Approximation Algorithms for Directed Steiner Tree Problems *

Moses Charikar[†] Chandra Chekuri[‡] Ashish Goel[§] Sudipto Guha[¶]

Dept of Computer Science
Stanford University
{moses,chekuri,agoel,sudipto}@cs.stanford.edu

March 20, 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 tree 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 the versions [11, 14, 17, 1, 15], 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 $O(k^\epsilon)$ in time $O(kn^{1/\epsilon})$ for any fixed $\epsilon > 0$, where k is the number of terminals. For the Directed Generalized Steiner tree 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.

---

*For the Directed Steiner tree problem, a result similar to ours has been obtained independently in [6] by To-yat Cheung, Zuo Dai and Ming Li of the Department of Computer Science, City University of Hong Kong.

[†]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.

[‡]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.

[§]Supported by ARO Grant DAAH04-95-1-0121 and NSF Grant CCR9304971

[¶]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.

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.

The Steiner tree problem is known to be NP-Hard even when the graph is induced by points in the plane [7], and is MAXSNP-hard [4] for general graphs. The first polynomial time approximation algorithms for this problem were developed by Kou, Markowsky and Berman [11] and by Takahashi and Matsuyama [14]. These algorithms achieved an approximation factor of $2 - \frac{1}{k}$ where k is the number of terminals. Zelikovsky [17] obtained a ratio of $\frac{11}{6}$ and this has been further improved by Berman and Ramaiyer [3] and Karpinski and Zelikovsky [13]. The best ratio achievable currently is 1.644. In a recent breakthrough 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 has several applications in network design, routing in multicast networks and other related areas. See [16] for a survey on the use of Steiner tree problems in networks. From a theoretical point of view it turns out to be useful in a number of reductions of problems involving connectivity and covering including, the Group Steiner tree problem in both directed and undirected graphs, 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 [9]. We also consider the Directed Generalized Steiner tree 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. Constant factor algorithms are known for the undirected case [1, 15].

A fairly easy reduction from the Set cover problem (as has been observed by many others) shows that it is hard to approximate Directed Steiner tree to a factor better than $\ln k$ where k is the number of terminals. 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 [18], where he gives an approximation algorithm of ratio $O(k^\epsilon)$ for any $\epsilon > 0$ for directed *acyclic* graphs. No algorithms were known for the case of general directed graphs.

In this paper, we present a factor $O(k^\epsilon)$ approximation algorithm for any $\epsilon > 0$ for the Directed Steiner tree problem. This is a new and simpler approach compared to Zelikovsky's approach in [18]. Our algorithm can be modified to give an $O(2^{\sqrt{\log k}})$ approximation in quasi-polynomial time. This fact hints at the possibility of obtaining polylogarithmic approximation guarantees for the problem. We also present an algorithm for the Directed Generalized Steiner tree problem which achieves an approximation factor of $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$. Our results imply similar ratios for several other variants including the Group Steiner tree problem and the Node Weighted Steiner tree problem in directed graphs via some well known simple approximation preserving reductions. 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 variants in Section 5.

2 Preliminaries

In this section we will prove some basic lemmas concerning decomposing trees and how to use them to obtain partial solutions for the problem at hand. We define a slightly more general version of the Directed Steiner tree problem below.

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 a $O(k^{\epsilon})$ approximation to the D-STEINER(k, r, X) problem which implies the same ratio for the Directed Steiner tree problem. It is also easy to see that this implies a similar ratio for the generalization of the k-MST problem [5, 8] 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. The following lemma shows that we can partition a tree into two almost equal parts.

Lemma 1 *We can partition any out-tree T into two trees T_1' and T_2' such that $c(T_1') + c(T_2') \leq c(T)$ and $k(T_1') + k(T_2') \geq k(T)$ and $k(T_i') \geq k(T)/3$. Such a partition is called a $\frac{1}{3} - \frac{2}{3}$ partition of T.*

Proof: For any node i in T, let T_i denote the subtree rooted at i. Among all nodes i such that $k(T_i) \geq \frac{2}{3}k(T)$, let x be the node which is farthest away (in terms of number of edges) from the root r. The node x is well defined since $k(T_r) = k(T)$. Let $x_1, \ldots, x_l$ be the children of x and without loss of generality assume that $k(T_{x_i})$ are non-decreasing. From the definition of x, it follows that $k(T_{x_i}) < \frac{2}{3}k(T)$ for all x_i. Suppose there is some i, $1 \leq i \leq l$, such that $\frac{1}{3}k(T) \leq k(T_{x_i}) < \frac{2}{3}k(T)$. Then $T_1' = T_{x_i}$ and $T_2' = T - T_{x_i}$ clearly satisfy the requirements. If there is no such node we have $k(T_{x_i}) < \frac{1}{3}k(T)$ for all i. Consider the largest i such that $\sum_{j \leq i} k(T_{x_j}) < \frac{2}{3}k(T)$. We claim that $\sum_{j \leq i} k(T_{x_j}) \geq \frac{1}{3}k(T)$. Suppose not. It must be the case that $i < l$ since $\sum_{j \leq l} k(T_{x_j}) = k(T_x) \geq \frac{2}{3}k(T)$. We have

$$\sum_{j \leq i+1} k(T_{x_j}) = \sum_{j \leq i} k(T_{x_j}) + k(T_{x_{i+1}}) < \frac{1}{3}k(T) + \frac{1}{3}k(T) = \frac{2}{3}k(T)$$

and this contradicts the choice of i. Let T_1 be the tree rooted at x with $T(x_1), \ldots, T(x_i)$ as its subtrees and let T_2 be $T - T_1$. It is easily verified that T_1 and T_2 satisfy the conditions required of them. ■

We can use this fact to show that, given a tree T, there exist smaller trees whose density is at most a fixed constant times more than that of T.

Lemma 2 *Given a tree T, for any $x \leq k(T)$, there is a tree T_x such that $k(T_x) \geq x$ and $c(T_x) < 3x \cdot d(T)$.*

Proof: We construct a sequence of trees $T = T_0, T_1, \ldots, T_x$ as follows. T_{i+1} is obtained from T_i in the following way: Construct a $\frac{1}{3} - \frac{2}{3}$ partition of T_i to get the trees T_1' and T_2'. Note that

$$\frac{c(T_1') + c(T_2')}{k(T_1') + k(T_2')} \leq \frac{c(T)}{k(T)} = d(T_i)$$

Without loss of generality, suppose $d(T_1') \leq d(T_2')$. Then $d(T_1') \leq d(T_i)$. Let $T_{i+1} = T_1'$. Note that $d(T_{i+1}) \leq d(T_i)$. Also,

$$\frac{1}{3}k(T_i) \leq k(T_{i+1}) \leq \frac{2}{3}k(T_i)$$

. We stop as soon as $x \leq k(T_i) < 3x$ for some i. Since the sequence $\{d(T_i)\}$ is non-increasing,

$$c(T_i)/k(T_i) = d(T_i) \leq d(T_0) = d(T)$$

which implies that

$$c(T_i) \leq k(T_i) \cdot d(T) < 3x \cdot d(T).$$

T_i is the required tree T_x. ■

Definition 3 *Define a* $f(k)$-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 3 *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. $k(T_1) = k_1$.

$$\begin{aligned} d(T_1) &= \frac{c(T_1)}{k_1} \leq f(k)\frac{c(T_{OPT})}{k} && (1) \\ c(T_1) &\leq k_1 \cdot \frac{f(k)}{k} \cdot c(T_{OPT}) \leq \left(\int_{k-k_1}^{k} \frac{f(x)}{x} dx \right) c(T_{OPT}) && (2) \end{aligned}$$

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.

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

Adding (2) and (3), we get

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

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

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 2 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.

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. Of course for this scheme to work, we need to guarantee the existence of a bunch with good density. We use the decomposition lemma (Lemma 2) for this purpose. Consider a subtree of the optimal, T_v with $k(T_v) = x$ terminals and density at most $3d(T_{OPT})$. This subtree of the optimal tree naturally defines a bunch. The cost of the bunch is composed of two parts. The cost of connecting r to v is at most C_{OPT}. The cost of connecting each of the terminals in T_v to v is at most $c(T_v)$. Therefore the cost of the bunch defined by T_v is $C_{OPT} + x \cdot c(T_v) = C_{OPT} + 3x^2 C_{OPT}/k$, and the density of the bunch is $(1/x + 3x/k)C_{OPT}$. Since Lemma 2 guarantees a subtree with the required properties for any x, we can choose $x = \sqrt{k}$ to minimize this expression. This proves the existence of a bunch with density at most $\sqrt{k}C_{OPT}/k = \sqrt{k}d(T_{OPT})$. Combined with the fact that we can find the bunch with the best density in polynomial time, we obtain a $O(\sqrt{k})$ approximation.

An obvious way to improve the result is to find structures which are more general than bunches that approximate the optimal more closely. We obtain an improved $O(k^\epsilon)$ 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 $B_i(k, r, X)$ such that B_i gives an $O(k^{1/i})$-approximation for D-STEINER(k, r, X).

$B_1(k, r, X)$ (and $A_1(k, r, X)$) operates as follows: Determine the k terminals in X closest to r. Connect the root r to each such terminal t by the shortest path from r to t. The tree returned is simply the union of all these shortest paths. Clearly, this is a k-approximation for D-STEINER(k, r, X).

Having defined $B_{i-1}(k, r, X)$, we will construct a partial approximation $A_i(k, r, X)$ for D-STEINER(k, r, X) and prove that $A_i(k, r, X)$ is an $O(k^{1/i})$-partial approximation. Applying Lemma 3, we obtain $B_i(k, r, X)$ which is an $O(k^{1/i})$-approximation algorithm for D-STEINER(k, r, X).

$A_i(k, r, X)$ works as follows:

1. For all nodes v, call $B_{i-1}(k^{1-1/i}, v, X)$. Let T_v be the tree returned by this procedure.

2. Find the node r' for which $d(r, v) + c(T_v)$ is minimized.

3. Add the path from r to r', and the tree $T_{r'}$ and return the resultant tree.

Lemma 4 *$A_i(k,r,X)$ gives an $O(k^{1/i})$ partial approximation for* D-STEINER(k,r,X) *and hence $B_i(k,r,X)$ gives an $O(k^{1/i})$ approximation for* D-STEINER(k,r,X).

Proof: (by induction on i.)

The claim is clearly true for $i=1$. We will prove the claim for i. Consider $A_i(k,r,X)$. Let T_{OPT} be an optimal solution to D-STEINER(k,r,X) and let C_{OPT} be its cost. $k(T_{OPT}) \geq k$. Then,

$$d(T_{OPT}) \leq \frac{C_{OPT}}{k}$$

Setting $T = T_{OPT}$ and $x = k^{1-1/i}$ in Lemma 2, we know that there exists a tree T_1 such that

$$\begin{aligned} k(T_1) &\geq k^{1-1/i} \\ c(T_1) &\leq 3x \cdot d(T_{OPT}) \leq 3\frac{C_{OPT}}{k^{1/i}} \end{aligned}$$

Let T_1 be rooted at r'. Note that $d(r,r') \leq c(T_{OPT})$. The optimal solution to D-STEINER$(k^{1-1/i}, r', X)$ has cost at most $c(T_1) \leq 3\frac{C_{OPT}}{k^{1/i}}$ By the inductive hypothesis, $B_{i-1}(k,r',X)$ gives an $O(k^{1/(i-1)})$ approximation to D-STEINER$(k^{1-1/i}, r', X)$, hence the cost of the tree $T_{r'}$ returned by the call to $B_{i-1}(k^{1-1/i}, r', X)$ is

$$\begin{aligned} c(T_{r'}) &\leq O((k^{1-1/i})^{1/(i-1)})c(T_1) \\ &\leq O(k^{1/i}) \cdot c(T_1) \\ &\leq O(C_{OPT}) \\ d(r,r') + c(T_{r'}) &\leq O(C_{OPT}) \end{aligned}$$

Hence the minimum cost tree T' returned by $A_i(k,r,X)$ has cost $O(C_{OPT})$. Also, the tree spans at least $k^{1-1/i}$ terminals. Hence, the density of the tree

$$\begin{aligned} d(T') &\leq O\left(\frac{C_{OPT}}{k^{1-1/i}}\right) \\ d(T') &\leq O(k^{1/i}) \cdot d(T_{OPT}) \end{aligned}$$

This proves that $A_i(k,r,X)$ gives an $O(k^{1/i})$ partial approximation to D-STEINER(k,r,X).

$B_i(k,r,X)$ uses $A_i(k,r,X)$ to construct an approximation for D-STEINER(k,r,X). Using Lemma 3, we know that $B_i(k,r,X)$ gives a $g(k)$ approximation to D-STEINER(k,r,X) such that

$$g(k) = \int_0^k \frac{f(x)}{x} dx$$

where $f(x) = O(x^{1/i})$. Hence $g(k) = O(k^{1/i})$. This completes the inductive step. ■

An analysis of the constant hidden in the $O(k^i)$ notation reveals that the constant grows fairly rapidly with i. Suppose $B_{i-1}(k,r,X)$ gives a $c_{i-1} \cdot k^{1/i}$ approximation. Suppose, in $A_i(k,r,X)$, we invoke $B_{i-1}(k',v,X)$ with $k' = x \cdot k^{1-1/i}$. Then, by the analysis in the above proof, the cost of the tree returned is at most

$$\begin{aligned} &C_{OPT} + c_{i-1}(xk^{1-1/i})^{\frac{1}{i-1}} \left(\frac{3C_{OPT}}{k}\right)(xk^{1-1/i}) \\ &= (1 + 3c_{i-1}x^{i/(i-1)})C_{OPT} \end{aligned}$$

Hence the density of the tree is at most

$$\frac{(1+3c_{i-1}x^{i/(i-1)})C_{OPT}}{xk^{1-1/i}} \leq \left(\frac{1}{x}+3c_{i-1}x^{1/(i-1)}\right)k^{1/i}\frac{C_{OPT}}{k}$$

Choosing $x = (\frac{i-1}{3c_{i-1}})^{(i-1)/i}$, so as to minimize this density, we get that $A_i(k,r,X)$ gives an $f(k)$ partial approximation where

$$f(k) = i\left(\frac{3c_{i-1}}{i-1}\right)^{(i-1)/i} k^{1/i}$$

Substituting this in $g(k)$, we get

$$g(k) = i^2\left(\frac{3c_{i-1}}{i-1}\right)^{(i-1)/i} k^{1/i}$$

This means that $B_i(k,r,X)$ gives a $c_i k^{1/i}$ approximation where

$$c_i = i^2\left(\frac{3c_{i-1}}{i-1}\right)^{(i-1)/i}$$

Solving for c_i, we get

$$c_i = i\left(\prod_{x=1}^{x=i} x^x\right)^{1/i} 3^{(i-1)/2}$$

or $c_i \approx (3i)^{i/2}$.

Lemma 5 *The running time of algorithm $A_i(k,r,X)$ is $O(k^{1-1/i}\cdot n^i)$ and that of $B_i(k,r,X)$ is $O(k\cdot n^i)$.*

Proof: We shall prove the claim by induction on i. Assume it is true for $i-1$. We shall prove the claim for i. $A_i(k,r,X)$ calls $B_{i-1}(k^{1-1/i},v,X)$ for all nodes v. The number of calls to $B_{i-1}(k^{1-1/i},v,X)$ is at most n, each with a running time of $O(k^{1-1/i}\cdot n^{i-1})$, giving a running time of $O(k^{1-1/i}\cdot n^i)$ for $A_i(k,r,X)$.

$B_i(k,r,X)$ invokes $A_i(k,r,X)$ repeatedly, each invocation takes time $O(k^{1-1/i}\cdot n^i)$ and further, the value of k drops by $k^{1-1/i}$. Thus, we can charge a running time of $O(n^i)$ per unit drop in the value of k. This gives a total running time of $O(k\cdot n^i)$ for $B_i(k,r,X)$.

■

Theorem 1 *For every $i>0$, there is an algorithm which gives an approximation ratio of $(3i)^i k^{1/i}$ for the Directed Steiner tree problem and runs in time $O(kn^i)$.*

Corollary 1 *There is a quasi-polynomial time algorithm which gives an approximation of $O(2^{\sqrt{\log k}})$ for the Directed Steiner tree problem.*

Zelikovsky conjectures in [18] that Directed Steiner tree problem cannot have a subpolynomial approximation guarantee unless $P=NP$. Contrary to his conjecture, we believe that Corollary 1 gives evidence that polylogarithmic approximation guarantees are possible. One reason for our belief is the fact that there is no problem known which has a hardness of $2^{\log^{1-\delta} n}$ for some *fixed* $\delta > 0$.

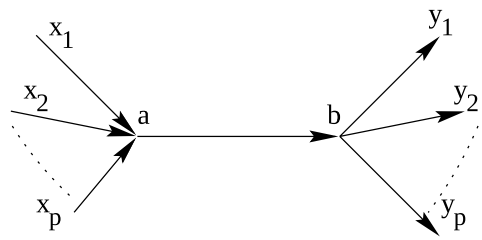

Figure 1: The bunch $< a, b, Y >$

4 Directed Generalized Steiner Trees

In this section we extend some of our techniques from Section 3 to obtain an approximation algorithm for the Directed Generalized Steiner tree 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, 15].

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.

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. A $f(k)$ 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 similar to that for the steiner tree problem. We can obtain a trivial k approximation by connecting each pair $< u_i, v_i >$ with a shortest path between them. As before, if optimal's solution is much better than this, it must be the case that there exists a long path which is shared by many pairs. Our objective is to find such a path and a set of terminals which share it, thus obtaining a subgraph of good density. However, unlike in the rooted case, the structure of the bunches we look for are different and proving the existence of good bunches involves more work. We formally define what we mean by a bunch below.

Definition 4 *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 1 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. So B may not be a subgraph of G. However, given a bunch B we can find a subgraph of G that has cost no more than $c(B)$ and satisfies each of the $k(B)$ node pairs satisfied by B.

In Figure 2 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

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 output B

Figure 2: Algorithm $\mathcal{C}(k, X)$

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 3 to obtain an approximation to DG-Steiner (k, x).

4.1 Finding the Best Bunch

To find the best bunch, we find the lowest cost bunch for all possible values of a, b, and p. We choose the minimum density bunch out of these (at most $n^2 k$) lowest cost bunches. Figure 2 formally describes an algorithm $\mathcal{C}(k, X)$ which finds the best bunch.

The proof of the following lemma is quite simple and hence we omit it.

Lemma 6 *The algorithm $\mathcal{C}(k, X)$ finds a minimum density bunch.*

4.2 Low Density Bunches Exist

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 2 *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 $k \geq 2$,

$$I_{MAX} \leq \left\lceil \log \frac{k^{\frac{1}{3}}}{6 \log^{\frac{1}{3}} k} \right\rceil \leq \log k$$

. 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 more than one shortest paths between u and v, choose one of them arbitrarily.

Claim 1 *There exists an edge $e \in E(H^*_{i'})$ such that at least $\frac{h(k)}{\log k} \cdot 2^{i'-1}$ of the paths pass through e.*

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, proving the above claim.

We now concentrate on the edge $e =< u, v >$ guaranteed by the above claim, and the set X_e 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 $D = MAX_{<a,b>\in X_e}(dist_{T_1}(a, u) + dist_{T_2}(b, v))$. 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 5 *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 $a' \in T_1$ which satisfy the following properties:*

1. *a' belongs to segment $i-1$ of T_1.*
2. *There exists no vertex a in segment $i-1$ of T_1 such that a lies on the path from a' to u.*
3. *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 7 *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 7: To prove this claim, we make the following observations:

$$\begin{aligned} \sum_{1\leq i,j\leq c} k_{ij} &\geq \frac{h(k)}{\log k} \cdot 2^{i'-1} \\ \sum_{1\leq i,j\leq c} n_{ij} &\leq c^2 \end{aligned}$$

The first observation follows from the facts that each node pair that belongs to X_e contributes to k_{ij} for some pair (i, j). For the second observation, we use the fact that $\sum_i n_i^{(1)}$ 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}) . $\sum_j n_j^{(2)}$ is similarly bounded. Now,

$$\begin{aligned}\sum_{1\le i,j\le c} n_{ij} &= \sum_{1\le i,j\le c} n_i^{(1)} n_j^{(2)} \\ &\le (\sum_{1\le i\le c} n_i^{(1)}) \cdot (\sum_{1\le j\le c} n_j^{(2)}) \\ &\le c^2\end{aligned}$$

Since $\sum_{1\le i,j\le c} k_{ij} / \sum_{1\le i,j\le c} n_{ij} \ge (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$, we are guaranteed that there exists a pair $(i,j), 1 \le i,j \le c$ which satisfies the required condition. This completes the proof of Lemma 7. ■

Claim 2 *There exist nodes a' and b' in T_1 and T_2, respectively, and a set $X' \subseteq X_e$ such that $|X'| \ge (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$ and for each node pair $< a,b > \in X'$ 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. *$dist_{T_1}(a,a') \le 2C_{OPT}/c$ and $dist_{T_2}(b',b) \le 2C_{OPT}/c$*

This claim follows from Lemma 7.

Let the bunch B' be defined as $< a', b', X' >$. Let $k' = k(B')$ be the number of node pairs satisfied by this bunch ($k' \ge (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$). 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, $c(B') \le D + 4k' \cdot C_{OPT}/c$. We now give a bound on the density of B'. Recall that $D \le (C_{OPT}/k)h(k) \cdot 2^{i'}$, $c = (\frac{k}{\log k})^{\frac{1}{3}}$, and $h(k) = 6k^{\frac{2}{3}}(\log k)^{\frac{1}{3}}$.

$$\begin{aligned}d(B') &= c(B')/k' \\ &\le D/k' + 4C_{OPT}/c \\ &\le \frac{(C_{OPT}/k)h(k)\cdot 2^{i'}}{(\frac{h(k)}{\log k}\cdot 2^{i'-1})/c^2} + 4C_{OPT}/c \\ &= \frac{C_{OPT}}{k} \cdot (4k/c + 2c^2 \log k) \\ &= \frac{C_{OPT}}{k} \cdot h(k)\end{aligned}$$

■

This completes the proof of Theorem 2. The following corollary follows from Theorem 2 and Lemma 6.

Corollary 2 *The algorithm $\mathcal{C}(k,X)$ produces a $h(k)$ partial approximation to* DG-Steiner (k,x).

Theorem 3 *$\mathcal{C}(k,X)$ 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 tree problem.*

Proof: Follows from Corollary 2 and Lemma 3. ■

5 Group Steiner Tree and Other 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.

5.1 The Group Steiner Tree Problem

The Group Steiner Tree Problem is the following: Given a graph $G(V, E)$, a root r, and a collection of groups $X = \{g_1, g_2, ..., g_k\}, g_i \subseteq V$, 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. Also, as in the Directed Steiner Tree Problem, there can be a variant where $|X| > k$ but we need to connect only k of the groups to the root.

It is easy to see that the Undirected Group Steiner Tree Problem is at least as hard as the Minimum Connected Dominating Set Problem in undirected graphs and is therefore unlikely to be approximated below a factor of $\log n$. Also, 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. Draw (directed) zero cost edges from x_i to each of the vertices in g_i. We now have a directed graph even if we started out from an undirected one. Construct an instance of the Directed Steiner Tree Problem on this new graph with the given root r and with $x_i \ldots x_k$ as the vertices that need to be connected. It is easy to see that any solution to the Group Steiner Tree Problem corresponds to an equal-cost solution of the Directed Steiner Problem, and vice versa. This gives an $O(k^{\epsilon})$ approximation for the Directed as well as the Undirected versions of the Group Steiner Tree Problem.

5.2 Node Weighted variants

For directed problems there is a simple approximation preserving reductions between the node weighted and the edge weighted cases. Therefore all our results carry over to the node weighted case as well.

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

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

[8] 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.

[9] 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).

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

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

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

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

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

[15] 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.

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

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

[18] A. Zelikovsky, "A Polynomial-Time Subpolynom-Approximation Scheme for the Acyclic Directed Steiner Tree Problem" , Tech. Report 85113-CS, University of Bonn, 1994.