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†† University of Waterloo

Abstract

We give the first non-trivial approximation algorithms for the Steiner tree problem and the generalized Steiner network problem on general *directed* graphs. These problems have several applications in network design and multicast routing. For both problems, the best ratios known before our work were the trivial $O(k)$-approximations. For the directed Steiner tree problem, we design a family of algorithms that achieves 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. Thus, an $O(k^\epsilon)$ approximation ratio can be achieved in polynomial time for any fixed $\epsilon > 0$. Setting $i = \log k$, 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 set TSP problem and several others in both directed and *undirected* graphs can be reduced in an approximation preserving fashion to the directed Steiner tree problem. Thus we obtain the first non-trivial approximations to those as well. All these problems are known be as hard as Set cover to approximate.

Key words: Approximation algorithm, directed graph, Steiner tree problem.

*This paper reports the combined version of the two papers [6] and [7] the results of which were obtained independently by the respective authors. A preliminary version appeared in the Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 1998 [5].

†Corresponding author. 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.stanford.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`.

§Department of Computer Science, City University of Hong Kong, 83 Tat-chee Avenue, Kowloon, Hong Kong. Tel:852-27887137, Fax: 852-27888614, 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. The work was done when the author was visiting City University of Hong Kong. Email: `mli@math.uwaterloo.ca`.

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 [10], and is MAX SNP-hard [3] for general graphs. The undirected version has been very well studied and algorithms achieving constant factors have been given by several authors [17, 23, 26, 2, 19].

The directed version of the Steiner tree problem is a natural extension of the undirected version and is defined as follows. Given a directed weighted 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 Steiner tree problem and its variants have several applications in network design and network routing. For example, multicasting involves the distribution of the same data from a central server to several nodes in the network and the problem is to choose a set of edges (or communication links) of minimum cost for the server to route the data. The connection between this problem and the Steiner tree problem is clear. For a general survey on the use of Steiner tree problems in networks, see [25]. Most of the theoretical work till now has focussed on the undirected versions of the Steiner problems. However, there are many networks in practice where the communications links are asymmetric and cannot be modeled by undirected edges. The problem of multicast routing in asymmetric networks has received considerable attention recently [21, 22].

From a theoretical point of view, the directed version of the Steiner tree problem is very useful for the following reason. 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 [20], node weighted Steiner tree problem [14, 18], several interesting problems in connected domination, namely edge weighted connected dominating sets, group (or set) TSP, node weighted Steiner connected domination, and others (see [13] for some of the reductions). These problems arise in several different applications and no non-trivial approximation algorithms were known for any of these. All of the above problems are known to be hard to approximate within a logarithmic factor from reductions via Set cover. Our result on directed Steiner tree gives a unified approximation algorithm for these problems. In addition, from a hardness of approximation point of view our result gives some evidence that these problems are in the class of problems approximable to within polylogarithmic factors in polynomial time. In fact for the group Steiner tree problem on undirected graphs, Garg et al. [12] have very recently obtained a polylogarithmic approximation.

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, 24]. Agrawal et al. [1] 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. Very recently, Jain [15] gave a factor 2 approximation algorithm for this problem. In this paper we restrict ourselves to the 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 $NP \subseteq DTIME[n^{O(\log \log n)}]$ [9]. 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 [27], 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. Zelikovsky further conjectures that no subpolynomial approximation guarantees are possible unless $P = NP$. There is a simple reduction from arbitrary directed graphs to acyclic graphs, therefore Zelikovsky's results carry over to the general case. In this paper, we present a factor $i(i-1)k^{1/i}$ approximation algorithm which runs in time $O(n^i k^{2i})$ for any $i > 0$ for the directed Steiner tree problem. Our approach is simpler compared to that of Zelikovsky [27]. Further, setting $i = \log k$, we obtain an $O(\log^2 k)$ approximation in $O(n^{3 \log k})$ time. Our result gives evidence that the problem can be approximated to within polylogarithmic factors in polynomial time contrary to Zelikovsky's conjecture. This is our main contribution. 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 $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$ where k is the number of pairs that need to be connected.

Related Work: Closely related to our work is the paper of Kortsarz and Peleg [16] on approximating shallow-light trees in undirected graphs that predates the conference version of this paper [5] . Such trees have applications in multicast routing with bounded path lengths. Our greedy algorithm for the directed Steiner tree problem is very similar to that in [16] for undirected shallow light trees. The reason for this is the close connection between the two problems that was first noticed by the authors of this paper. Kortsarz and Peleg claim a $O(\log^2 n)$-approximation in quasi-polynomial time for their algorithm. However the analysis of their algorithm is incorrect for a subtle technical reason. Our analysis, though similar to theirs, differs in that technical detail and we provide a correct analysis that yields our claimed bound.

Our contribution is the correct analysis, and generalizing the algorithm to the directed Steiner tree problem and related problems. We also provide the first approximations for the directed generalized Steiner network problem.

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 are given a directed graph $G(V,E)$ with costs associated with edges. Let $c(e)$ denote the *cost* of edge e. 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 [4, 11] to directed graphs, where the objective is to find an arborescence on k vertices of G of minimum cost.

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. Our algorithm is based on repeatedly finding trees with good density each spanning only a subset of the terminals. The final solution is the union of all these trees. The following definition of a partial approximation procedure and the next lemma relate the quality of the final solution to the density of the trees found at each step.

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 $k_1 = k$, T_1 is returned. Otherwise $\mathcal{B}(k, r, X)$ returns the union of T_1 and the tree returned by a recursive call to $\mathcal{B}(k - k_1, r, X - X_1)$ where X_1 is the set of terminals spanned by T_1.

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 $\mathcal{B}(k, r, X)$ gives 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. $k(T_1) = k_1$.

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

$$c(T_1) \leq k_1 \cdot \frac{f(k)}{k} \cdot c(T_{OPT}) \tag{2}$$

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

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 X, 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.

$$c(T_2) \leq \left(\int_0^{k-k_1} \frac{f(x)}{x} dx \right) c(T_{OPT}) \tag{4}$$

Adding (3) and (4), 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 [27]) *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).

3 Directed Steiner Tree Problem

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 the optimal tree, 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. 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 tree 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) and runs in time $n^{O(i)}$. We can set $i = \log k$ to obtain an $O(\log^2 k)$ algorithm in quasi-polynomial time.

Algorithm $A_i(k,r,X)$
OUTPUT: `A tree` T `rooted at` r `that satisfies at least` k `terminals in` X.

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

1 $T \leftarrow \phi$;

2 while `k>0`

    3 $T_{BEST} \leftarrow \phi$;
    4 for each vertex $v \in V$, and each $k', 1 \le k' \le k$
        5 $T' \leftarrow A_{i-1}(k', v, X) \cup \{(r,v)\}$;
        6 if $d(T_{BEST}) > d(T')$ **then** $T_{BEST} \leftarrow T'$
    7 $T \leftarrow T \cup T_{BEST}$; $k \leftarrow k - |X \cap V(T_{BEST})|$; $X \leftarrow X - V(T_{BEST})$

8 return T

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

Let $T_i(k,r,X)$ refer to the tree returned by $A_i(k,r,X)$. Let $T^{(i)}_{OPT}(k,r,X)$ denote the optimum i-level tree that solves D-STEINER(k,r,X) with cost $C^{(i)}_{OPT}(k,r,X)$, and let $d^{(i)}_{OPT}(k,r,X) = C^{(i)}_{OPT}(k,r,X)/k$ be its density.

$A_1(k,r,X)$ 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 $k', 1 \le k' \le k$ 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.

We now prove the following lemma 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 $A_i, i \ge 2$ in step 7 have the following property: $d(T_{BEST}) \le (i-1) \cdot d^{(i)}_{OPT}(k,r,X)$, 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 at most an $i-1$ level tree. We can always add zero cost edges and assume that T_v is exactly an i level tree. Notice that this is why the algorithm does not exclude the vertex while finding its best ratio subtree of lower level. 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). Let v be the child of r which minimizes 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 behavior of the algorithm $A_{i-1}(k_v, v, X)$. Let s_j denote the total number of terminals picked up by $A_{i-1}(k_v, v, X)$ at the end of the jth iteration of the while loop. Since $A_{i-1}(k_v, v, X)$ eventually picks at least k_v terminals, there is an l such that $s_l < k_v/(i-1)$ and $s_{l+1} \ge k_v/(i-1)$. Let $k_1 = s_l$ and $k_2 = s_{l+1}$ and let T_1 and T_2 denote the corresponding

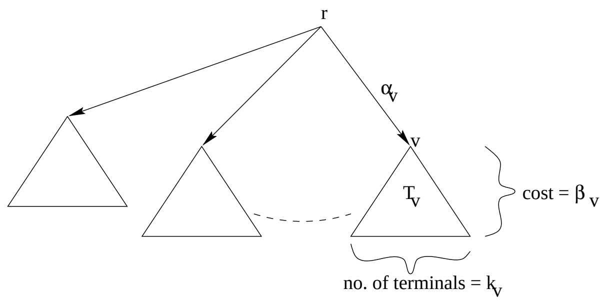

Figure 2: The optimum i level tree

trees constructed by $A_{i-1}(k_v, v, X)$ at end of iterations l and $(l+1)$ respectively. At the end of iteration l we observe that $|T_v \cap X| \geq k_v - k_1 \geq k_v - k_v/(i-1) = k_v(i-2)/(i-1)$. Thus since so many terminals are left and there exists a tree of cost at most β_v to connect them, the density of the optimal $i-1$ level tree rooted at v before the start of the $(l+1)$st iteration can be no more than $(\beta_v/k_v) \cdot (i-1)/(i-2)$.

¿From our inductive hypothesis it follows that the density of the tree found in the $(l+1)$st iteration can be no worse than $(\beta_v/k_v) \cdot (i-1)$. Also notice that the density of the best $(i-1)$ level tree rooted at v can be bounded by the same quantity for the first l iterations as well. Therefore it follows that the density of the tree T_2 at the end of iteration $(l+1)$ is at most $(\beta_v/k_v) \cdot (i-1)$ and further the number of terminals in T_2 is k_2 which is at least $k_v/(i-1)$.

Notice that the algorithm $A_{i-1}(k_2, v, X)$ would produce exactly the tree T_2 described above. When $A_i(k, r, X)$ invokes $A_{i-1}(k_2, v, X)$, it would get back a tree satisfying $k_2 \geq k_v/(i-1)$ terminals. Taking into account the connection cost from r to v, the density of this tree would be at most $\alpha_v/k_2 + (\beta_v/k_v) \cdot (i-1) \leq (i-1)d_v$. Recall that $d_v \leq d^{(i)}_{OPT}(k, r, X)$.

This completes the proof of the inductive hypothesis. □

Theorem 4 *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 $O(n^i k^{2i})$.*

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

Proof: Follows from Theorem 4. □

Zelikovsky conjectures in [27] that directed Steiner tree problem cannot have a subpolynomial approximation guarantee unless $P = NP$. Contrary to his conjecture, Theorem 5 gives some 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 from the root to each of the terminals). This problem where the number of hops are bounded by d, has been studied by Kortsarz and Peleg in [16] in the undirected setting where they refer such trees "shallow-light trees". They prove a lemma very similar to Lemma 3. However, their proof is incorrect. They claim an $O(i^2 \log k \cdot k^{1/i})$ approximation for shallow-light trees in time $n^{O(1/i)}$, but in fact their results only guarantee an $O(2^i k^{1/i})$ approximation in time $n^{O(1/i)}$. Based on this for fixed d their algorithm will only yield a $O(2^d \log n)$ approximation and an $O(2^i k^{1/i})$ approximation in the general case. This does not lead to a polylogarithmic approximation in quasi-polynomial time. Our proof of lemma 3 can be used to get an algorithm that achieves the bounds claimed in [16]. The algorithm repeatedly contracts the tree $A_d(k, r, X)$ in the first case and $A_i(k, z, X)$ in the second. Based on this we can prove the following theorem for shallow-light trees:

Theorem 6 *For any fixed $d > 0$, there is an algorithm which runs in time $n^{O(d)}$ and gives a tree of cost at most $O(d \log k)$ times the cost of the best Steiner tree with path lengths bounded by d. Furthermore for arbitrary d, there is an algorithm which achieves an approximation ratio of $i(i-1)k^{1/i}$ and runs in time $n^{O(i)}$ for all $i > 1$. Setting $i = \log k$ we obtain an $O(\log^2 k)$ approximation in quasi-polynomial time.*

We briefly explain the technical difference between our analysis and that of Kortsarz and Peleg [16]. In our algorithm, the recursive procedure $A_i(k, r, X)$ returns a tree with at least k terminals (if there are k reachable from r in the first place) but it could have many more than k terminals. However the guarantee we make is that the density, or in other words the cost per terminal, of the tree returned is no worse than certain factor away from the optimal tree's (with k terminals) density. The analysis of [16] does not make use of this subtelity. This results in an exponential dependence on the number of levels.

4 Directed Generalized Steiner Network Problem

Formally the problem is the following. Given a directed graph $G(V, E)$ and a set $X = \{(a_i, b_i)\}$ of k node pairs, find the minimum cost subgraph H of G such that for each node pair $(a_i, b_i) \in X$, 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, 24]. However the primal-dual algorithm does not extend because the cuts do not define a submodular function due to asymmetry, for definition of submodular function see [24].

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 $H^*(k, X)$ denote an optimal solution to DG-Steiner (k, X) and let C_{OPT} denote its cost. Define $k(H)$ to be the minimum of k and the number of node pairs in X that get satisfied

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

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

for all pairs $(u, v) \in V(G) \times V(G)$

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

  sort X in increasing order of s. Let (a_j, b_j) refer to the jth pair in this sorted list.

  for p going from 1 to k

    $C \leftarrow c((u, v)) + s[a_1, b_1] + s[a_2, b_2] + \ldots + s[a_p, b_p]$

    if $C/p \leq d$ **then** $(d \leftarrow C/p;\ B \leftarrow (u, v, \{(a_1, b_1), \ldots, (a_p, b_p)\})$

return B

Figure 3: Algorithm $\mathcal{C}(k, X)$

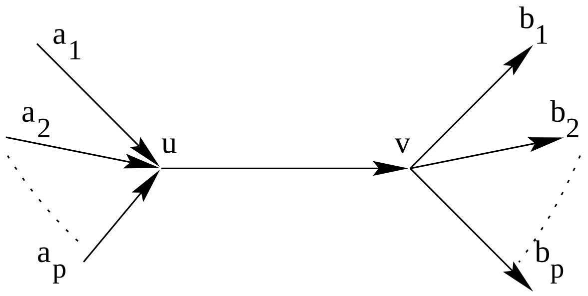

Figure 4: The bunch (u, v, Y)

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).

Recall 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 $Y = \{(a_1, b_1), \ldots, (a_p, b_p)\}$ be a subset of X containing $p \leq k$ node pairs. Also let u and v be vertices of G. A* bunch *$B = (u, v, Y)$ is defined as a graph with the vertex set $\{u, v, a_1, a_2, \ldots a_p, b_1, b_2, \ldots b_p\}$. B has an edge (u, v) of cost identical to that of the edge (u, v) in the graph G. B also has edges (a_i, u) and (v, b_i), with costs identical as in G.*

Figure 4 illustrates the structure of a bunch. The density of a bunch is defined to be the ratio of its cost to the number of pairs it connects. In Figure 3 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 $O(d(H^*(k, X)) \cdot k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$. 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 u, v, 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 7 *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 8 *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 $(a, b) \in X$ such that the distance between a and b is no more than $(C_{OPT}/k) \cdot h(k)$. Then $(a, b, \{(a, b)\})$ 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 (a, b) such that the distance from a to b 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 $(a, b) \in X_{i'}$, let $P_{H^*_{i'}}(a, b)$ be the shortest path between a and b in $H^*_{i'}$. If there are multiple shortest paths between a and b, 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 $i \geq 1$ of T_1 if $dist_{T_1}(a, u) \in \left[i \cdot \frac{C_{OPT}}{c}, (i + 1) \cdot \frac{C_{OPT}}{c}\right]$. Similarly, node b belongs to segment $j \geq 1$ of T_2 if $dist_{T_2}(v, b) \in \left[j \cdot \frac{C_{OPT}}{c}, (j + 1) \cdot \frac{C_{OPT}}{c}\right]$. If a node is within a distance C_{OPT}/c of the root, it is still said to belong to segment 1.

Definition 6 *We define k_{ij} $(1 \leq i, j \leq c)$ 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 $n^{(1)}_i$ $(i \geq 2)$ be the number of nodes a' in a segment $s' \geq i - 1$ of T_1 which satisfy the following properties:*

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

2. *There exists a vertex a in some segment $s \geq i$ 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 completely cross segment $i-1$ of T_1.

Lemma 9 *There exist $1 \leq i, j \leq c$ such that $k_{ij}/n_{ij} \geq (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$.*

Proof: 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 $\sum_i n_i^{(1)}$ (and also $\sum_j n_j^{(2)}$) 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 9, we can claim the existence of nodes $a' \in T_1$ and $b' \in T_2$ and a set $X' \subseteq X$, $|X'| \geq (\frac{h(k)}{\log k} \cdot 2^{i'-1})/c^2$ such that the following properties hold for all $(a,b) \in X'$:

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 10 *$\mathcal{C}(k, X)$ 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 8 and Lemma 7 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. □

We can construct an example where the algorithm constructs a solution whose cost is $\Omega(\sqrt{k})$ times the optimal cost. Consider the following example (see Figure 5). The vertices are divided into two sets of *blocks*. There are k pairs in all. The k nodes at which the directed paths are required to start are called source nodes. The nodes at which the paths end are destinations. These vertices are disjoint from the source nodes. The source vertices are present in $\sqrt{k}$ blocks, each block having $\sqrt{k}$ vertices in turn. The nodes in a block are connected to a vertex (one separate vertex for each block) with 0 cost edges. These $\sqrt{k}$ vertices are connected to a special vertex u with edges of cost 1. A source vertex is indexed by the ordered pair (i,j) where

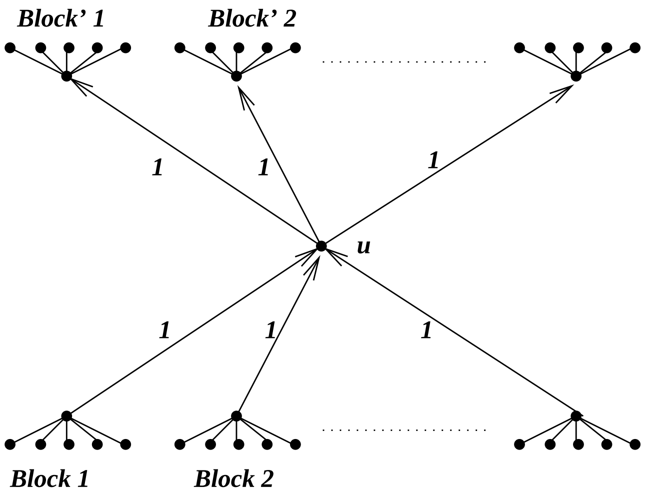

Figure 5: Lower bound for algorithm

$1 \leq i, j \leq \sqrt{k}$ and the node is the jth node in ith block. The destination nodes are numbered similarly. The source node (i, j) and the destination node (j, i) are the pairs.

The figure depicts the optimal tree. The cost of of the optimal solution is $2\sqrt{k}$. But it is easy to observe that there exists no bunch of density less than 1. Therefore the cost of the solution produced by the algorithm is at least k. This shows that the algorithm cannot achieve an approximation guarantee better than $\sqrt{k}/2$.

Remark 1 *Since k could be $\Theta(n^2)$, the approximation ratio we guarantee is $\omega(n)$. 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.*

The $O(n^{1+\epsilon})$ result follows if one solves a directed Steiner tree instance for each possible root vertex u with the set of terminals $\{v|(u, v)$ is a specified pair$\}$. Recently, Dodis and Khanna [8] showed that, unless $NP \subseteq DTIME(n^{polylog(n)})$, there is no polynomial time $O(2^{\log^{1-\epsilon} n})$-approximation algorithm for the DG-Steiner problem. The generalized problem thus appears to be much harder than the rooted version. Interestingly, the hardness reduction in [8] uses a graph in which no path has more than 3 edges. Thus it is unlikely that an analog of Zelikovsky's lemma 2 can be extended to the DG-Steiner problem.

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. This problem arises in wire routing in VLSI design [20]. 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 on undirected graphs can also be reduced similarly. For the group Steiner tree problem on undirected graphs, Garg et al. [12] have recently obtained a polylogarithmic approximation.

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 polylogarithmic approximation ratio for the directed Steiner tree problem. Our results give some evidence of the existence of such an algorithm. For the group Steiner problem on undirected graphs Garg et al. [11] give an $O(\log^2 n \log\log n \log k)$ approximation in polynomial time. Our result gives an $O(\log^2 k)$ approximation in quasi-polynomial time. It would be interesting to improve the results in [11] to obtain a ratio that depends poly-logarithmically only on k, and not on n. For the Steiner network problem, we are able to prove a lower bound of $\Omega(\sqrt{k})$ on the performance of our algorithm, however the upper bound we can guarantee is only $O(k^{\frac{2}{3}} \log^{\frac{1}{3}} k)$. We would like to see the gap bridged. Improving the approximation guarantee for this problem is an interesting open problem though the best we can hope for, in light of the hardness result in [8], is a polynomial factor. 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. No non-trivial upper bound is known for this problem.

Acknowledgments

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

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

[4] 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).

[5] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation Algorithms for Directed Steiner Problems", In *Proceedings of Ninth ACM-SIAM Symposium on Discrete Algorithms*, 192–200,1998.

[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] Y. Dodis and S. Khanna, "Designing Networks With Bounded Pairwise Distance", to appear in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, (1999).

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

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

[11] 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).

[12] N. Garg, G. Konjevod and R. Ravi, "A polylogarithmic approximation algorithm for the group Steiner tree problem", *Proceeding of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (1998).

[13] S. Guha and S. Khuller, "Approximation algorithms for Connected Dominating Sets", Algorithmica (1998) 20:347-387. A preliminary version appeared in *Proceedings of 4th Annual European Symposium on Algorithms* (1996).

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

[15] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem", *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 448–457, (1998).

[16] G. Kortsarz and D. Peleg, "Approximating shallow-light trees", *Proceeding of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 103–110 (1997).

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

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

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

[20] G. Reich and P. Widmayer, "Beyond Steiner's problem: A VLSI oriented generalization " *Proc. of the 15-th Int. Workshop on Graph-Theoretic Concepts in Computer Science, LNCS 411* (1989)

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

[22] 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).

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

[24] 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).

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

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

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