On Directed Steiner Trees

Leonid Zosin* Samir Khuller†

Abstract
The directed Steiner tree problem is the following: given a directed graph $G = (V, E)$ with weights on the edges, a set of terminals $S \subseteq V$, and a root vertex r, find a minimum weight out-branching T rooted at r, such that all vertices in S are included in T. This problem is known to be NP-hard. Recently, non-trivial polynomial time approximation algorithms have been developed for this problem with worst case approximation guarantees of $O(k^\epsilon)$ for any fixed $\epsilon > 0$. We consider a natural LP relaxation of this problem. Using a dual formulation we construct a simple deterministic $(D + 1)$-approximation algorithm for a special case when the subgraph induced by $V \setminus S$ is a tree with depth D (for example, this can be shown to include the group Steiner tree problem as a special case, by the loss of poly-log factors in the approximation guarantee). We also show that this LP has an integrality gap of $\Theta(\sqrt{k})$ for the general problem.

1 Introduction

The directed Steiner tree problem is the following: given a directed graph $G = (V, E)$ with weights on the edges, a set of terminals $S \subseteq V$, and a root vertex r, find a minimum weight out-branching T rooted at r, such that all vertices in S are included in T. (We will refer to this minimum weight out-branching as a directed tree.) The weight of the branching is defined as the sum of the weight of the edges.

The undirected Steiner tree problem has been widely studied, and several constant factor approximation algorithms have been obtained for it. Generalizations of Steiner trees, such as the directed Steiner tree problem and the group Steiner tree problem have only recently received attention in terms of worst case approximation guarantees. (There are several heuristics for these problems that appear to do reasonably well, but they had no good worst case approximation guarantees.) There are several applications of these problems in networks and VLSI design.

This problem is known to be NP-hard [4]. There are many papers that give approximation algorithms for this problem starting with the work by Zelikovsky [7], and more recently by Charikar, Chekuri, Cheung, Dai, Goel, Guha and Li [2]. In these papers a polynomial time algorithm is developed with an approximation ratio of $O(k^\epsilon)$ for any fixed $\epsilon > 0$. In [2] this is extended to a poly-log approximation algorithm, but the running time is not polynomial. (For other results, see the references [2, 7].)

This problem is a generalization of the group Steiner tree problem, for which a polylog approximation was obtained by Garg, Konjevod and Ravi [5]. In the group Steiner tree problem, we are given an undirected graph G with subsets $S_j \subseteq V (j = 1 \ldots k)$ and we wish to compute a minimum weight tree T such that for each $j = 1 \ldots k$, we include at least one vertex from S_j in the tree T.

We now describe the main ideas behind the algorithm for computing a directed Steiner tree. Even though the algorithm was developed in an attempt to obtain a poly-log approximation for the general directed Steiner tree problem, the algorithm does not give an approximation bound that is poly-log in k; in fact we are able to establish an integrality gap of $\Theta(\sqrt{k})$. Since this example has size exponential in k, it suggests that bounds that are poly-log in n and k are possible (and likely). Using a dual formulation we construct a simple deterministic $(D + 1)$-approximation algorithm for a special case when the subgraph induced by $V \setminus S$ is a tree with depth D. In fact, the algorithm yields an alternate method for obtaining a poly-log approximation factor for the group Steiner tree problem. See [3] for other poly-log factor (deterministic) algorithms for the group Steiner tree problem. Our approximation factor is the same, but our algorithm for rounding the fractional soution is simpler than previous deterministic algorithms.

2 Directed Steiner Tree Algorithm

We assume that between each ordered pair of vertices there is an edge of weight equal to the shortest distance between the pair of vertices.

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

The following lemma proven by Zelikovsky [7] shows the following extremely useful property.

LEMMA 2.1. *For all $\ell \geq 1$, there exists an ℓ level tree*

*Current address: ITG Inc. 44 Farnsworth Str. Boston, MA 02210 Part of this work was done while visiting the University of Maryland. E-mail:`lzosin@itginc.com`.

†Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:`samir@cs.umd.edu`. Research supported by NSF awards CCR-9820965 and CCR-0113192.

that provides a $k^{\frac{1}{\ell}}$ approximation to the optimal Steiner tree.

In other words, suppose we work with the following "layered" graph G_ℓ. Create ℓ copies of the vertices of $G-r$. These are denoted by $V_1, \ldots, V_\ell$, and let $V_0 = \{r\}$. Introduce a complete (directed) subgraph between V_i and V_{i+1} $i = 0 \ldots (\ell - 1)$. The weight of each such edge from $u \in V_i$ to $v \in V_{i+1}$ is exactly the distance between u and v in G. Restrict the set of terminals S to a subset of V_ℓ.

Suppose there is an optimal solution T_{OPT} of weight C_{OPT} in G then by Lemma 2.1, there is a tree T in the graph G_ℓ rooted at r with weight at most $k^{\frac{1}{\ell}} \cdot C_{OPT}$ that includes all the terminals in S.

By making $\ell = \log k$, we get a ℓ-level tree that is of cost at most $2C_{OPT}$.

The density of a tree with cost C is defined as the ratio of its cost to the number of terminals it reaches (say k'), namely C/k'. A *minimum density ℓ-level Steiner tree* is an ℓ-level directed tree that reaches $k' \geq 1$ terminals from S, and is a tree with minimum density.

The following lemma is easy to prove.

LEMMA 2.2. *There is an ℓ-level tree in G_ℓ with density at most $\frac{k^{\frac{1}{\ell}} C_{OPT}}{k}$ where k is the number of terminals in S.*

One can solve the directed Steiner problem by designing an approximation algorithm for the minimum density ℓ-level Steiner tree problem. If we have an α approximation for the minimum density Steiner tree problem, by repeatedly finding (approximately) minimum density trees, we can obtain an approximation ratio of $O(\alpha \log k)$ for directed Steiner trees. (This follows from standard techniques that are used for set cover type problems. In the set cover problem, since we can find a minimum ratio set, $\alpha = 1$ and we obtain a $\log k$ approximation.)

We find a low density tree, delete the terminals that are reached by this tree (remove them from S), find another low density tree, and repeat this until we reach all the terminals. To obtain a solution we simply take all the low density trees chosen in each iteration.

3 Finding Low Density Trees in Level Graphs

Let d_{OPT} be the density of the minimum density ℓ-level Steiner tree.

The main approach to this problem is to define a linear programming instance, whose optimal value is at most the density of the minimum density ℓ-level Steiner tree. Suppose this optimal LP value is z^*. We then write the LP-dual for this LP instance, and solve the dual to optimality, obtaining the same optimal value $z^* \leq d_{OPT}$.

We give an algorithm that uses the fractional dual solution to build a low density tree. We prove that if the algorithm outputs a non-trivial tree T then its density is at most αz^* where $\alpha = D + 1$, where D is the depth of the tree. While our algorithm and its description are given for layered graphs; we will show that if the algorithm is actually run on a graph with the property that the subgraph induced by the vertices after deleting the terminals is a tree, then the algorithm always terminates with a non-trivial low density tree. This suffices to yield a poly-log approximation for the group Steiner tree problem.

The following unit-flow LP is defined for the ℓ-level graph G_ℓ. Let $w(e)$ be the weight of edge e, and x_e a variable which denotes the (fractional) capacity of the edge. Let $f_i(e)$ denote the flow on edge e for commodity i (corresponding to $i \in S$). Note that edge e goes from a vertex in V_i to a vertex in V_{i+1}.

$$
\begin{array}{ll}
\min & \sum_e w(e) x_e \\
f_i(e) \leq x_e & \forall e \text{ and } i \in S \\
\multicolumn{2}{l}{\sum_{e=(u,v)} f_i(e) = \sum_{e'=(v,u)} f_i(e') \text{ for } i \in S \text{ and}} \\
\multicolumn{2}{l}{\quad v \in G_\ell - (S \cup \{r\}) \text{ (Flow conservation)}} \\
\multicolumn{2}{l}{\sum_{i \in S} \sum_{e=(r,v)} f_i(e) = 1 \text{ (unit flow leaves root)}} \\
\multicolumn{2}{l}{\qquad 0 \leq x_e \leq 1}
\end{array}
$$

Note that if there is a tree T' of density d that includes a subset $R \subseteq S$ of cardinality k', then we can set the x_e value for each edge in T' to be $\frac{1}{k'}$. We can then send a $\frac{1}{k'}$ unit of flow along the tree edges, for each commodity $i \in R$. Since there are k' commodities, the total amount of flow leaving the root is 1. The total cost of the fractional solution is exactly $\frac{C(T')}{k'}$, which is d. ($C(T')$ is the cost of the tree.) Since this is a feasible fractional solution for the LP, the optimal solution z^* only has lower cost. Hence $z^* \leq d_{OPT}$.

Let $P(r,i)$ be a directed path in G_ℓ from r to $i, i \in S$. We can view the above LP in the following way. For each path $P(r,i)$ from the root r to terminal i we can define a flow variable $f_{P(r,i)}$ that denotes the flow on this path. The above LP can easily be re-written using these flow variables without any explicit flow conservation constraints. Since each flow path variable carries a certain amount of flow from the root to a terminal, flow conservation on all internal vertices is maintained. We now write the dual LP corresponding to the unit-flow LP described above.

$$
\begin{array}{lll}
\max & z & \\
 & \sum_{i \in S} y_i(e) \leq w(e) & \text{for edge } e \\
 & z \leq \sum_{e \in P(r,i)} y_i(e) & \forall P(r,i), i \in S \\
 & y_i(e) \geq 0 &
\end{array}
$$

Even though this dual is described using an exponential number of constraints, we can define a new set of variables, namely, $P_i(v)$ as the "potential" of node v for terminal i. For every edge $e = (u, v)$ we can define $P_i(v) - P_i(u) = y_i(e)$, and set $P_i(r) = 0$. (The node potentials encode the shortest path length from r to v using the edge length function $y_i(e)$.) Using this alternate dual formulation we have a polynomial number of constraints.

A strictly optimal solution is one where if a constraint is tight, then it is tight in all optimal solutions. We could find such solution, using the following method. First solve the LP and find the optimal value of the objective function. Then add a new constraint, where the objective function is equal to the optimal value. Now for every edge in the graph solve the original LP with the new constraint and an objective function to maximize the slack on the constraint for this edge. If there exists an optimal solution to the original problem where the constraint is not tight, then we should get a solution Y_e to the new problem, corresponding to edge e, with non-zero slack. To build a strictly optimal solution, we will take an average solution $Y = \sum_{e \in E} Y_e/|E|$ where E is set of all edges in the graph. If there is an optimal solution where the constraint for an edge e is not tight then this constraint is not tight for the solution Y_e, and then it is also not tight for the solution Y. Therefore Y is a strictly optimal solution. soln) We will use the strictly optimal solution to the above dual LP, to obtain a low density tree.

We now summarize the steps of the algorithm.

Algorithm Minimal Density Directed Steiner Tree

Input: A directed graph G_ℓ with root r, and a set of terminals $S = \{s_1, \ldots, s_k\}$.
Output: A directed tree rooted at r, spanning $1 \le k' \le k$ terminals in S.

1. Compute a strictly optimal[a] solution for the dual of the unit flow problem.
2. Let $y_i(e)$ ($1 \le i \le k$, $e \in E$) be such a solution. Compute a GOOD tree T using Algorithm GOODTREE.

[a] A strictly optimal solution is one where if a constraint is tight, then it is tight in all optimal solutions.

We will need some definitions to describe Step 2 of this algorithm.

First, we get a graph G' by removing from G_ℓ all edges e where there is a slack in the dual constraint $\sum_{i \in S} y_i(e) < w(e)$. From strict optimality of the solution in the remaining graph, every path from r to a terminal has length z.

DEFINITION 3.1. *For each edge e in G', let $N_{G'}(e)$ be the subset of terminals such that there is a directed path in G' from e to the terminal.*

DEFINITION 3.2. *Let T be a directed sub-tree of G, rooted at r and spanning some set $N(T)$ of terminals. For edge e in T, define the set of terminals $N_T(e)$ such that there exists a path in T from e to the terminals.*

DEFINITION 3.3. *Define edge e in T to be a BAD edge if $N_T(e) < \sum_{x \in S \setminus N_T(e)} \frac{y_x(e)}{z}$.*

We now elaborate on Step 2 of our algorithm.

Algorithm GOODTREE
$G_0 = G'$, $T_0 = P_x$ (where P_x is a directed path in G_0 from r to an arbitrary terminal x);
$i = 0$;
while (T_i contains BAD edges) do
  let e be the furthest BAD edge from r in T_i,
  if $\exists j \in N_{G_i}(e) \setminus N_{T_i}$ then
    $T_{i+1} = T_i + P_j$[a], $G_{i+1} = G_i$;
  else
    $T_{i+1} = T_i -$(subtree rooted at e);
    $G_{i+1} = G_i - e$;
  $i = i + 1$;
  if no BAD edges in T_i output T_i; *stop*.
end-while

[a] take any path in G' from edge e to terminal j

THEOREM 3.1. *If Algorithm GOODTREE terminates with a non-trivial tree T that spans p terminals then density of T is at most $z(D+1)$, where D is the depth of the tree.*

Proof. We can express the weight of the tree $w(T)$ as follows:

$$w(T) = \sum_{e \in T} w(e)$$
$$= \sum_{e \in T} \sum_{x \in S} y_x(e)$$
$$= \sum_{e \in T} \sum_{x \in N_T(e)} y_x(e) + \sum_{e \in T} \sum_{x \in S \setminus N_T(e)} y_x(e)$$

Since no edge of T is BAD we obtain:

$$W(T) \le p \cdot z + \sum_{e \in T} z \cdot |N_T(e)|$$
$$\le p \cdot z + z \cdot D \cdot p$$
$$= p \cdot z \cdot (D+1)$$

The last inequality holds because in T every path has length at most D. Therefore we get that density of T, $w(T)/p$ is at most $z(D+1)$.

THEOREM 3.2. *If the input graph is a tree when we delete the set of terminals and if Algorithm GOODTREE removes the first path from T_i, then the dual solution is not strictly optimal.*

Proof. Define the following modification to the dual solution: At every step when we are removing a BAD edge e from tree T_i, we will add ϵ to $y_x(e)$ for every $x \in N_{T_i}(e)$ and remove $\epsilon \cdot \frac{y_x(e)}{z}$ from $y_x(e)$ for every $x \notin N_{T_i}(e)$. We claim that this modified solution is feasible and has the same value of objective function, but all the BAD edges now have slack. For every BAD edge e we have $N_T(e) < \sum_{x \in S \setminus N_T(e)} \frac{y_x(e)}{z}$ therefore after modification of the dual variables the edge constraint remains feasible and will even have a slack. This modification maintains the same objective function value, and will not decrease the length of any path from r to any terminal. If on some path from r to a terminal x we have a number of BAD edges $e_1, \ldots, e_l$ where we removed $\epsilon \cdot \frac{y_x(e_i)}{z}$ then we will establish that there exists one BAD edge e where we added ϵ, or there is an edge on the path with non-zero slack. Assume for now that there is an edge where we added ϵ. In this case we can upper-bound the total decrease in path length by the increase in the path length as follows.

$$\frac{\epsilon}{z} \cdot \sum_{i=1}^{l} y_x(e_i) \leq \frac{\epsilon}{z} \cdot z = \epsilon$$

This contradicts the strict optimality of the original solution.

The tree T at each step of the algorithm has the following property: All BAD edges are on one path in the tree. This property is maintained from step to step, because we add new paths to the tree from the BAD edge that is the furthest from the root. Suppose there exists a path from the root r to some terminal x, where we delete some BAD edges. If this path includes edges with slack, then we can choose ϵ to be significantly smaller than $\min_{\{e|slack(e)\neq 0\}} \frac{slack(e)}{|S|}$ where $slack(e)$ is the slack on edge e. This ensures that any decrease can be compensated by using the slack. If this path includes only tight edges, suppose e is the furthest (from the root) BAD edge, which we deleted from the path, and T was the current tree at that time. If $x \in N_T(e)$ then we will add ϵ to the $y_x(e)$, this addition will compensate all possible subtractions on this path. If $x \notin N_T(e)$ then x should be currently in T, otherwise we should add the path from e to x to the current tree T instead of deleting e. But we know that all BAD edges in T are on this path and also we know that at some step we will delete terminal x from T. We could do it only by deletion of some BAD edge e' which is on the path from r to x, but then we will add ϵ to the $y_x(e')$; this addition will compensate all possible subtractions on this path.

THEOREM 3.3. *For the group Steiner tree problem, we can obtain a approximation algorithm with ratio $O(\log n \log\log n \log k \log \delta)$ δ is the diameter of the graph and k is the number of groups.*

Proof. We first convert an arbitrary metric to a tree metric (see [1, 3]). This causes an expected increase in length of each edge by an $O(\log n \log\log n)$ factor. Once the problem reduces to a tree, we can root the tree at r and direct all edges away from the root. For each group S_j, we can create a new terminal j with edges from all nodes in S_j to j. This graph satisfies the property that once we delete the terminals, the graph is a tree. We can apply the algorithm described above to obtain a $\log \delta$ approximation factor (depth of the tree) for the densest Steiner tree problem. Repeated applications of finding densest trees yields an extra $O(\log k)$ factor.

Remark: The $\log \delta$ term in the above bound can be replaced by a $\log N$ bound, where N is the maximum size of a group. Thus this method gives the same bounds as the method by Garg, Konjevod and Ravi [5]. (Basically, the depth of the graph can be bounded by $\log N$.)

4 Integrality Gap

Now we define a graph for which an integrality gap between optimal integer solution for the minimum density directed tree problem and it's LP relaxation is as big as $\Theta(\sqrt{k})$.

The graph G consists of 5 levels of nodes and 4 levels of edges. The five levels are:

1. the root r;
2. the $C_k^{\sqrt{k}}$ (k choose $\sqrt{k}$) nodes, a node a_S for each set S of $\sqrt{k}$ elements from k elements;
3. the $C_k^{\sqrt{k}+1}$ (k choose $\sqrt{k}+1$) nodes, a node $b_{S'}$ for each set S' of $\sqrt{k}+1$ elements from k elements;
4. the $C_k^{\sqrt{k}+1}$ (k choose $\sqrt{k}+1$) nodes, a node $c_{S'}$ for each set S' of $\sqrt{k}+1$ elements from k elements;
5. the k terminals (one terminal for each element).

The four levels of edges are defined as follows:

1. from root r to every node a_S on level 2, the weight of this edge is k;

2. from nodes a_S on level 2 to nodes $b_{S'}$ on level 3 according to the inclusion property: there is an edge from a node a_S on level 2 to a node $b_{S'}$ on level 3 if a set$S$ for the node on level 2 is included in a set S' for the node on level 3, the weight of this edge is 0;

3. from every node $b_{S'}$ on level 3 there is one edge to a node $c_{S'}$ on level 4, that corresponds to the same set S' of $\sqrt{k}+1$ elements, the weight of this edge is $\sqrt{k}$;

4. from a node $c_{S'}$ on level 4 to terminal on level 5 if the terminal belongs to the set S' of the node, the weight of this edge is 0.

LEMMA 4.1. *There exists a feasible unit flow in the G with total fractional weight $O(1)$.*

Proof. Push $\frac{1}{(k-\sqrt{k})\cdot C_k^{\sqrt{k}}}$ units of flow of $k-\sqrt{k}$ commodities from root r to every node a_S, such that every commodity corresponds to some terminal i that is not belonging to the set S. The total flow from r will be 1 and total fractional weight of edges on level 1 will be $\frac{1}{1-1/\sqrt{k}}$. From node a_S we push $\frac{1}{(k-\sqrt{k})\cdot C_k^{\sqrt{k}}}$ units of flow of commodity i to the node $b_{S'}$ such that $i \in S'$. From node $b_{S'}$ we push all the flow of $\sqrt{k}+1$ commodities that enter the node to the node $c_{S'}$ and from it to the corresponding terminals. The total fractional weight of edges on level 3 will be $\frac{1}{1+1/\sqrt{k}}$. Therefore the flow is a feasible unit flow with total fractional weight $O(1)$.

LEMMA 4.2. *Every directed tree that spans n terminals in the G, has weight at least $\Theta(n\sqrt{k})$ or density $\Theta(\sqrt{k})$.*

Proof. Let T is a directed tree that spans n terminals. We could assume without lost of generality, that T has only one edge on level 1. If it does not, we could consider it's minimal density sub-tree. If $n \le \sqrt{k}$ than it's weight is at least k (the weight of edge from level 1) and it's density is at least $\sqrt{k}$. If $n \ge \sqrt{k}$ than it has at least $n-\sqrt{k}+1$ edges from level 3 and it's weight is at least $(n-\sqrt{k}+1)\cdot\sqrt{k}+k$, and it's density is at least $\sqrt{k}$.

Comment: One could try to add additional constraints to the original LP relaxation in such a way, that its optimal value will remain at most the density of the minimal density Steiner tree, but the gap between these values will decrease for the graph in the above example. Unfortunately, we were able to find a family of such graphs with a similar integrality gap, for every such modification of the original LP, that we had in mind.

Acknowledgements: We thank Chandra Chekuri for useful discussions.

References

[1] Y. Bartal, "Probabilistic approximations of metric spaces and its algorithmic applications", In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, (1996).

[2] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha and M. Li, "Approximation algorithms for directed Steiner tree problems", In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 192-200 (1998).

[3] M. Charikar, C. Chekuri, A. Goel, S. Guha, "Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-medians", In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 114–123 (1998).

[4] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco, (1979).

[5] N. Garg, G. Konjevod, R. Ravi, "A polylogarithmic approximation for the group Steiner tree problem", In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 253-259 (1998).

[6] R. E. Tarjan, "Data Structures and Network Algorithms", Society for Industrial and Applied Mathematics, (1983).

[7] A. Zelikovsky, "A series of approximation algorithms for the acyclic directed Steiner tree problem", *Algorithmica*, 18(1):99-110, (1997).