subgraphs in a neutral split network. Let u be a vertex not in the neutral split network of $\mathrm{MSN}(D(G), S, T)$. Let N'_{in} be an incoming arborescence by adding edges to N_{in} from each root in N_{in} to u. Let $D(G)'$ be the resulting graph obtained by adding the same set of edges to $D(G)$. The weight of each added edge is zero.

Lemma 2.7 *The graph N'_{in} is a minimum Steiner incoming arborescence in $D(G)'$ and can be obtained in $O(n \cdot m + \ell^2 \cdot n^{\ell - t})$ time, where $\ell > 2$ is the number of leaves in N_{in}, n the number of vertices in G, m the number of edges in G, and t the number of trees in N_{in}.*

Proof: It takes $O(n \cdot m)$ time to find the distance network for a directed acyclic graph [13]. Note that the distance network contains $O(n^2)$ edges. The minimum spanning arborescence in a graph with x vertices and y edges can be found in $O(x \cdot \log x + y)$ time [5]. There are less than $\ell - t$ Steiner vertices in a minimum Steiner arborescence. □

It is easy to see that given N'_{in}, N_{in} can be obtained in time linear in the number of vertices in N_{in}. It is also true that N_{in} is a *minimum Steiner incoming arborescence* in $D(G)$.

Let N'_{out} be an outgoing arborescence by adding edges to N_{out} from u to each root in N_{out}. Let $D(G)''$ be the resulting graph obtained by adding the same set of edges to $D(G)$ as in the previous construction. The weight of each added edge is zero.

Lemma 2.8 *N'_{out} is a minimum Steiner outgoing arborescence in $D(G)''$ and can be obtained in $O(n \cdot m + \ell^2 \cdot n^{\ell - t})$ time, where $\ell > 2$ is the number of leaves in N_{out}, n the number of vertices in G, m the number of edges in G, and t the number of trees in N_{out}.* □

It is easy to see that given N'_{out}, N_{out} can be obtained in time linear in the number of vertices in N'_{out}. It is also true that N_{out} is a *minimum Steiner outgoing arborescence* in $D(G)$.

Corollary 2.9 *Let ℓ_{in} be the number of leaves in N_{in} and let ℓ_{out} be the number of leaves in N_{out}.*
(1) The number of internal vertices in N_{in} is less than ℓ_{in}.
(2) The number of internal vertices in N_{out} is less than or equal to ℓ_{out}.

Proof: The degree of each internal vertex in N_{in} is at least 2. Thus (1) holds by using a simple induction argument. Note that the degree of the root in a tree in N_{out} could be one. Thus (2) holds by using a simple induction argument. □

2.2 The Algorithm

Using Lemmas 2.5, 2.7, and 2.8, we can construct a minimum Steiner network as follows. First we obtain (by exhaustive enumeration) $S_1 \subseteq S$ and $T_1 \subseteq T$ such that $|S_1| = |T_1|$ and construct N_u by finding a minimum-cost bipartite perfect matching between S_1 and T_1 in $D(G)$. We then choose the set of roots $\mathcal{R}$ in the neutral split network of $\mathrm{MSN}(D(G), S, T)$. Note that $\mathcal{R}$ consists of $\mathcal{R}_1 \subseteq V$, $\mathcal{R}_2 \subseteq (S \setminus S_1)$, and $\mathcal{R}_3 \subseteq (T \setminus T_1)$.

After choosing $\mathcal{R}$, we find a minimum Steiner incoming forest N_{in} on the set of leaves $S \setminus (S_1 \cup \mathcal{R})$ and the set of roots $\mathcal{R} \setminus \mathcal{R}_2$. By Corollary 2.9, there can be at most $|S| - |S_1| - 1$ Steiner vertices (including vertices in $\mathcal{R}$). We also find a minimum Steiner outgoing forest

N_{out} on the set of leaves $T \setminus (T_1 \cup \mathcal{R})$ and the set of roots $\mathcal{R} \setminus \mathcal{R}_3$. By Corollary 2.9, there can be at most $|T| - |T_1|$ Steiner vertices (including vertices in $\mathcal{R}$). Note that the sets of Steiner vertices in N_{in} and in N_{out} are disjoint. By properly choosing S_1, T_1, and $\mathcal{R}$, the cost of MSN$(D(G), S, T)$ is equal to the cost of $N_u \cup N_{in} \cup N_{out}$. It is also easy to construct MSN$(D(G), S, T)$ from N_u, N_{in}, and N_{out}. From the above discussion, we have the following theorem.

Theorem 2.10 *Let $\alpha = \max\{|S|, |T|\}$ and let $\beta = \min\{|S|, |T|\}$. A minimum Steiner network MSN(G, S, T) can be constructed in $O(n \cdot m + 2^{|S|+|T|} \cdot \alpha^3 \cdot (n^{\alpha-2} + n^{\beta-1}))$ time, where n and m are the number of vertices and edges in G, respectively.*

Proof: Assume without loss of generality that $\alpha = |S|$ and $\beta = |T|$, since otherwise we can reverse the direction of each edge in G to satisfy this condition. Note that $|S_1| = |T_1| \leq \beta$. Our algorithm first constructs the distance network. Then we enumerates all candidates for a directed minimum Steiner network in the distance network. Given a candidate, assume that there are i pairs of vertices in N_u. Note that $i \leq \beta$. There are up to $\begin{pmatrix} |S| \\ i \end{pmatrix} \begin{pmatrix} |T| \\ i \end{pmatrix}$ candidates for the set of vertices in N_u. Let $|\mathcal{R}| = j$. By Corollary 2.6, $j \leq \left\lfloor \frac{\alpha - i}{2} \right\rfloor + \left\lfloor \frac{\beta - i}{2} \right\rfloor$, which is no more than $\alpha - i$. There are up to $\begin{pmatrix} n \\ j \end{pmatrix}$ candidates for the set of vertices $\mathcal{R}$. Once we pick $\mathcal{R}$, we partition $\mathcal{R}$ into $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$, where $\mathcal{R}_1 = \mathcal{R} \setminus (S \cup T)$, $\mathcal{R}_2 = \mathcal{R} \cap S$, and $\mathcal{R}_3 = \mathcal{R} \cap T$. Thus the set of vertices in N_{in} is $\mathcal{R}_1 \cup \mathcal{R}_3 \cup (S \setminus \mathcal{R}_2)$ and the set of vertices in N_{out} is $\mathcal{R}_1 \cup \mathcal{R}_2 \cup (T \setminus \mathcal{R}_3)$. Thus our algorithm takes time

$$O((A) + \sum_{i=1}^{\beta} \begin{pmatrix} |S| \\ i \end{pmatrix} \begin{pmatrix} |T| \\ i \end{pmatrix} \sum_{j=1}^{\alpha - i} \begin{pmatrix} n \\ j \end{pmatrix} [(B) + (C) + (D)]),$$

where n is the number of vertices in $D(G)$, (A) is the time needed to construct the distance network from G and is equal to $O(n \cdot m)$ for a directed acyclic graph [13], (B) is the time needed to find a minimum cost bipartite perfect matching in a bipartite graph with $2 \cdot i$ vertices and $O(i^2)$ edges and is $O((2 \cdot i)^2 \log(2 \cdot i) + (2 \cdot i) \cdot i^2)$ [4], (C) is the time needed to find a minimum Steiner incoming forest with j roots, $|S| - i$ leaves, and up to $|S| - i - j - 1$ internal vertices given $D(G)$, and (D) is the time needed to find a minimum Steiner outgoing forest with j roots, $|T| - i$ leaves, and up to $|T| - i - j$ internal vertices given $D(G)$. By Lemmas 2.7 and 2.8, we know that (C) is $O(|S|^2 \cdot n^{|S|-i-j-1})$ and (D) is $O(|T|^2 \cdot n^{|T|-i-j})$. Thus (B)+(C)+(D) = $O(\alpha^2 \cdot (n^{\alpha-i-j-1} + n^{\beta-i-j}))$.
Since

$$\begin{pmatrix} n \\ j \end{pmatrix} \leq n^j, \begin{pmatrix} n \\ j \end{pmatrix} ((B) + (C) + (D)) = O(\alpha^2 \cdot (n^{\alpha-i-1} + n^{\beta-i})).$$

Thus

$$\sum_{j=1}^{\alpha-i} \binom{n}{j} ((B)+(C)+(D))) = O(\alpha^3 \cdot (n^{\alpha-i-1} + n^{\beta-i})).$$

Since

$$\sum_{i=1}^{\beta} \binom{|S|}{i} \binom{|T|}{i} \leq \sum_{i=1}^{\beta} \binom{\beta}{i} \sum_{i=1}^{\alpha} \binom{\alpha}{i} \text{ and } \sum_{i=1}^{\alpha} \binom{\alpha}{i} = 2^{\alpha},$$

$$\sum_{i=1}^{\beta} \binom{|S|}{i} \binom{|T|}{i} = O(2^{\alpha} \cdot 2^{\beta}).$$

Thus the overall time complexity is $O(n \cdot m + 2^{|S|+|T|} \cdot \alpha^3 \cdot (n^{\alpha-2} + n^{\beta-1}))$. □

By Lemma 2.4, the number of Steiner vertices in a minimum Steiner network can be as large as $|S| + |T| - 2$. Thus finding a minimum Steiner network by brute force may need to perform a minimum spanning arborescence computation $O(n^{|S|+|T|-2})$ times. Following from Theorem 2.10 a minimum Steiner network can be found by performing a minimum spanning arborescence computation $O(2^{|S|+|T|} \cdot \max\{|S|, |T|\} \cdot (n^{\max\{|S|,|T|\}-2} + n^{\min\{|S|,|T|\}-1}))$ times. This is a substantial saving in computation time if $\max\{|S|, |T|\}$ is $O(\log n)$.

Remark: We can solve within the same time complexity the *strong minimum directed Steiner network* problem in which no S-T path contains vertices in S and T as intermediate node. In other words, all starting vertices are of in-degree 0 (sources) and all terminating vertices are of out-degree 0 (sinks). This problem can be solved by finding $\text{MSN}(G', S, T)$ where G' is obtained by removing all incoming edges to a starting vertex and all outgoing edges to a terminating vertex.

We can also solve within the same time complexity the *minimum Steiner network* problem on an undirected graph in which each S-T path in the solution is undirected by applying the algorithm in Section 2.2. In using the algorithm, we find undirected paths and forests. The correctness of the algorithm which is established in properties proved in Section 2.1 for the directed case can be applied on undirected graphs as well. Observing that a solution for an undirected minimum Steiner network contains no cycles. Thus there is an unique orientation for the undirected solution by orienting edges in every S-T path from a vertex in S towards a vertex in T. All properties in Section 2.1 can be applied on this oriented solution.

2.3 Algorithm for a Special Case

Note that if $\min\{|S|, |T|\} = 1$, then the problem of finding a minimum Steiner network reduces to the problem of finding a minimum Steiner arborescence. Let (i, j)-minimum

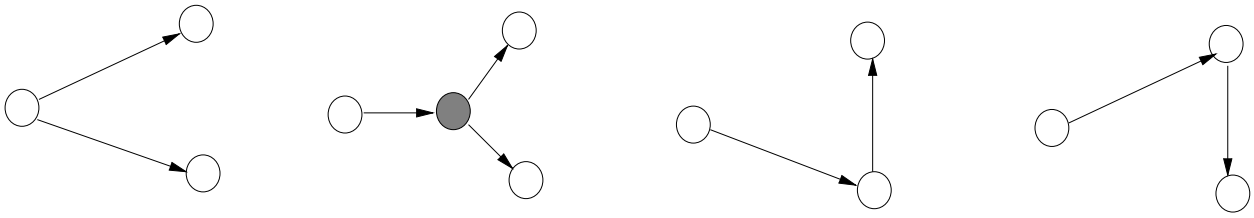

Figure 2: All possible configurations for a solution of the (1,2)-minimum Steiner network problem in the distance network where the starting vertex is s and the set of terminating vertices is $\{t_1, t_2\}$. The shaded vertex is a Steiner vertex.

Steiner network problem denote the minimum Steiner network problem with i starting vertices and j terminating vertices. In Figure 2, we list the four possible configurations for the (1,2)-minimum Steiner network problem.

For the minimum Steiner arborescence problem in which $\max\{|S|, |T|\} = 2$ we have the following result which shows that the computation of the distance network can be avoided when $|S \cup T| = 3$.

Theorem 2.11 *If* $\min\{|S|, |T|\} = 1$ *and* $\max\{|S|, |T|\} = 2$, *then MSN*(G, S, T) *can be computed in* $O(n + m)$ *time (instead of* $O(n \cdot m)$ *time).*

Proof: Note that there are at most one Steiner vertex in MSN$(D(G), S, T)$. Any path between any two vertices u and v in MSN(G, S, T) is also a shortest path between u and v in G. Assume without loss of generality that $|S| = 1$ and $|T| = 2$. Our algorithm first computes the single source shortest path (in G) for the vertex in S. Let $r(G)$ be the resulting graph obtained by reversing the direction of each edge in G. We compute the single source shortest paths (in $r(G)$) for the two vertices in T. The above computation takes $O(m)$ time for a directed acyclic graph.
We distinguish two cases.

Case 1: There is no Steiner vertex.
The cost of a minimum Steiner network is equal to the sum of edge-costs in the two paths from the vertex in S to the two vertices in T (Figure 2.(1)), or in the *one* path from the vertex in S to the two vertices in T (Figures 2.(3) and 2.(4)).

Case 2: There is exactly one Steiner vertex u.
Note that u could be a vertex in T and that this case reduces to those shown in Figures 2.(3) and 2.(4). We can try all possible candidates for u. Whenever we fix a

candidate for u, the cost of the Steiner network with only one Steiner vertex u can be computed in constant time by adding the shortest path distance (in G) from the vertex in S to u, and the costs of the two shortest paths (in $r(G)$) from the two vertices in T to u.

One of the above solutions with the minimum cost has the same cost as a minimum Steiner network. The corresponding network can also be constructed in linear time. □

Remark: We can also solve in linear time the *minimum Steiner network* problem on an undirected graph in which each S-T path in the solution is undirected by using the same algorithm.

3 The Minimum Union Paths Problem

Given a weighted directed acyclic graph $G = (V, E)$ and three non-empty mutually disjoint subsets of vertices S, H, and T, a set of *S-H-T union paths* of G is a subgraph G' of G with the following properties:

1. For every $s \in S$ there is a path in G' from s to a vertex in T.
2. For every $t \in T$ there is a path in G' from a vertex in S to t.
3. For every $h \in H$ there is a path in G' passing through h which starts from some vertex in S and ends with some vertex in T.

The set of vertices S is referred to as *starting vertices*; the set of vertices H *hitting vertices* and the set of vertices T *terminating vertices*. The *minimum union paths* problem (or *minimum directed Steiner network problem with a hitting set*) is to find a set of *S-H-T minimal* union paths in G with a minimum total edge weight,[4] A set of minimum union paths with the minimum total edge weight is denoted $\mathrm{MUP}(G, S, H, T)$.

Lemma 3.1 *There are at most* $|S|+|T|+2\cdot|H|-4$ *Steiner vertices in* $MUP(D(G), S, H, T)$ *for* $|H| \geq 1$.

Proof: We prove this lemma by induction on $|S| + |T| + 2 \cdot |H|$. This lemma is obviously true when $|S| = 1$, $|T| = 1$ and $|H| = 1$. For the induction step, we observe that adding a

[4]Here we also consider the network with the least number of edges among those that have a minimum total edge weight.

vertex to S or T creates at most one Steiner vertex and that adding a vertex to H creates at most two Steiner vertices. Hence the lemma is true. □

Similar to Lemma 2.1 we have the following.

Lemma 3.2 *MUP(G, S, H, T) can be constructed from MUP$(D(G), S, H, T)$ in linear time.*

Without loss of generality we may assume that $|S| = |T| = 1$. To see this we can augment the given graph by adding two new vertices s_0 and t_0 not in V and the set of edges $E^* = \{(s_0, s) \mid s \in S\} \cup \{(t, t_0) \mid t \in T\}$ whose cost is an arbitrary positive constant. Let the augmented graph be denoted by $\tilde{G}$. From MUP$(\tilde{G}, \{s_0\}, S \cup H \cup T, \{t_0\})$ we can obtain the solution MUP(G, S, H, T) by deleting the vertices s_0 and t_0 and all their incident edges.

From here on we assume that both S and T are singleton sets and that we consider the distance network $D(G)$ of G.

3.1 Preliminaries

Before we present a solution to MUP$(D(G), \{s\}, H, \{t\})$, we first discuss some properties of an optimal solution. Given MUP$(D(G), \{s\}, H, \{t\})$, we first observe that s has no incoming edge and t has no outgoing edge. Let L_i denote the subset of vertices in H such that for every vertex $v \in L_i$ there is a path from s to v containing at most i vertices in H. Note that when we traverse any path from s to t in MUP$(D(G), \{s\}, H, \{t\})$, the first vertex in H encountered is in L_1. Let ℓ be the smallest integer such that $L_\ell = L_{\ell+1}$. Then MUP$(D(G), \{s\}, H, \{t\})$ has ℓ levels.

According to the definition of L_i, we partition H as follows. Let $H_1 = L_1$ and $H_i = L_i \setminus L_{i-1}$ for all $2 \leq i \leq \ell$. Let $H_0 = \{s\}$ and let $H_{\ell+1} = \{t\}$.

Lemma 3.3 *It is not possible to have a path in MUP$(D(G), \{s\}, H, \{t\})$ from a vertex in H_i to a vertex in H_j, if $i + 1 < j$ and $j \neq \ell + 1$.* □

A vertex $v \in H_i$ is *backward* if there is a path from v to some vertex $v' \in H_j$, $j \leq i$, without going through any other vertex in H. The vertex v' is a *backward successor* of v. A vertex $v \in H_i$ is *forward* if there is a path from v to t without going through any other vertex in H. The path which makes the vertex forward (respectively, backward) is a *forward* (respectively, *backward*) *path*. The following lemma states that a vertex cannot be both forward and backward. Furthermore, there is at most one forward path or backward path.

Lemma 3.4 *For any vertex in H there exists at most one forward path or one backward path, but not both.*

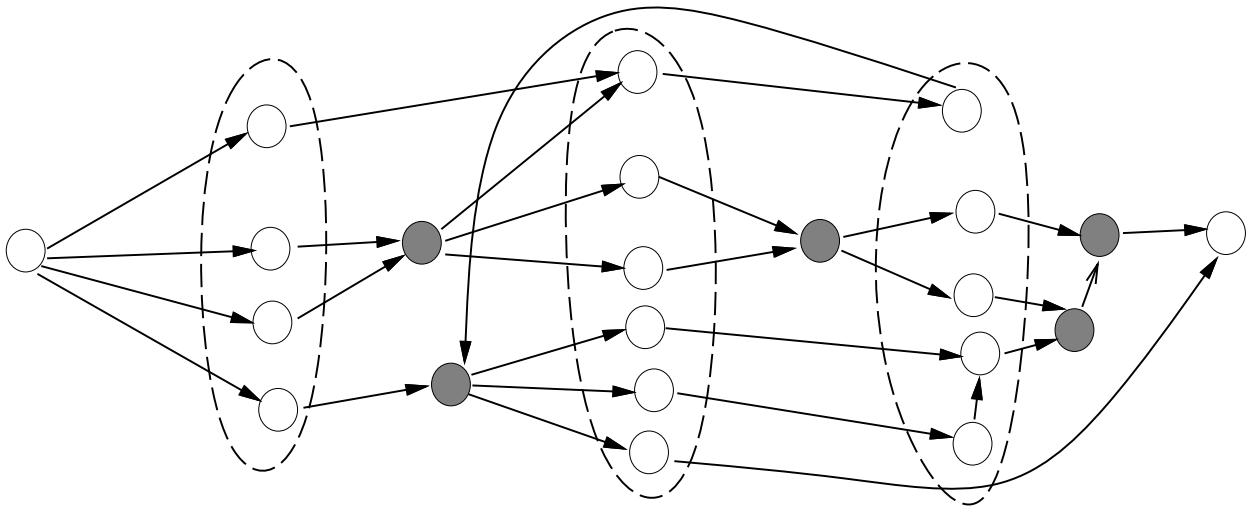

Figure 3: A set of minimum union paths on the starting vertex set {19}, the hitting vertex set {1,2,3,4,5,6,7,8,9,10,11,12,13,21,22} and the terminating vertex set {20}. The hitting set is partitioned into $H_1 = \{1, 2, 3, 4\}$, $H_2 = \{5, 6, 7, 8, 9, 21\}$, and $H_3 = \{10, 11, 12, 13, 22\}$. Vertex 10 is a backward vertex whose backward index is 1. Vertex 22 is also a backward vertex whose backward index is 2. Vertex 9 is a forward vertex.

Proof: If a vertex v is both forward and backward, let P_f and P_b be its forward and backward paths, respectively. Let $e = (x, y)$ be the first edge not in P_b encountered when we traverse P_f starting from v. Since t has no outgoing edge, e must exist. Let G' be the resulting graph by removing the edge e from MUP($D(G), \{s\}, H, \{t\}$). For any $\{s\}$-$\{t\}$ path P in MUP(D(G),{s},H,{t}) that passes through the edge e, there is an $\{s\}$-$\{t\}$ path P' in G' by first traversing P from s to x, then following P_b to the backward successor u of v. Finally, we follow a path from u to t. Thus G' is a set of $\{s\}$-H-$\{t\}$ union paths with a smaller total edge weight or fewer edges than MUP($D(G), \{s\}, H, \{t\}$). Thus we know v cannot be forward and backward.

We now suppose that v has two backward paths P_1 and P_2. Let e be the first edge not in P_2 encountered when we traverse P_1 starting from v. Then by an argument similar to the one given before we can derive a contradiction. Similarly one can prove that it is impossible for v to have more than one forward path. □

An example of a set of minimum union paths is shown in Figure 3.

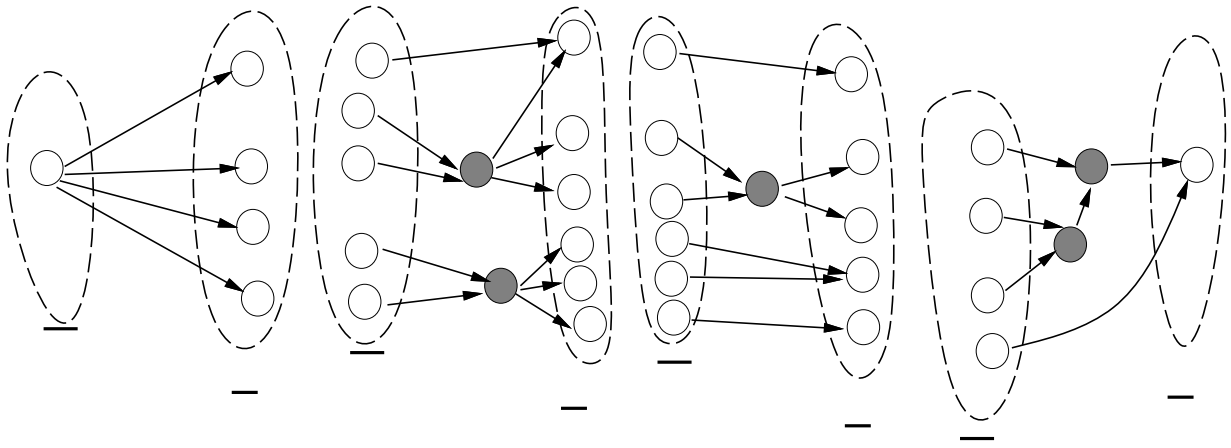

Figure 4: The pairwise extension and the split partition graphs of the minimum union paths shown in Figure 3.

Let H_f (respectively, H_b) be the set of forward (respectively, backward) vertices in H. Let H'_i be the set of vertices in H_i that is neither forward nor backward. We define a *split extension graph* for MUP$(D(G), \{s\}, H, \{t\})$ as follows. For every vertex $u \in (H_f \cup H_b)$, we split u into two vertices u_{in} and u_{out} where u_{in} inherits all incoming edges and u_{out} inherits all outgoing edges. The rest of the edges and the vertices remain unchanged.

Given a backward vertex u whose backward successor is in H_i, its *backward index* $bi(u)$ is $i-1$. The backward index of a forward vertex is ℓ and the backward index of a vertex that is neither forward nor forward is ∞. The *level index* of u, $\ell i(u) = i$ if $u \in H_i$. We partition edges in the split extension graph for MUP$(D(G), \{s\}, H, \{t\})$ into $G_0 \cup G_1 \cup \cdots \cup G_\ell$ where G_i is the induced subgraph of MUP$(D(G), \{s\}, H, \{t\})$ on the two sets of vertices $\overline{H}_i = H'_i \cup \{u_{out} \mid bi(u) = i\}$ and $\underline{H}_i = H'_{i+1} \cup \{u_{in} \mid \ell i(u) = i+1\}$. The sets of vertices $\overline{H}_0, \overline{H}_1, \ldots, \overline{H}_\ell$, $\underline{H}_0, \underline{H}_1, \ldots,$ and $\underline{H}_\ell$ are the *pairwise extension* of $H_0, H_1, \ldots,$ and $H_{\ell+1}$.

In Figure 4, we illustrate an example of the pairwise extension of the set of minimum union paths in Figure 3

Lemma 3.5 *G_i is a minimum Steiner network MSN$(G, \overline{H}_i, \underline{H}_i)$.*

Proof: Note that G_i is a directed Steiner network in G for the set of starting vertices $\overline{H}_i$ and the set of terminating vertices $\underline{H}_i$. If G_i is not a minimum Steiner network, then let G'_i be a minimum Steiner network. We delete edges in G_i from MUP$(D(G), \{s\}, H, \{t\})$ and replace them with edges in G'_i. We obtain a set of union paths with smaller total edge weights or fewer number of edges. This is a contradiction. □

3.2 The Algorithm

Given a MUP($D(G), \{s\}, H, \{t\}$), its *configuration* is the pairwise extension of $H_0, H_1, \ldots,$ and $H_{\ell+1}$. By Lemma 3.5, we can find a MUP($D(G), \{s\}, H, \{t\}$) as follows. We first find its configuration. Then we construct each G_i by using the algorithm in Section 2.2 for finding a minimum Steiner network. Our algorithm finds the correct configuration by enumerating all possible configurations given H.

Lemma 3.6 *There are $O(k! \cdot 2^k \cdot k^k)$ possible configurations, where $k = |H|$.*

Proof: Let $T(k)$ be the number of different ways that one can partition a set of k vertices. $T(0) = 0$ and

$$T(k) = \sum_{i=1}^{k} \left(\begin{array}{c} k \\ i \end{array} \right) \cdot T(k-i).$$

Thus $T(k) = O(k! \cdot 2^k)$. Given a partition for H, each vertex in H can be either a forward vertex, a backward vertex, or a vertex that is neither forward nor backward. The backward index of a backward vertex can also have $\ell - 1$ choices. However, the total number of choices for any vertex equals to the level number for its partition, which is less than or equal to k. Thus there are $O(k^k)$ pairwise extensions for a partition. Hence the lemma holds. □

Theorem 3.7 *MUP($D(G), \{s\}, H, \{t\}$) can be found in $O(k! \cdot (8 \cdot k)^k \cdot k^3 \cdot n^{k-1} + n \cdot m)$ time. where n and m are the numbers of vertices and edges in G, respectively, and $k = |H|$.*

Proof: Note that $|\overline{H}_i| \leq k$ and $|\underline{H}_i| \leq k$ for all $0 \leq i \leq \ell$. Note also that $|\cup_{i=0}^{\ell} (\overline{H}_i \cup \underline{H}_i)| \leq 2k+2$. Given a partition extension of a partition of H, by Theorem 2.10, the time to compute all G_i is bounded by computing a minimum Steiner network from k starting vertices to k terminating vertices, which is $O(4^k \cdot k^3 \cdot n^{k-1} + n \cdot m)$, where $O(n \cdot m)$ is the time to compute the distance network. Given G, the distance network for G needs only to be computed once. Thus by Lemma 3.6, the theorem holds. □

Theorem 3.8 *MUP($D(G), S, H, T$) can be found in $O(k! \cdot (8 \cdot k)^k \cdot k^3 \cdot n^{k-1} + n \cdot m)$ time, where n and m are the numbers of vertices and edges of G, respectively, and $k = |S| + |H| + |T|$.*

Proof: By Lemma 3.2 and Theorem 3.7. □

By Lemma 3.1, the number of Steiner vertices in a minimum Steiner network can be as large as $|S| + |T| + 2 \cdot |H| - 4$. Thus finding a minimum Steiner network by brute force may need to perform a minimum spanning arborescence computation $O(n^{|S|+|T|+2 \cdot |H|-4})$ times. Theorem 3.8 states that by using our algorithm, a set of minimum union paths can be found by performing a minimum spanning arborescence computation $O(k! \cdot (8 \cdot k)^k \cdot k \cdot n^{k-1})$ times

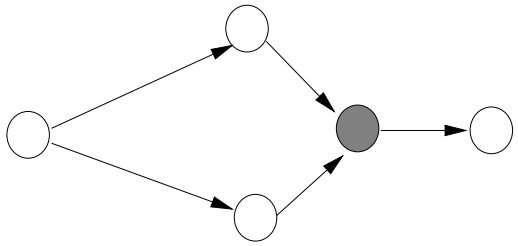

Figure 5: A possible solution for the (1,2,1)-minimum union paths problem in the distance network where the starting vertex is s, the terminating vertex is t, and the set of hitting vertices is $\{h_1, h_2\}$. The shaded vertex is a Steiner vertex.

where $k = |S|+|T|+|H|$. For a fixed $k = |S|+|T|+|H|$ our algorithm runs in $O(n \cdot m+n^{k-1})$ time, while a naive approach takes $O(n \cdot m + n^{k-1+|H|-3})$ time. Thus our algorithm runs asymptotically faster when $|H|$ is a constant greater than 3.

3.3 Algorithm for a Special Case

Let (i, j, k)-minimum union paths problem denote the minimum union paths problem with i starting vertices, j hitting vertices and k terminating vertices. In this section, we give a linear time algorithm to solve the (1,2,1)-minimum union paths problem. Note that the original algorithm needs to compute a distance network and thus takes at least $O(n \cdot m)$ time. We will show that the computation of the distance network can be avoided as in Section 2.3 for the (1,2)-minimum Steiner network problem.

We first analyze all possible configurations for a solution for the (1,2)-minimum Steiner network problem when there is one starting vertex and two terminating vertices. As we shall show below, the solution for the (1,2,1)-minimum union paths problem can be decomposed into the union of solutions to two instances of the (1,2)-minimum Steiner network problem.

In Figure 2, we list the four possible configurations for the (1,2)-minimum Steiner network problem. It is easy to see that the solution for the (1,2,1)-minimum union paths problem can be obtained by the union of solutions to two (1,2)-minimum Steiner network problems. For example, the solution in Figure 5 is a combination of type (1) in Figure 2 and type (2) in Figure 2 (directions of edges reversed). Note that if the first part of the solution is type (3) or (4) in Figure 2, then we only have to find a path from a hitting vertex to the terminating vertex to form a solution we need.

Thus to find the solution for the (1,2,1)-minimum union paths problem with one starting vertex s, two hitting vertices h_1 and h_2, and one terminating vertex t, we first find MSN$(G, \{s\}, \{h_1, h_2\})$. By Theorem 2.11, this can be done in linear time. According to the

discussion in Section 3.1, the set H of hitting vertices can be partitioned into up to 2 levels. When all hitting vertices are partitioned into two levels with exactly one vertex in each level, then there exists neither a backward vertex nor a forward vertex.
We distinguish two cases.

Case 1: The hitting vertices are in the same level.

This corresponds to the case when MSN$(G, \{s\}, \{h_1, h_2\})$ is either type (1) or (2) in Figure 2. Let $r(G)$ be the resulting graph obtained from G by reversing the direction of each edge in G. We find MSN$(r(G), \{t\}, \{h_1, h_2\})$. Since both MSN$(r(G), \{t\}, \{h_1, h_2\})$ and MSN$(G, \{s\}, \{h_1, h_2\})$ are optimal solutions and G is acyclic, no vertex other than h_1 and h_2 can appear in both MSN$(r(G), \{t\}, \{h_1, h_2\})$ and MSN$(G, \{s\}, \{h_1, h_2\})$.

If MSN$(r(G), \{t\}, \{h_1, h_2\})$ is either type (1) or (2) in Figure 2, then the solution we want is MSN$(G, \{s\}, \{h_1, h_2\}) \cup r(MSN(r(G), \{t\}, \{h_1, h_2\}))$. If MSN$(r(G), \{t\}, \{h_1, h_2\})$ is either type (3) or (4) in Figure 2, without loss of generality, let h_1 be the vertex with in-degree 0 in $r($MSN$(r(G), \{t\}, \{h_1, h_2\}))$. A shortest path from s to h_1 together with $r($MSN$(r(G), \{t\}, \{h_1, h_2\}))$ is the solution we want.

Case 2: The two hitting vertices are in different levels.

This corresponds to MSN$(G, \{s\}, \{h_1, h_2\})$ is either type (3) or (4) in Figure 2. Without loss of generality, let h_1 be the hitting vertex whose out-degree is 0 in MSN$(G, \{s\}, \{h_1, h_2\})$. We wrap up the computation by finding a shortest path from h_1 to t.

Hence we have the following theorem.

Theorem 3.9 *The (1,2,1)-minimum union paths problem can be solved in linear time.* □

Remark: When $|H|$ is small, the above enumerative approach by analyzing the partition of H is an effective way to find a solution for the minimum union paths problem.

4 Concluding Remarks

We have described an enumerative approach to two variations of the minimum-cost Steiner problem on a directed acyclic graph with non-negative edge weights. Properties of the solutions were presented and exploited to obtain algorithms more efficient than the naive methods for these two problems. Whether or not additional properties of the graph can be utilized to improve the time complexity of the algorithms for computing optimal solutions remains to be seen.

References

[1] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[2] G. Dahl. Directed Steiner problems with connectivity constraints. *Discrete Applied Math.*, 47:109–128, 1993.

[3] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[4] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[5] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[6] S. L. Hakimi. Steiner's problem in graphs and its applications. *Networks*, 1:113–133, 1971.

[7] F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 22:55–89, 1992.

[8] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem.* Annals of Discrete mathematics 53. North-Holland, 1992.

[9] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart, and Winston, New York, 1976.

[10] S. Martello and P Toth. Finding a minimum equivalent graph of a digraph. *Networks*, 12:89–100, 1982.

[11] L. Nastansky, S. M. Selkow, and N. F. Stewart. Cost-minimal trees in directed acyclic graphs. *Zeitschrift für Operations Research*, pages 59–67, 1974.

[12] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor. The rectilinear Steiner arborescence problem. *Algorithmica*, pages 277–288, 1992.

[13] R. E. Tarjan. *Data Structures and Network Algorithms.* SIAM Press, Philadelphia, PA, 1983.

[14] S. Voss. Worst-case performance of some heuristics for Steiner's problem in directed graphs. *Information Processing Letters*, 48:99–105, 1993.

[15] P. Winter. Steiner problem in networks: A survey. *Networks*, 17:129–167, 1987.

[16] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graphs. *Math. Programming*, 28:271–287, 1984.