Approximation Algorithms for Directed Steiner Problems*

MOSES CHARIKAR† — Stanford University
CHANDRA CHEKURI‡ — Stanford University
TO-YAT CHEUNG§ — City University of Hong Kong
ZUO DAI¶ — City University of Hong Kong
ASHISH GOEL‖ — Stanford University
SUDIPTO GUHA** — Stanford University
MING LI†† — City University of Hong Kong

April 25, 1997

Abstract

The Steiner tree problem which is known to be NP-Complete is the following. Given a weighted undirected graph $G = (V, E)$, and a set $X \subseteq V$ of *terminals*, the objective is to find a tree of minimum cost which connects all the terminals. If the graph is *directed*, in addition to X, we are given a root $r \in V$, and the objective is to find a minimum cost arborescence which connects the root to each of the terminals. In the Generalized Steiner network problem, we are given a set X of pairs of vertices, and the goal is to find a subgraph of minimum cost such that each pair in X is connected. In the undirected case, constant factor algorithms are known for both versions [14, 19, 22, 1, 20], but essentially no approximation algorithms were known for these problems in the directed case, other than the trivial $O(k)$-approximations. We obtain the first non-trivial approximation algorithms for both problems in general *directed* graphs. For the Directed Steiner tree problem, we design a family of algorithms that achieve an approximation ratio of $i(i-1)k^{1/i}$ in time $O(n^i k^{2i})$ for any fixed $i > 1$, where k is the number of terminals. For $i = \log k$, we obtain an $O(\log^2 k)$ approximation ratio in quasi-polynomial time. For the Directed Generalized Steiner network problem, we give an algorithm that achieves an approximation ratio of $O(k^{2/3} \log^{1/3} k)$, where k is the number of pairs of vertices that are to be connected. Related problems including the Group Steiner tree problem, the Node Weighted Steiner tree problem and several others can be reduced in an approximation preserving fashion to the problems we solve, giving the first non-trivial approximations to those as well.

*This paper reports the combined version of the two papers [6] and [7] the results of which were obtained independently by the respective authors.

†Supported by an ARO MURI Grant DAAH04-96-1-0007 and NSF Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. Email: moses@cs.standford.edu.

‡Supported by an ARO MURI Grant DAAH04-96-1-0007 and NSF Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. Email: chekuri@cs.stanford.edu.

§Supported by City University of Hong Kong. Email: cscheung@cityu.edu.hk.

¶Supported by City University of Hong Kong. Email: 00410822@cityu.edu.hk.

‖Supported by ARO Grant DAAH04-95-1-0121 and NSF Grant CCR9304971. Email: agoel@cs.standford.edu.

**Supported by an ARO MURI Grant DAAH04-96-1-0007 and NSF Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. Email: sudipto@cs.stanford.edu.

††Supported in part by the NSERC Operating Grant OGP0046506, ITRC and a CGAT grant. On research leave from University of Waterloo. Email: mli@cs.cityu.edu.hk.

1 Introduction

The Steiner tree problem is defined as follows: given a graph $G = (V, E)$ with a cost function c on the edges, and a subset of vertices $X \subseteq V$ (called *terminals*), the goal is to find a minimum cost tree that includes all the vertices in X. The cost of the tree is defined as the sum of the costs of the edges in the tree. Note that the tree may include vertices not in X as well (These are known as steiner vertices).

The Steiner tree problem is NP-Complete even when the graph is induced by points in the plane [9], and is MAXSNP-hard [4] for general graphs. Polynomial time approximation algorithms for this problem were first developed by Kou, Markowsky and Berman [14] and by Takahashi and Matsuyama [19]. These algorithms achieved an approximation factor of $2 - \frac{1}{k}$ where k is the number of terminals. Zelikovsky [22] obtained a ratio of $\frac{11}{6}$ and this has been further improved by Berman and Ramaiyer [3] and Karpinski and Zelikovsky [16]. The best ratio achievable currently is 1.644. In a recent result, Arora [2] has given a polynomial time approximation scheme for the Steiner tree problem on graphs induced by points in the plane.

The directed version of the Steiner tree problem is a natural extension of the undirected version and is defined as follows. Given a directed graph $G = (V, A)$, a specified root $r \in V$, and a set of terminals $X \subseteq V(|X| = k)$, the objective is to find the minimum cost arborescence rooted at r and spanning all the vertices in X (In other words r should have a path to every vertex in X). The Directed Steiner tree problem is clearly the right model for finding cheap multicast trees in asymmetric networks [17, 18] – a problem that has attracted considerable attention recently. Multicasting in networks is the scenario where a single server wants to distribute the same data to several nodes in the network. The Directed Steiner tree problem has several other applications in network design which we elaborate on at the end of the paper. For a general survey on the use of Steiner tree problems in networks, see [21]. From a theoretical point of view, the Directed Steiner tree problem is very useful since a wide variety of problems involving connectivity and covering in both directed and *undirected* graphs can be reduced to this problem in approximation preserving ways. These include the Group Steiner tree problem, node weighted Steiner tree problem [12, 15], several interesting problems in connected domination, namely Edge Weighted Connected Dominating sets, Group (or Set) TSP, Node weighted Steiner Connected domination, and others. For some of the reductions regarding connected domination see [11].

We also consider the Directed Generalized Steiner network problem where instead of a root and a set of terminals, we are given a set of k pairs of vertices, and the objective is to find a minimum cost subgraph which connects each pair. In the undirected case, it is easy to see that the optimal solution is a forest but that is not necessarily true in the directed case. The general version has several applications in network design and network reliability though it is only recently that progress has been made in terms of obtaining good approximation algorithms even in the undirected case. The first constant factor algorithms for the undirected case were given in [1, 20]. Agrawal et al., also give $O(\log R)$ approximations for the more general case where each pair has a connectivity requirement r_{ij} (the number of edge disjoint paths required for the pair (i, j)) and R is the maximum connectivity requirement. In this paper we restrict ourselves to case where $r_{ij} = 1$ for all pairs.

A fairly easy reduction from the Set cover problem shows that it is hard to approximate Directed Steiner tree to a factor better than $\ln k$ where k is the number of terminals unless $NP \subseteq DTIME[n^{O(\log \log n)}]$ [8]. It is also easy to obtain an approximation factor of k, by connecting every terminal to the root via a shortest path. The only known polynomial time approximation algorithm even for a special case is due to Zelikovsky [23], where he gives an approximation algorithm which achieves a ratio of $(2 + \ln k)^{i-1} k^{1/i}$ for any $i > 0$ for directed *acyclic* graphs. No algorithms were known for the case of general directed graphs. Zelikovsky further conjectures that no subpolynomial approximation guarantees are possible unless $P = NP$. In this paper, we present a factor $i(i-1)k^{1/i}$ approximation algorithm for any $i > 0$ for the Directed Steiner tree problem. Not only is our approach new and simpler compared to that of Zelikovsky [23], but also generalizes to arbitrary directed graphs. Further, setting $i = \log k$, we obtain an $O(\log^2 k)$ approximation in $O(n^{3 \log k})$ time. This fact gives strong evidence that the problem can be approximated to within poly-logarithmic factors in polynomial time contrary to Zelikovsky's conjecture. Many of the problems we mentioned earlier which reduce to the steiner tree problem are also hard to approximate to within a logarithmic factor. Thus our results have implications for the complexity of those problems as well. For the Directed Generalized Steiner tree problem we present an algorithm which achieves an approximation ratio of $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$.

The rest of the paper is organized as follows. In Section 2 we set up the notation and prove some basic lemmas. Sections 3 and 4 describe the algorithms for the Directed Steiner tree and the generalized versions respectively. We briefly sketch the reductions of some of the other problems in Section 5.

2 Preliminaries

In this section we will give some basic definitions and lemmas concerning partial solutions and l-level trees. We define a slightly more general version of the Directed Steiner tree problem below. We assume without loss of generality that between every pair of vertices u and v, there exists an edge $< u, v >$ of cost equal to the shortest path distance from u to v in G.

Definition 1 *Given root $r \in V(G)$, an integer k and a set $X \subseteq V$ of terminals with $|X| \geq k$, the problem* D-STEINER(k, r, X) *is to construct a tree rooted at r, spanning any k terminals in X and of minimum possible cost.*

We give an $i(i-1)k^{1/i}$ approximation to the D-STEINER(k, r, X) problem for any $i > 1$. This then implies the same approximation ratio for the Directed Steiner tree problem. Our approach gives similar approximation ratios for connectivity problems where it is sufficient to connect any k of a given set of terminals. For example, we can solve the generalization of the k-MST problem [5, 10] to directed graphs, where the objective is to find an arborescence on k vertices of G of minimum cost.

Let $c(e)$ denote the *cost* of edge e and let $c(T)$ denote the *cost* of a tree T, or the sum of the costs of the edges in T. Let $k(T)$ denote the number of terminals in T; in other words $k(T) = T \cap X$.

Definition 2 *Define $d(T)$, the* density *of tree T to be the ratio of the cost of the tree to the number of terminals in T; in other words $d(T) = c(T)/k(T)$.*

We will assume without loss of generality that all terminals are at the leaves of any Steiner tree. We can arrange this by connecting a dummy terminal to the real terminal with a zero cost edge.

Definition 3 *Define an $f(k)$-*partial approximation procedure *for* D-STEINER(k, r, X) *to be a procedure which constructs a tree T' rooted at r, spanning $1 \leq k' \leq k$ terminals in X such that $d(T') \leq f(k) \cdot \frac{C_{OPT}}{k}$, where C_{OPT} is the cost of an optimal solution to* D-STEINER(k, r, X).

If $\mathcal{A}(k, r, X)$ is a partial approximation procedure for D-STEINER(k, r, X), we can apply $\mathcal{A}$ repeatedly to obtain an approximation algorithm $\mathcal{B}(k, r, X)$ for D-STEINER(k, r, X) as follows. $\mathcal{B}(k, r, X)$ first calls $\mathcal{A}(k, r, X)$. Suppose this returns a tree T_1 spanning k_1 terminals. If $k_1 = k$, return T_1. Else, let X_1 be the set of terminals spanned by T_1. Call $\mathcal{B}(k-k_1, r, X-X_1)$ and let T_2 be the tree returned by this procedure. Return $T_1 \cup T_2$.

Lemma 1 *Given $\mathcal{A}(k,r,X)$, an $f(k)$-partial approximation for* D-STEINER(k,r,X) *where $f(x)/x$ is a decreasing function of x, the algorithm $\mathcal{B}(k,r,X)$ is a $g(k)$-approximation algorithm for* D-STEINER(k,r,X) *where $g(k)=\int_0^k \frac{f(x)}{x}dx$.*

Proof: We will prove the claim by induction on k. The base case $k=1$ follows as $f(1) \leq \int_0^1 \frac{f(x)}{x}dx$ (by the decreasing property of $\frac{f(x)}{x}$). Suppose it is true for all values less than k. We will prove the claim for k. Let T_{OPT} be an optimal solution to D-STEINER(k,r,X). Suppose the call to $\mathcal{A}(k,r,X)$ returns tree T_1 rooted at r spanning k_1 terminals, i.e. $k(T_1)=k_1$.

$$d(T_1) = \frac{c(T_1)}{k_1} \leq f(k)\frac{c(T_{OPT})}{k} \tag{1}$$

$$c(T_1) \leq k_1 \cdot \frac{f(k)}{k} \cdot c(T_{OPT}) \leq \left(\int_{k-k_1}^{k} \frac{f(x)}{x}dx\right) c(T_{OPT}) \tag{2}$$

where the last inequality follows from the decreasing property of $\frac{f(x)}{x}$. If $k_1 = k$, the algorithm returns T_1. For this case, $c(T_1) \leq g(k) \cdot c(T_{OPT})$ proving that the algorithm gives a $g(k)$-approximation.

Suppose $k_1 < k$. Let X_1 be the set of terminals spanned by T_1. Let T_2 be the tree returned by the recursive call to $\mathcal{B}(k-k_1, r, X-X_1)$. Since T_{OPT} spans k terminals in T, it spans at least $k-k_1$ terminals in $X-X_1$. Hence the minimum cost tree on $k-k_1$ terminals in $X-X_1$ has cost at most $c(T_{OPT})$. By the inductive hypothesis, $c(T_2) \leq g(k-k_1) \cdot c(T_{OPT})$, i.e.

$$c(T_2) \leq \left(\int_0^{k-k_1} \frac{f(x)}{x}dx\right) c(T_{OPT}) \tag{3}$$

Adding (2) and (3), we get

$$c(T_1) + c(T_2) \leq g(k)c(T_{OPT})$$

This proves that for this case too, the algorithm gives a $g(k)$-approximation. ■

Definition 4 *An l-level tree is a tree where no leaf is more than l edges away from the root.*

Lemma 2 (Zelikovsky [23]) *For all $l \geq 1$ there exists an l-level tree that provides a $k^{\frac{1}{l}}$ approximation to* D-STEINER(k,r,X).

We use $T_{OPT}^{(l)}$ to refer to the optimum l-level tree that solves D-STEINER(k,r,X) and $C_{OPT}^{(l)}$ to refer to its cost.

3 Directed Steiner Tree

Before we describe our algorithm formally, we will provide some intuition and an outline of our techniques. The density of a tree can be interpreted as the average cost of connecting a terminal to the root. Lemma 1 shows that a partial approximation procedure, which finds a tree with density close to that of optimal, leads to a good approximation algorithm. Our algorithm to find trees of good density is motivated by the following.

Input: (k, r, X); Output: A tree T rooted at r that satisfies at least k terminals in X.

if there do not exist k terminals in X reachable from r, **then return** ϕ.

$T \leftarrow \phi$;

while k>0

  $T_{BEST} \leftarrow \phi$;

  for each vertex $v \in V$, and each $k', 1 \leq k' \leq k$

    $T' \leftarrow A_{i-1}(k', v, X) \cup \{< r, v >\}$;

    if $d(T_{BEST}) > d(T')$ **then** $T_{BEST} \leftarrow T'$

  $T \leftarrow T \cup T_{BEST}$; $k \leftarrow k - |X \cap V(T_{BEST})|$; $X \leftarrow X - V(T_{BEST})$

return T

Figure 1: Algorithm $A_i(k, r, X)$

A trivial algorithm for the problem is to compute shortest paths from each of the terminals to the root and combine them. It is instructive to consider an example for which this algorithm gives a ratio of k. Consider a graph where there is path of cost C_{OPT} from the root to a vertex v, and zero cost edges from v to each of the terminals. In addition, there are paths of cost $C_{OPT} - \epsilon$ from the root to each of the terminals. The naive algorithm picks each of the shortest paths and does not use the path through v, thus incurring a cost $k(C_{OPT} - \epsilon)$.

Motivated by the above extreme example, we can think of finding subtrees of good density which have the following structure. The tree consists of a path from the root r to an intermediate node v, and v is connected to some set of terminals using a shortest path to each of them. For obvious reasons, we call such a structure a *bunch*. The advantage of choosing such a simple structure is that we can compute, in polynomial time, a bunch with the best density. It is not difficult to see that a 2-level tree can be decomposed into disjoint bunches. Therefore, we can use Lemma 2 to claim that the best bunch has density no more than $\sqrt{k}$ times the density of the optimal tree. This immediately gives us a simple $\sqrt{k}$ approximation.

An obvious way to improve the result is to find structures which are more general than bunches and approximate the optimal more closely. We obtain a much improved ratio using this approach but the simplest way to describe it is via recursion, which we proceed to do next. We will define a sequence of algorithms $A_i(k, r, X)$ such that A_i gives a $i(i-1)k^{1/i}$-approximation for D-Steiner(k, r, X). Given the fact that the running time of A_i is $O(n^{O(i)})$, we can set $i = \log k$ to obtain an $O(\log^2 k)$ algorithm in quasi-polynomial time.

Let $T_i(k, r, X)$ refer to the tree returned by $A_i(k, r, X)$. Let $C^{(i)}_{OPT}(k, r, X)$ denote the cost of the optimum i-level tree that solves D-Steiner(k, r, X), and let $d^{(i)}_{OPT}(k, r, X) = C^{(i)}_{OPT}(k, r, X)/k$ be its density. Let $< u, v >$ refer to the edge from u to v in the graph G.

$A_1(k, r, X)$ is simple to describe – it finds the k terminals which are closest to the root and connects them to the root using shortest paths. $A_i(k, r, X)$ repeatedly finds a vertex v and a number $k', 1 \leq k' \leq k$ such that the density of the tree $T_{i-1}(k', v, X) \cup \{< r, v >\}$ is the least among all trees of this form. The algorithm $A_i(k, r, X)$ is described formally in Figure 1.

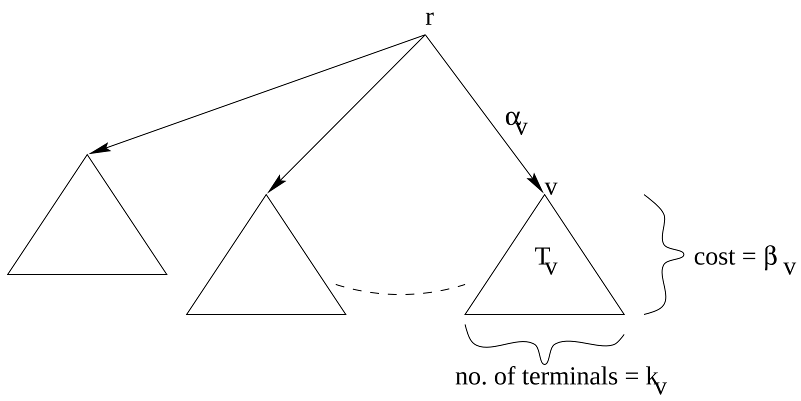

Figure 2: The optimum i level tree

We now prove the following lemma [1] for arbitrary i. Recall that we are working with the transitive closure of the original graph.

Lemma 3 *The trees T_{BEST} chosen by the algorithm $A_i, i \geq 2$ in step 7 have the following property: $d(T_{BEST}) \leq (i-1) \cdot d^{(i)}_{OPT}(k,r,X)$, where k and X refer to the current values being used by the algorithm A_i.*

Proof: The proof is by induction. Since $A_2(k,r,X)$ actually finds the best density tree, the lemma holds for $i = 2$. We now assume that the lemma holds for all j less than i.

Consider an outgoing edge $< r, v >$ in the tree $T^{(i)}_{OPT}(k,r,X)$. Let α_v denote the cost of the edge $< r, v >$, and let T_v represent the subtree of $T^{(i)}_{OPT}(k,r,X)$ rooted at v. Notice that T_v must be an $i-1$ level tree. Let β_v be the cost of T_v, and $k_v < k$ be the number of terminals satisfied by T_v (see Figure 2). We choose v to minimize the quantity $d_v = (\alpha_v + \beta_v)/k_v$. It is easy to see that d_v can be no more than $d^{(i)}_{OPT}(k,r,X)$.

Now consider the algorithm $A_{i-1}(k_v, v, X)$. We look at the *last* stage during the execution of $A_{i-1}(k_v, v, X)$ when the number of terminals picked up is less than $k_v/(i-1)$. At this stage, there are at least $k_v - k_v/(i-1) = k_v(i-2)/(i-1)$ more terminals that need to be picked up by $A_{i-1}(k_v, v, X)$. The density of the optimal $i-1$ level tree that satisfies these terminals can be no more than $(\beta_v/k_v) \cdot (i-1)/(i-2)$. Let T' refer to the tree found by $A_{i-1}(k_v, v, X)$ just after this stage when the number of terminals exceeds $k_v/(i-1)$ for the first time. It follows from our inductive hypothesis that the density of T' can be no worse than $(\beta_v/k_v) \cdot (i-1)$. Let k' be the number of terminals in T'.

Notice that the algorithm $A_{i-1}(k', v, X)$ would produce exactly the tree T' described above. When $A_i(k,r,X)$ invokes $A_{i-1}(k', v, X)$, it would get back a tree satisfying $k' \geq k_v/(i-1)$ terminals. Taking into account the connection cost from r to v, the density of this tree would be at most $\alpha_v/k' + (\beta_v/k_v) \cdot (i-1) \leq (i-1)d_v$. Recall that $d_v \leq d^{(i)}_{OPT}(k,r,X)$.

This completes the proof of the inductive hypothesis. ■

Theorem 1 *For $i > 1$, $A_i(k,r,X)$ provides an $i(i-1)k^{1/i}$ approximation to* D-Steiner(k,r,X) *in time $O(n^i k^{2i})$.*

[1] Lemma 3 is similar to Lemma 3.5 in the paper by Korsartz and Peleg [13]. However their proof has a bug. We also believe that a lemma as they state it cannot be true. Our lemma which is simpler and can be used to prove the results in [13] as well.

Proof: We divide the execution of $A_i(k,r,X)$ into stages. Each stage corresponds to one iteration of the outer loop (line 2, Figure 1). Let k_j be the number of terminals that still need to be satisfied at the beginning of stage j, and let X_j be the set of unsatisfied terminals. Using Lemma 3, we can claim that the density d_j of A_i during stage j is no worse than $(i-1)\cdot C^{(i)}_{OPT}(k_j,r,X_j)/k_j$. We can now invoke Lemma 2 to claim that $C^{(i)}_{OPT}(k_j,r,X_j) \leq k_j^{1/i} C_{OPT}$. Therefore, $d_j \leq (i-1)\cdot C_{OPT}\cdot k_j^{1/i}$. Notice that each stage behaves like an $(i-1)\cdot k_j^{1/i}$-partial approximation to D-Steiner(k_j,r,X_j). Using Lemma 1 we obtain the following bound on the cost of the solution produced by $A_i(k,r,X)$.

$$\begin{aligned} cost(T_i(k,r,X)) &\leq (i-1)\cdot C_{OPT}\int_0^k \frac{y^{1/i}dy}{y} \\ &= i(i-1)k^{1/i}\cdot C_{OPT} \end{aligned}$$

We observe that A_i invokes A_{i-1} at most nk^2 times and the running time bound follows. ■

Theorem 2 *$A_{\lceil \log k \rceil}(k,r,X)$ provides an $O(\log^2 k)$ approximation to the Directed Steiner Tree problem in time $O(n^{3\log k})$.*

Proof: Follows from Theorem 1. ■

Zelikovsky conjectures in [23] that Directed Steiner tree problem cannot have a subpolynomial approximation guarantee unless $P = NP$. Contrary to his conjecture, Theorem 2 gives strong evidence that polylogarithmic approximation in polynomial time is possible.

For many applications of multicasting there is a restriction on the number of hops (the number of edges used in the path between the root and the terminals). This problem has been studied by Korsartz and Peleg in [13] where they call such trees "shallow-light trees". We can extend our techniques to obtain the following theorem.

Theorem 3 *For any fixed $i > 0$, there is an algorithm which runs in time $O((ink^2)^i)$ and gives an approximation ratio of $O(i\log k)$ for the best steiner tree with path lengths bounded by i. Furthermore for arbitrary i, there is an algorithm which achieves an approximation ratio of $O(\log^2 k)$ and runs in time $n^{O(\log k)}$.*

This is an improvement over the $O(\log^3 k)$ approximation given in [13].

4 Directed Generalized Steiner Network Problem

Formally the problem is the following. Given a directed graph $G(V,E)$ and a set $X = \{< u_i, v_i >\}$ of k node pairs, find the minimum cost subgraph H of G such that for each node pair $< u_i, v_i > \in X$, there exists a directed path from u_i to v_i in H. The cost of the subgraph H is the sum of the cost of all the edges in H. This problem has been well studied in undirected graphs and constant factor approximations are presented in [1, 20].

As before, we work with a slightly more general problem we call DG-Steiner (k,X) which is the problem of finding a minimum cost subgraph H of G that satisfies at least k node pairs from the set X. Clearly, Directed Generalized Steiner tree is a special case of DG-Steiner (k,X). In this section, we present an $O(k^{\frac{2}{3}}\log^{\frac{1}{3}} k)$ approximation to the DG-Steiner (k,X) problem.

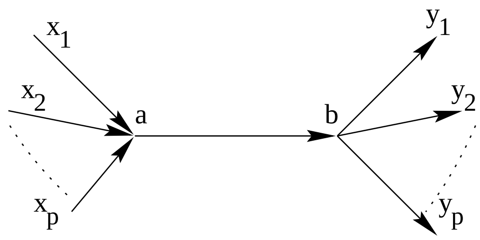

Figure 3: The bunch $< a, b, Y >$

1 $d \leftarrow \infty$; $B \leftarrow \phi$

2 **for** `all pairs` $< a, b > \in V(G) \times V(G)$

  3 **for** `each pair` $< u, v > \in X$, $s[u, v] \leftarrow c(< u, a >) + c(< b, v >)$

  4 **sort** X `in increasing order of` s. `Let` $< u_j, v_j >$ `refer to the` j`th pair in this sorted list.`

  5 **for** p `going from 1 to` k

    6 $C \leftarrow c(< a, b >) + s[u_1, v_1] + s[u_2, v_2] + \ldots + s[u_p, v_p]$

    7 **if** $C/p \leq d$ **then** $(d \leftarrow C/p$; $B \leftarrow < a, b, \{< u_1, v_1 >, \ldots, < u, v_p >\})$

8 **return** B

Figure 4: Algorithm $\mathcal{C}(k, X)$

Let $C_{OPT}(k, X)$ denote the cost of the optimal solution to DG-Steiner (k, X). Define $k(H)$ to be the minimum of k and the number of node pairs in X that get satisfied by H. Further, let $c(H)$ denote the cost of H, and $d(H) = c(H)/k(H)$ be the density of H. We will omit the parameters (k, X) where their values are clear from the context. An $f(k)$ partial approximation to DG-Steiner (k, X) can be defined in a manner similar to that for D-Steiner(k, X).

We assume without loss of generality that between every pair of vertices $< u, v >$, there exists an edge of cost equal to the shortest path distance from u to v in G. The main idea of the algorithm is to repeatedly find good bunches, where a bunch is defined as follows.

Definition 5 *Let* $Y = \{< u_1, v_1 >, \ldots, < u_p, v_p >\}$ *be a subset of* X *containing* $p \leq k$ *node pairs. Also let* a *and* b *be vertices of* G. *A* bunch $B =< a, b, Y >$ *is a graph with the vertex set* $\{a, b, x_1, x_2, \ldots x_p, y_1, y_2, \ldots y_p\}$. B *has an edge* $< a, b >$ *of cost identical to that of the edge* $< a, b >$ *in the graph* G. B *also has edges* $< x_i, a >$ *and* $< b, y_i >$, *with costs identical to those of* $< u_i, a >$ *and* $< b, v_i >$ *in* G.

Figure 3 illustrates the structure of a bunch. Notice that the x_i s and y_i s are unique even though the u_i s and v_i s may not be. In Figure 4 we give an algorithm $\mathcal{C}(k, X)$ that finds a bunch of minimum density. We then prove the existence of a bunch $B(k, X)$ that has density no more than $O(d(H_{OPT}) \cdot k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$. This allows us to claim that $\mathcal{C}$ provides an $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$ partial-approximation to DG-Steiner (k, x). We then apply Lemma 1 to obtain an approximation to DG-Steiner (k, x).

Finding the best bunch is straightforward – we find the lowest cost bunch for all possible values of a, b, and p and then choose the minimum density bunch out of these (at most n^2k) lowest cost bunches. The proof of the following lemma is quite simple and hence we omit it.

Lemma 4 *The algorithm $\mathcal{C}(k, X)$ finds a minimum density bunch.*

We now prove the existence of low-density bunches. For convenience, we define $h(k), k > 1$ to be the quantity $6k^{\frac{2}{3}} \log^{\frac{1}{3}} k$. Also, let $h(1) = 1$.

Theorem 4 *There exists a bunch $B(k, X)$ such that $d(B) \leq (C_{OPT}/k) \cdot h(k)$.*

Proof: Suppose there exists a node pair $< u, v > \in X$ such that the distance between u and v is no more than $(C_{OPT}/k) \cdot h(k)$. Then $< u, v, \{< u, v >\} >$ is the required bunch, and we are done. Notice that if $k = 1$ then such a node pair always exists. Therefore we will implicitly assume $k \geq 2$ for the rest of the proof.

If there exists no such node pair, then in the optimal solution $H^*(k, X)$ each node pair must be separated by a distance of at least $(C_{OPT}/k) \cdot h(k)$. Also, no node pair can be separated by more than C_{OPT}. For $i \geq 1$, let X_i denote the set of node pairs $< u, v >$ such that the distance from u to v in $H^*(k, X)$ lies in the range $\left[2^{i-1} \cdot h(k)\frac{C_{OPT}}{k}, 2^i \cdot h(k)\frac{C_{OPT}}{k}\right]$. Let I_{MAX} denote the maximum value of i for which X_i is non empty. Since $k \geq 2$, $I_{MAX} \leq \log k$. Therefore there exists an i' such that $|X_{i'}| \geq \frac{k}{\log k}$. Let $H^*_{i'}$ be a minimum cost subgraph of H^* satisfying all node pairs in $X_{i'}$. For each node pair $< u, v > \in X_{i'}$, let $P_{H^*_{i'}}(u, v)$ be the shortest path between u and v in $H^*_{i'}$. If there are multiple shortest paths between u and v, choose one of them arbitrarily.

The sum of the shortest paths between all the node pairs in $X_{i'}$ is greater than $2^{i'-1}h(k) \cdot \frac{C_{OPT}}{k} \cdot (\frac{k}{logk})$. But the cost of $H^*_{i'}$ is no more than C_{OPT}. Thus there has to be an edge which is shared by at least $\frac{h(k)}{\log k} \cdot 2^{i'-1}$ paths. Let this edge be $e =< u, v >$, and let X_e be the set of node pairs $< a, b >$ such that $P_{H^*_{i'}}(a, b)$ passes through e. Each of these paths can be split into three parts, $P(a, u)$, the edge $< u, v >$ and $P(v, b)$. Let H_1 be the union of paths of the form $P(a, u)$ (ie. the first components) and H_2 be the union of paths of the form $P(v, b)$ (ie. the third components). Clearly $c(H_1) \leq C_{OPT}$ and $c(H_2) \leq C_{OPT}$. Let T_1 be a shortest-incoming-path tree (rooted at u) in the graph H_1 and T_2 be a shortest-outgoing-path tree (rooted at v) in the graph H_2.

Let $dist_{T_1}(a, u)$ be the cost of the path from a to u in T_1. $dist_{T_2}(v, b)$ is defined similarly. Let $D = MAX_{<a,b>\in X_e}(dist_{T_1}(a, u) + dist_{T_2}(v, b))$. By definition of the sets X_i, D is no more than $(C_{OPT}/k)h(k) \cdot 2^{i'}$. Let c represent the quantity $(\frac{k}{\log k})^{\frac{1}{3}}$. We now divide the trees T_1 and T_2 into at most c segments, each of depth C_{OPT}/c. Node a belongs to segment i of T_1 if $dist_{T_1}(a, u) \in \left[(i-1) \cdot \frac{C_{OPT}}{c}, i \cdot \frac{C_{OPT}}{c}\right]$. Similarly, node b belongs to segment j of T_2 if $dist_{T_2}(v, b) \in \left[(j-1) \cdot \frac{C_{OPT}}{c}, j \cdot \frac{C_{OPT}}{c}\right]$.

Definition 6 *We define k_{ij} ($1 \leq i \leq c, 1 \leq j \leq c$) to be the number of node pairs $< a, b > \in X_e$ such that a belongs to segment i of T_1 and b belongs to segment j of T_2. Also, let $n_i^{(1)}$ ($i \geq 2$) be the number of nodes a' in segment $i-1$ of T_1 which satisfy the following properties:*

1. *There exists no vertex a in segment $i-1$ of T_1 such that a lies on the path from a' to u.*
2. *There exists a vertex a in segment i of T_1 such that a' lies on the path from a to u.*

Define $n_1^{(1)}$ to be 1. The quantities $n_i^{(2)}$ are similarly defined on the tree T_2. Let n_{ij} be the product of $n_i^{(1)}$ and $n_j^{(2)}$.

Informally, $n_i^{(1)}$ represents the number of branches of the tree T_1 that cross the entire $(i-1)$-th segment of T_1.

Lemma 5 *There exist* $1 \leq i \leq c$ *and* $1 \leq j \leq c$ *such that* $k_{ij}/n_{ij} \geq (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$.

Proof of Lemma 5: Each node pair that belongs to X_e contributes to k_{ij} for some pair (i,j). Therefore $\sum_{i,j} k_{ij} \geq (h(k)/\log k) \cdot 2^{i'-1}$. Now notice that $\sum_i n_i^{(1)}$ $(\sum_i n_i^{(1)})$ is bounded by c, since the number of branches that cross an entire segment of depth C_{OPT}/c can be no more than c (remember that the cost of T_1 can be no more than C_{OPT}). Now, $\sum_{i,j} n_{ij} \leq (\sum_i n_i^{(1)}) \cdot (\sum_j n_j^{(2)}) \leq c^2$. Therefore, we are guaranteed that there exists a pair $(i,j), 1 \leq i,j \leq c$ which satisfies the required condition. ■

Using Lemma 5, we can claim the existence of nodes $a' \in T_1$ and $b' \in T_2$ and a set $X' \subseteq X_e$, $|X'| \geq (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$ such that the following properties hold:

1. a' lies on the path from a to u in T_1 and b' lies on the path from v to b in T_2.

2. $dist_{T_1}(a,a') \leq 2C_{OPT}/c$ and $dist_{T_2}(b',b) \leq 2C_{OPT}/c$.

Let the bunch B' be defined as $< a', b', X' >$. Let $k' = k(B')$ be the number of node pairs satisfied by this bunch ($k' \geq (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$). Also, let $c(B')$ be the cost of this bunch. The shortest path from any first component of X' to a' is at most $2C_{OPT}/c$, by construction. The shortest path from a' to b' is at most D. The shortest path from b' to any second component of X' is also at most $2C_{OPT}/c$. Therefore, $c(B') \leq D + 4k' \cdot C_{OPT}/c$. Recall that $D \leq (C_{OPT}/k)h(k) \cdot 2^{i'}$, $c = (\frac{k}{\log k})^{\frac{1}{3}}$, and $h(k) = 6k^{\frac{2}{3}}(\log k)^{\frac{1}{3}}$. Substituting these values, we obtain $d(B') \leq \frac{C_{OPT}}{k} \cdot h(k)$. ■

Theorem 5 $\mathcal{C}(k,X)$ *can be used to obtain a polynomial time algorithm which gives an approximation ratio of* $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$ *for the Directed Generalized Steiner network problem.*

Proof: From Theorem 4 and Lemma 4 it follows that $\mathcal{C}(k,X)$ produces an $h(k)$ partial approximation to DG-Steiner (k,x). We can now invoke Lemma 1 to obtain the desired result. ■

Remark 1 *Since k could be $\Theta(n^2)$, the approximation ratio we guarantee is $\omega(n)$. We remark here that we can use our result on directed steiner trees to obtain a $O(\min\{k^{2/3}\log^{\frac{1}{3}} k, n^{1+\epsilon}\})$ algorithm for the generalized version. Obtaining a $O(n)$ approximation for all k seems non-trivial.*

5 Related Problems

In this section we show how the results in Section 3 can be used to obtain $O(k^\epsilon)$-approximations for several related problems which are at least as hard as set cover.

The Group Steiner Tree Problem: The Group Steiner Tree Problem is the following: Given an undirected graph $G(V,E)$, a root r, and a collection of groups $X = \{g_1, g_2, \ldots, g_k\}, g_i \subseteq V$, find the minimum cost tree T that connects at least one vertex in each of the groups g_i to the root. There is also a directed version of this problem. The undirected (as well as directed) Group Steiner Tree Problem reduces to the Directed Steiner Tree Problem as follows: For each group g_i, introduce a new dummy vertex x_i and connect x_i using zero cost directed edges to each of the vertices in g_i. These dummy vertices are the terminals in a directed Steiner tree instance with the same root.

Some other related problems like Set TSP can also be reduced similarly.

Node Weighted Variants: There is a simple approximation preserving reduction between the node weighted and the edge weighted cases when the graph is directed. Therefore all our results carry over to the node weighted cases as well.

The Strongly Connected Steiner Subgraph problem: Given a digraph $G(V, E)$ and $V_s \subseteq V$ where $\mid V_s \mid = k$, find a minimum cost subgraph S such that the vertices in V_s are strongly connected in S. It can be approximated to a factor of $2i(i-1)k^{1/i}$ by a simple reduction to directed Steiner tree.

6 Conclusions

Our work leaves several interesting open problems. The most important of them is to obtain a polynomial time algorithm with a poly-logarithmic approximation ratio for the directed steiner tree problem. Our results give strong evidence of the existence of such an algorithm. For the generalized version, the worst ratio we can prove for our algorithm is $\Omega(\sqrt{k})$ but the upper bound we can guarantee is only $O(k^{2/3} \log^{\frac{1}{3}} k)$. We would like to see the gap bridged. It is surprising that obtaining even a $O(n)$ algorithm seems non-trivial while the best hardness result is the same as that for set cover. Improving the approximation guarantee or the hardness for this problem would be very interesting. There is also the general case where each pair specifies a connectivity requirement r_{ij} which gives the number of edge disjoint paths required between i and j for which no non-trivial upper bound is known.

Acknowledgements

We thank Samir Khuller for being a Steiner node in connecting the authors.

References

[1] A. Agrawal, P. Klein and R. Ravi. "When trees collide: An approximation algorithm for generalized Steiner tree problem on networks" *23rd ACM Symposium on Theory of Computing*, 134-144,(1991)

[2] S. Arora. "Polynomial-time approximation schemes for Euclidean TSP and other geometric problems". *Proceedings of 37th IEEE Symp. on Foundations of Computer Science*, pp. 2-12 (1996).

[3] P. Berman and V. Ramaiyer, "Improved approximation algorithms for the Steiner tree problem", *J. Algorithms*, 17:381–408 (1994).

[4] M. Bern and P. Plassmann, "The Steiner problems with edge lengths 1 and 2 ", *Information Processing Letters*, 32:171–176 (1989).

[5] A. Blum, R. Ravi and S. Vempala, "A constant factor approximation for the k-mst problem", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* , 442–448 (1996).

[6] M. Charikar, C. Chekuri, A. Goel, and S. Guha, "Approximation Algorithms for Directed Steiner Problems", *Technical Report STAN-CS-TN-97-56*, Department of Computer Science, Stanford University, March 1997.

[7] T. Cheung, Z. Dai and M. Li, "Approximating the Steiner Problems on Directed Graphs", *Technical Report TR-97-1*, Department of Computer Science, City University of Hong Kong, March 1997.

[8] U. Feige, "A threshold of $\ln n$ for approximating set-cover", *Proceedings of the 28th ACM Symposium on Theory of Computing*, 314–318 (1996).

[9] M. R. Garey and D. S. Johnson, "Computers and Intractability: A guide to the theory of NP-completeness", *Freeman, San Francisco* (1978).

[10] N. Garg, "A 3-approximation for the minimum tree spanning k vertices", *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, 302–309 (1996).

[11] S. Guha and S. Khuller, "Approximation algorithms for Connected Dominating Sets", To appear in Algorithmica. A primary version appeared in *Proc. of 4th Annual European Symposium on Algorithms* (1996).

[12] S. Guha and S. Khuller, "Approximation algorithms for Node Weighted Steiner Trees", Submitted for publication.

[13] G. Korsartz and D. Peleg, "Approximating shallow-light trees", *Proceeding of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 103–110 (1996).

[14] L. Kou, G. Markowsky and L. Berman, "A fast algorithm for Steiner trees", *Acta Informatica*, 15, pp. 141–145 (1981).

[15] P. N. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner trees", *J. Algorithms*, 19(1):104–114 (1995).

[16] M. Karpinsky and A. Zelikovsky, "New approximation algorithms for the Steiner tree problem", *Technical Report, Electronic Colloquium on Computational Complexity* (ECCC): TR95-030 (1995).

[17] S. Ramanathan, "Multicast tree generation in networks with asymmetric links", *IEEE INFOCOM '96, IEEE/ACM Transactions on Networking*, Vol. 4, pp. 558-568 (1996).

[18] H.F. Salama, "Evaluation of multicast routing algorithms for real-time communication on high-speed networks", *IFIP Sixth International Conference on High Performance Networking*, pp. 27-42 (1995).

[19] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs", *Math.Japonica*, Vol. 24, pp. 573–577 (1980).

[20] D. Williamson, M. Goemans, M. Mihail and V. Vazirani, "A primal-dual approximation algorithm for generalized Steiner network problems", *Proceedings of 25th Annual Symposium on the Theory of Computing*, 16–18 (1993).

[21] P. Winter, "Steiner problem in networks: a survey", *Networks*, 17:129–167 (1987).

[22] A. Zelikovsky, "An 11/6 approximation algorithm for the network Steiner problem", *Algorithmica*, 9: 463–470 (1993).

[23] A. Zelikovsky, "A series of approximation algorithms for the Acyclic Directed Steiner Tree problem", *Algorithmica*, 18: 99-110 (1997).