Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets*

Sudipto Guha†
Dept. of Computer Science
Stanford University
Stanford, CA 94305

Samir Khuller‡
Computer Science Dept. and UMIACS
University of Maryland
College Park, MD 20742

Abstract

A greedy approximation algorithm based on "spider decompositions" was developed by Klein and Ravi for node weighted Steiner trees. This algorithm provides a worst case approximation ratio of $2\ln k$, where k is the number of terminals. However, the best known lower bound on the approximation ratio is $\ln k$, assuming that $NP \not\subseteq DTIME[n^{O(\log\log n)}]$, by a reduction from set cover [9, 4].

We show that for the unweighted case we can obtain an approximation factor of $\ln k$. For the weighted case we develop a new decomposition theorem, and generalize the notion of "spiders" to "branch-spiders", that are used to design a new algorithm with a worst case approximation factor of $1.5\ln k$. This algorithm, although polynomial, is not very practical due to its high running time; since we need to repeatedly find many minimum weight matchings in each iteration. We are able to generalize the method to yield an approximation factor approaching $1.35\ln k$. We also develop a simple greedy algorithm that is practical and has a worst case approximation factor of $1.6103\ln k$. The techniques developed for the second algorithm imply a method of approximating node weighted network design problems defined by 0-1 proper functions.

These new ideas also lead to improved approximation guarantees for the problem of finding a minimum node weighted connected dominating set. The previous best approximation guarantee for this problem was $3\ln n$ [7]. By a direct application of the methods developed in this paper we are able to develop an algorithm with an approximation factor approaching $1.35\ln n$.

1. Introduction

The Steiner tree problem is a classical problem in networks, and is of wide interest. The problem is known to be NP-hard for graphs, as well as in most metrics [5]. Much effort has been devoted to the study of polynomial time approximation algorithms for this problem [1, 8, 9, 10, 13].

The Steiner tree problem is defined as follows: given a graph $G = (V, E)$ and a subset of vertices $S \subseteq V$ we wish to compute a minimum weight tree that includes all the vertices in S. The tree may include other vertices not in S as well. The vertices in S are also called *terminals* (sometimes these are referred to as "required" vertices). For results on edge weighted problems see [14, 2, 1, 10]. In

*A preliminary version of this paper has been submitted to the *1998 Integer Programming and Combinatorial Optimization Conference*.

†Part of this work was done while S. Guha was at the University of Maryland and his research was supported by NSF Research Initiation Award CCR-9307462. Email addr: `sudipto@cs.stanford.edu`

‡Research supported by NSF Research Initiation Award CCR-9307462, and NSF CAREER Award CCR-9501355. Email addr: `samir@cs.umd.edu`

this paper, we concentrate on the study of the node weighted version, where the nodes, rather than the edges have weights. This is a more general problem, since one can reduce the edge weighted problem to the node weighted problem by subdividing edges and giving the new vertices the weight corresponding to the subdivided edge.

The first non-trivial polynomial time approximation factor for this problem was achieved by Klein and Ravi [9]. Their algorithm achieves a worst case approximation factor of $2 \ln k$ where k is the number of terminals. They showed that the problem is at least as hard to approximate as the set-cover problem, for which a polynomial time approximation algorithm with a factor of $(1-\epsilon)\ln k$ would imply that $NP \subseteq DTIME[n^{O(\log\log n)}]$ [4].

The Klein-Ravi algorithm [9] is based on an earlier heuristic by Rayward-Smith [12] and may be viewed as a generalization of the set-cover greedy approach [3]. In this scheme, at each step a "spider" is chosen so as to minimize the **ratio** of the weight of spider to the number of terminals that it connects. They prove that the process of greedily picking spiders yields a good solution.

Our first algorithm is based on a new decomposition theorem, using which we can establish a bound of $1.5 \ln k$ for the approximation factor. We show that we can decompose the solution into more complex objects called branch-spiders. Unfortunately, finding branch-spiders of minimum ratio is computationally very intensive as it uses weighted matchings repeatedly, so this algorithm is not practical for large graphs. This algorithm is described in Section 3. We then show how to use generalizations of branch-spiders to develop a new algorithm with an approximation factor approaching $1.35 \ln k$.

Our second approach yields a much faster algorithm and also addresses generalizations to the case when the optimal solution is a collection of components. It has a worst case approximation factor of $1.6103 \ln k$. It is not difficult to observe that this algorithm can be extended easily to problems defined by 0-1 proper function[9, 6]. This algorithm is described in Section 4.

In Section 5 we give an algorithm for the case when all the node weights are 1. This algorithm has an approximation factor of $\ln k$.

In Section 6 we show how to use the method developed to solve the node weighted Steiner tree problem to solve the Connected dominating set problem. This improves the $3 \ln n$ factor shown for the weighted CDS problem in [7].

2. Preliminaries

We assume that the input graph is connected and that only the vertices have weights. Without loss of generality the subset of required vertices, also called terminals have zero weight since they are included in every solution. We assume that these have degree one, since for each terminal s, we can create a new vertex s' and add the edge (s, s') and consider s' as the terminal.

Definition 1: *A spider is defined as a tree having at most one node of degree more than two. Such a node (if one exists) is called the center of the spider.*

Definition 2: *An m spider ($m > 2$) is defined as a spider with a center of degree m.*
A 2 spider is one with no node of degree more than two.

An m spider has m leaves, and each path to a leaf from its center is called a *leg*. A 2 spider is a path. (By our assumption on the terminals, all terminals are leaves.)

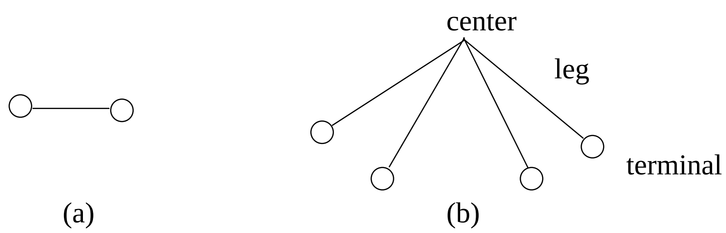

Figure 1: (a) a 2 spider (b) a 4 spider.

Definition 3: *An $m+$ spider ($m > 2$) is defined as a spider with one node of degree at least m.*

The *cost* of a spider is the sum of the weights of the nodes in the spider. The *ratio* of a spider is the ratio of its cost to the number of its terminals. The leaves of a minimal ratio spider are terminals.

Contracting a spider is the operation of contracting all nodes of the spider to form one vertex. If we contract a spider S in graph G, making the contracted vertex into a terminal, then it is easy to argue that the cost of the optimal solution is at most the cost of the spider together with the cost of the optimal solution for the new graph. The shrunken node has zero weight, and we make it a degree one node once again.

Klein and Ravi [9] give an algorithm that repeatedly contracts the min-ratio spider. The reason one obtains a $2 \ln k$ factor is that one may repeatedly contract 2 spiders, and each time a spider is contracted, we create a new terminal to denote the contracted spider (see [9] for the proof). If we could restrict our attention to larger spiders, then we get a better algorithm. However, we cannot show that a decomposition into large spiders exists – to achieve this, we modify spiders into more general structures.

The next two lemmas are due to Klein and Ravi [9], and used in their proof for the approximation factor of $2 \ln k$.

Lemma 2.1: *Any solution to the node weighted Steiner tree problem can be decomposed into spiders having terminals as leaves, without increasing the cost.*

Lemma 2.2: *In iteration i having n_i terminals to connect, the minimum ratio spider has min ratio cost $\gamma_m \leq \frac{w(OPT)}{n_i}$.*

3. Algorithm for Node Weighted Steiner Trees

Before describing the new decomposition theorem, we introduce the notion of branch-spiders.

Definition 4: *A branch is defined as a tree with at most three leaves. We refer to one of the leaves as the root.*

Definition 5: *A branch-spider is constructed by merging the roots of a collection of branches into a single vertex, called the center.*

Fig. 2(a) shows a picture of a branch with three leaves and Fig. 2(b) shows a picture of a branch-spider.

Definition 6: *A* 3+ branch-spider *is one with at least three terminals.*

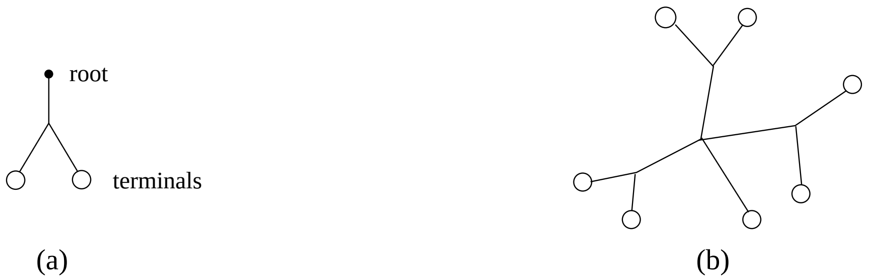

Figure 2: (a) A branch with three leaves. (b) A branch-spider with four branches and seven terminals.

Lemma 3.1: *Any solution to the node weighted Steiner tree problem, with at least three terminals, can be decomposed into* 3+ *branch-spiders having terminals as leaves.*

Proof: Consider a solution to the node weighted Steiner tree problem. This is a tree that includes all the terminals as leaf nodes. The depth of a node is the distance of a node from an arbitrarily chosen root. The theorem can be proven by induction on the number of terminals. Choose a node v of maximum depth, such that the subtree rooted at v contains at least three terminals. We immediately obtain a branch-spider (with at least three terminals) rooted at v. Note that no proper descendant of v has three terminals in its subtree, hence this satisfies the requirement of being a branch-spider. Delete the subtree rooted at v. If the tree still contains at least three terminals, by the induction hypothesis we can find a decomposition of the remaining tree into 3+ branch-spiders, and we are done. If there are at most two terminals, we can attach them to v while maintaining a branch-spider rooted at v. This concludes the proof. □

We now address the issue of computing minimum ratio 3+ branch-spiders. (The ratio of a branch-spider is defined in the same way as for spiders, the total weight divided by the number of terminals that it connects.)

We show how to find a minimum weight branch-spider centered at vertex v that has exactly ℓ terminals in it. Using this procedure it is easy to compute the minimum ratio branch-spider with at least three terminals by simply enumerating over all possible centers and sizes of branch-spiders ($\ell \geq 3$).

Algorithm for finding a minimum weight 3+ *branch-spider* (G^*, v, ℓ)

Step 1. Construct a weighted graph $G'_v = (V'_v, E'_v)$ where $V'_v = \{$ all terminals in $G^*\}$ and $w(x, y) =$ weight of the minimum weight Steiner tree in G^* connecting terminals $\{x, y, v\}$ (in this calculation we do not include the weight of the center v).

Step 2.
Case (a) If ℓ is odd, for each terminal x, we find a a minimum weight matching M_x of cardinality $\lfloor \frac{\ell}{2} \rfloor$ in $G'_v - \{x\}$. The total weight of the spider is $w(v) + w(M_x) + w(x)$ where $w(x)$ is the distance

from v to x in the graph G^*. We take the minimum weight spider over all choices of x.
Case (b) If ℓ is even, we find a a minimum weight matching M of cardinality $\frac{\ell}{2}$ in G'_v. The total weight of the spider is $w(v) + w(M)$.

(The problem of finding a min weight matching of cardinality $\lfloor \frac{\ell}{2} \rfloor$ in $H = (V_H, E_H)$ may be reduced to minimum weight perfect matching by creating $|V_H| - 2\lfloor \frac{\ell}{2} \rfloor$ dummy vertices, and by adding zero weight edges from each vertex in H to each dummy vertex. A minimum weight perfect matching in the new graph permits all vertices, except for $2\lfloor \frac{\ell}{2} \rfloor$ vertices, to be matched at zero cost to the dummy vertices. The remaining vertices are forced to match each other at minimum cost, yielding a matching M of cardinality $\lfloor \frac{\ell}{2} \rfloor$.)

Lemma 3.2: *The algorithm described above computes a 3+ branch-spider of minimum ratio.*

Proof: Let the minimum ratio 3+ branch-spider have its center at vertex v' and have ℓ terminals. A pair of branches, each having a single terminal, can be viewed as a single branch with two terminals. In this way we can pair up branches with only one terminal, leaving at most one unpaired branch. This naturally induces a matching in G'_v of cardinality $\lfloor \frac{\ell}{2} \rfloor$. This shows that there is a matching of size $\lfloor \frac{\ell}{2} \rfloor$, the center vertex and possibly a single branch of total cost at most the cost of the branch-spider. When the algorithm tries v' as its center, with the correct choice of ℓ and tries x as the unpaired branch, we should compute a structure of cost at most the cost of the 3+ branch-spider. □

The algorithm works iteratively. In each iteration, let n_i denote the number of terminals remaining at the start of iteration i. Initially, $n_1 = k$ the number of terminals in S.

Node Steiner Tree Algorithm:I

Repeat the following steps until we have at most two terminals left, and then connect the terminals optimally.

Step 1. Find a 3+ branch-spider in G with minimum ratio.

Step 2. Contract the chosen branch-spider, and update G.

The following lemma can be proven quite easily.

Lemma 3.3: *In iteration i having n_i terminals to connect, the minimum ratio branch-spider has ratio at most $\frac{w(OPT)}{n_i}$.*

Theorem 3.4: *The algorithm described above yields a node weighted Steiner tree of cost at most $1.5 \ln k$ times the optimal, where k is the initial number of terminals.*

Proof: Denote the cost of the spider chosen at iteration i to be C_i.

We will prove that $C_i \leq 1.5 w(OPT) \ln \frac{n_i}{n_{i+1}}$.

Summing up all the C_i values (over all z iterations) gives the required bound.

$$\sum_{i=1}^{z} C_i \leq \sum_{i=1}^{z} 1.5 w(OPT) \ln \frac{n_i}{n_{i+1}}$$

$$\sum_{i=1}^{z} C_i \leq 1.5 w(OPT) \ln k$$

Assume the minimum ratio 3+ branch-spider has t terminals. Since $t \geq 3$, we have $t - 1 \geq \frac{2}{3}t$. From Lemma 3.3,

$$\frac{C_i}{t} \leq \frac{w(OPT)}{n_i}$$

This gives us that $\frac{t}{n_i} \geq \frac{C_i}{w(OPT)}$. Since $n_{i+1} = n_i - (t-1)$, we get

$$n_{i+1} \leq n_i - \frac{2}{3}t \leq n_i(1 - \frac{2C_i}{3w(OPT)}).$$

$$\ln \frac{n_i}{n_{i+1}} \geq -\ln(1 - \frac{2C_i}{3w(OPT)}) \geq \frac{C_i}{1.5w(OPT)} .$$

The last step uses the fact that $-\ln(1-x) \geq x$. We conclude that

$$C_i \leq 1.5w(OPT)\ln \frac{n_i}{n_{i+1}}.$$

□

Further Improvements:
We can improve the approximation ratio by restricting ourselves to minimum ratio objects that have size at least four. However, to perform a decomposition of the optimal solution into structures of size at least four, we need to prove a lemma like Lemma 3.1.

Definition 7: *A bramble is defined as a tree with at most four leaves. We refer to one of the leaves as the root.*

Definition 8: *A bramble-spider is constructed by merging the roots of a collection of brambles into a single vertex, called the center.*

Definition 9: *A* 4+ bramble-spider *is one with at least four terminals.*

Fig. 3(a) shows a picture of a bramble and Fig. 3(b) shows a picture of a bramble-spider.

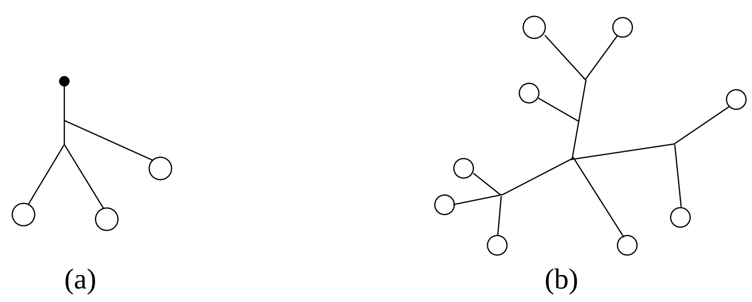

Figure 3: (a) A bramble with four leaves (b) A bramble-spider with four brambles and nine terminals.

The difficulty is that we do not know how to find the min ratio 4+ *bramble-spider* in polynomial time. We will use a greedy approach to approximate this structure.

Node Steiner Tree Algorithm:II

Let $w = w(OPT)/n_i$ and $\delta = 1.35$, and $C = \frac{1}{\epsilon}$.

Repeat the following steps until we have at most C terminals left, and then connect the remaining terminals optimally.

Step 1. Compute the best ratio spider. Ratio is γ_m.

Step 2. Compute the best ratio 3+ spider. Ratio is γ_{3+}.

Step 3. For each $j = 0 \ldots C$ compute the best ratio 4+ branch-spider with at least 4 terminals, and exactly j brambles with three terminals in each, attached to it. This can be done by enumerating over all j subsets of three terminals and then finding the minimum cost branch-spider of each possible size. Let the ratio be γ.

Step 4. Compute γ^{apx}, an approximate min ratio bramble spider that has at least C terminals (We will try each size, and greedily construct one of that size).

Step 5. If $2\gamma_m \leq \delta w$ then shrink spider from step 1. If $1.5\gamma_{3+} \leq \delta w$ then shrink spider from step 2. Else shrink spider achieving the minimum in $\min(\delta\gamma, \gamma^{apx})$.

For this algorithm to work, we have to know the weight $w(OPT)$ of the optimal solution. Since we only know an upper bound on this weight (sum of the weight of all vertices), we have to to "guess" the weight approximately, and run the algorithm for each possible guessed value. Suppose the cost of each iteration is C_i. In each iteration, let n_i denote the number of terminals remaining at the start of iteration i. Initially, $n_1 = k$ the number of terminals in S.

We first prove the following theorem.

Theorem 3.5: *If C_i is the cost of the spider contracted in iteration i, then $C_i \leq \delta(\frac{1+\epsilon}{1-\epsilon})w(OPT)\ln\frac{n_i}{n_{i+1}}$, for any $\epsilon > 0$.*

We will prove Theorem 3.5 in two steps.

Lemma 3.6: *In step 5 if we contract the spider chosen in Step 1 or Step 2, then $C_i \leq \delta w(OPT)\ln\frac{n_i}{n_{i+1}}$.*

Proof: Assume we contracted the spider chosen in Step 1. Suppose the minimum ratio spider has t terminals. Since $t \geq 2$, we have $t - 1 \geq \frac{t}{2}$. By the condition in Step 5, we have

$$2\gamma_m \leq \delta w$$

$$2\frac{C_i}{t} \leq \delta\frac{w(OPT)}{n_i}.$$

Thus $\frac{t}{2} \geq \frac{n_i C_i}{\delta w(OPT)}$. Since $n_{i+1} = n_i - (t-1)$, we get

$$n_{i+1} \leq n_i - \frac{t}{2} \leq n_i(1 - \frac{C_i}{\delta w(OPT)}).$$

$$\ln\frac{n_i}{n_{i+1}} \geq -\ln(1 - \frac{C_i}{\delta w(OPT)}) \geq \frac{C_i}{\delta w(OPT)}\ .$$

The last step uses the fact that $-\ln(1-x) \geq x$. We conclude that

$$C_i \leq \delta w(OPT)\ln\frac{n_i}{n_{i+1}}.$$

Now suppose we contracted the spider chosen in Step 2. Suppose the minimum ratio 3+ spider has t terminals. Since $t \geq 3$, we have $t - 1 \geq \frac{2t}{3}$. By the condition in Step 5, we have

$$\frac{3}{2}\gamma_{3+} \leq \delta \frac{w(OPT)}{n_i}$$

$$\frac{3}{2}\frac{C_i}{t} \leq \delta \frac{w(OPT)}{n_i}.$$

This gives us that $\frac{2t}{3} \geq \frac{n_i C_i}{\delta w(OPT)}$. Since $n_{i+1} = n_i - (t-1)$, we get

$$n_{i+1} \leq n_i - \frac{2t}{3} \leq n_i(1 - \frac{C_i}{\delta w(OPT)}).$$

$$\ln \frac{n_i}{n_{i+1}} \geq -\ln(1 - \frac{C_i}{\delta w(OPT)}) \geq \frac{C_i}{\delta w(OPT)} .$$

The last step uses the fact that $-\ln(1-x) \geq x$. We conclude that

$$C_i \leq \delta w(OPT) \ln \frac{n_i}{n_{i+1}}.$$

□

Lemma 3.7: *If the first two conditions in Step 5 are not met, and the best ratio bramble-spider is large ($> C$ triples) then $\gamma^{apx} \leq \delta(1+\epsilon)w$ where w is $w(OPT)/n_i$.*

Proof: Consider the min ratio bramble-spider with a large ($> C$) number of brambles of three terminals. Let w_3 (n_3') be the cost (number) of the brambles with three terminals, and w_2 (n_2') be the cost of the brambles with two terminals. Let w_1 be the cost of the single leg (if one exists), and c be the cost of center. Since the ratio cost is at most w, we have

$$w_3 + w_2 + w_1 + c \leq (3n_3' + 2n_2' + 1)w.$$

We can view the brambles with three terminals, as brambles with two terminals by dropping a terminal from consideration. Hence, if we can find the best ratio branch-spider, and if the ratio cost is at most δw, we get an approximation factor of $(\delta + \epsilon)\ln k$. (We need to elaborate on this – basically, this implies the existence of a large branch-spider with low ratio weight.)

Hence we may assume that

$$w_3 + w_2 + w_1 + c \geq \delta(2n_3' + 2n_2' + 1)w.$$

This gives us the following equation

$$(3 - 2\delta)n_3' \geq (\delta - 1)2n_2' + (\delta - 1).$$

Take three of the brambles with two terminals and make them into two brambles with three terminals each. Thus we increase the cost of pairs by $\frac{4}{3}$ because the cheapest pair is now repeated. We now greedily pick brambles with three terminals in each. We only pick $(n_3' + 2n_2'/3)$ terminals, because what we pick may lay waste to at most three good tridents. Our net cost is at most,

$$(w_3 + \frac{4w_2}{3})\frac{1}{3} + w_1 + c \leq \frac{1}{3}(w_3 + w_2 + w_1 + c) + \frac{1}{9}(w_2 + w_1 + c) + \frac{5}{9}(w_1 + c).$$

By the assumption that the first two conditions in Step 5 are not met, we have the following properties. The min ratio spider has ratio cost at least $\frac{\delta w}{2}$, hence $w_2 \geq n'_2 \delta w$. Since the min ratio 3+ spider has ratio cost at least $\frac{2}{3}\delta w$, hence $w_3 \geq 2n'_3\delta w$. We can bound the total cost as

$$\frac{1}{3}(3n'_3 + 2n'_2 + 1)w + \frac{1}{9}((3n'_3 + 2n'_2 + 1)w - 2n'_3\delta w) + \frac{5}{9}((3n'_3 + 2n'_2 + 1)w - 2n'_3\delta w - n'_2\delta w).$$

$$\leq w(3n'_3 + 2n'_2 - \frac{4\delta}{3}n'_3 - \frac{5\delta}{9}n'_2 + 1).$$

Now we remember the equation we first derived, multiply it by $\frac{w}{2}$ and add it to what we have. Total cost is at most

$$\leq w((\frac{9}{2} - \frac{7\delta}{3})(n'_3 + \frac{2}{3}n'_2) + \frac{3}{2} - \frac{\delta}{2}).$$

Since the number of terminals in this spider is at least $n'_3 + \frac{2}{3}n'_2$, dividing by the number of terminals yields a ratio cost of at most

$$w(\frac{9}{2} - \frac{7\delta}{3} + \epsilon\frac{3}{2} - \frac{\delta}{2}).$$

The last term is obtained by using the fact that $n'_3 > C = \frac{1}{\epsilon}$. Setting $\delta = 1.35$ yields a ratio cost of $w\delta(1+\epsilon)$. $\square$

Proof: (Of Theorem 3.5). If we contract the spider chosen in Step 1, or Step 2, we can use Lemma 3.6 to prove the claim. If we contract the spider chosen in Steps 3 or 4, we have to consider the following cases.

Case 1: suppose the best ratio bramble-spider is small ($\leq C$ brambles of size three)

(a) $\delta\gamma \leq \gamma^{apx}$. In this case $\gamma \leq w$. Since $\frac{C_i}{t} \leq \frac{w(OPT)}{n_i}$ and $t - 1 \geq \frac{3}{4}t$ (we contract at least 4 terminals), we get $C_i \leq \frac{4}{3}w(OPT)\ln\frac{n_i}{n_{i+1}}$.

(b) $\gamma^{apx} \leq \delta\gamma(\leq \delta w)$. Since in step 4 we find a spider with at least C terminals, we get a bound of $C_i \leq \frac{\delta}{1-\epsilon}w(OPT)\ln\frac{n_i}{n_{i+1}}$.

Case 2: suppose the best ratio bramble-spider is big ($> C$ brambles of size three)

(a) $\delta\gamma \leq \gamma^{apx}$ In this case, by Lemma 3.7, $\delta\gamma \leq \gamma^{apx} \leq \delta(1+\epsilon)w$ and we get $C_i \leq \frac{4}{3}(1+\epsilon)w(OPT)\ln\frac{n_i}{n_{i+1}}$.

(b) By Lemma 3.7 $\gamma^{apx} \leq \delta(1+\epsilon)w$. Since it has at least C terminals, we obtain $C_i \leq \delta\frac{1+\epsilon}{1-\epsilon}w(OPT)\ln\frac{n_i}{n_{i+1}}$.

$\square$

Theorem 3.8: *The node weighted Steiner tree problem can be approximated to a factor of* $1.35(1+\epsilon')\ln k$ *for any* $\epsilon' > 0$.

Proof: Summing up all the C_i values gives a total cost of $\delta(\frac{1+\epsilon}{1-\epsilon})w(OPT)\ln k$. For any given $\epsilon' > 0$, we set $\delta = 1.35$ and set $\epsilon = \frac{\epsilon'}{2+\epsilon'}$ so that $\frac{1+\epsilon}{1-\epsilon} = 1 + \epsilon'$ to obtain the required approximation factor. $\square$

4. 0-1 Proper functions

A considerable effort has been spent on solving a generalized network design problem where the connectivity requirements are specified as a function of a subset of vertices. These problems assume edge and node weights, and the objective function is to minimize the sum of the weights. Proper functions have several restrictions imposed on them. A 0-1 proper function is a general class of functions where the connectivity requirement is 0 or 1 across a cut. For details and definition of proper functions, the reader is referred to [6].

Klein and Ravi show that the node weighted Steiner tree algorithm can be modified to find a solution for the more general class of problems. A cut which is unsatisfied is called an active component. They show that active components behave as terminals in a node weighted Steiner tree formulation. However the final solution need not have a single connected component. Hence the decomposition lemmas developed for branch-spiders do not hold. It may be the case that the optimal solution has connected components having only two terminals in each. Typical problems include generalizations of Steiner trees.

We present a greedy algorithm for node weighted Steiner trees, which is practical and extends to the class of 0-1 proper functions as well. This algorithm has the same complexity as the original algorithm of Klein and Ravi [9]. For simplicity we describe it only for the node weighted Steiner tree problem.

Algorithm

Repeat the following steps until we have at most two terminals left, and then connect the terminals optimally.

Step 1. Find a spider with the minimum ratio. (this can be done by using the method by Klein and Ravi [9]). Let the ratio be γ_m. If it is a 3+ spider contract it.

Step 2. Else if the minimum ratio spider is a 2 spider, find the 3+ spider with the minimum ratio. Let this ratio be γ_{3+}.
For each terminal j find its closest terminal, with the distance being the sum of the weights of the nodes on the path. Call this path P_j. Order these P_j's in increasing order of cost. (We use P_j to denote both the path and its cost as convenient.)

Define $S = \{j | P_j \leq 2 \cdot \min\left[\frac{4\gamma_m}{3}, \gamma_{3+}\right]\}$. Let the number of distinct paths P_j such that $j \in S$ be ℓ_i. Denote the forest induced by these ℓ_i paths be T. Let $Cost(T)$ denote the cost of this forest.

Consider $\min\left[\frac{Cost(T)}{-\ln(1-\frac{\ell_i}{n_i})}, 2n_i\gamma_m, \frac{3}{2}n_i\gamma_{3+}\right]$.

Subcase (a). If minimum is achieved with the first term, contract the forest induced by the paths.

Subcase (b). If minimum is achieved with the second term, contract the minimum ratio spider.

Subcase (c). If minimum is achieved with the third term, contract the minimum ratio 3+ spider.

4.1. Proof of Approximation Factor

In this section we prove the following theorem.

Theorem 4.1: *The algorithm described above yields a node weighted Steiner tree of cost at most $1.6103 \ln k$ times the optimal, where k is the initial number of terminals.*

Denote the cost paid at iteration i to be C_i.

Lemma 4.2: *In iteration i, if only step 1 is executed, then $C_i \leq 1.5 w(OPT) \ln \frac{n_i}{n_{i+1}}$.*

This proof is the same as the proof of Theorem 3.4.

Lemma 4.3: *In iteration i, if step 2 is taken, then* $\min \left[\frac{Cost(T)}{-\ln(1-\frac{\ell_i}{n_i})},\ 2n_i\gamma_m,\ \frac{3}{2}n_i\gamma_{3+} \right] < 1.6103\, w(OPT)$.

We will prove this lemma a little later. We first see how to finish the proof using this lemma.

Lemma 4.4: *In iteration i, if step 2 is taken, then $C_i < 1.6103 w(OPT) \ln \frac{n_i}{n_{i+1}}$.*

Proof: If subcase (a) is chosen, then $C_i = Cost(T)$ and $n_{i+1} = n_i - \ell_i$. Since $1 - \frac{\ell_i}{n_i} = \frac{n_{i+1}}{n_i}$, and $\frac{Cost(T)}{w(OPT)} < 1.6103(-\ln(1 - \frac{\ell_i}{n_i}))$,

$$\frac{C_i}{w(OPT)} < 1.6103 \ \ln \frac{n_i}{n_{i+1}}$$

If subcase (b) is chosen then $C_i = 2\gamma_m$, and $n_{i+1} = n_i - 1$. From the lemma above, $2\gamma_m n_i < 1.6103\ w(OPT)$, we get $\frac{1.6103 w(OPT)}{n_i} > C_i$.

$$\ln \frac{n_i}{n_{i+1}} = -\ln(1 - \frac{1}{n_i}) > \frac{1}{n_i}.$$

Thus,

$$1.6103\ w(OPT) \ln \frac{n_i}{n_{i+1}} > C_i.$$

If subcase (c) is chosen, let the minimum ratio 3+ spider have t legs. Then $C_i = t\gamma_{3+}$, and $n_{i+1} = n_i - (t-1)$. By virtue of the above lemma,

$$\frac{3C_i n_i}{2t} < 1.6103\ w(OPT).$$

Now,

$$\ln \frac{n_i}{n_{i+1}} = \ln \frac{n_i}{n_i - t + 1} = -\ln(1 - \frac{t-1}{n_i}) \geq \frac{t-1}{n_i}.$$

Since $t \geq 3$, thus $(t-1) \geq \frac{2t}{3}$,

$$\ln \frac{n_i}{n_{i+1}} \geq \frac{t-1}{n_i} \geq \frac{2t}{3n_i} > \frac{C_i}{1.6103\ w(OPT)}.$$

This proves the lemma for all the cases. □

The proof of Theorem 4.1 now follows easily.

Proof: (Of Theorem 4.1) Let there be $z-1$ iterations. Then $n_z = 1$. Also if initially there were k terminals to connect, $n_1 = k$. For all iterations i, we have $1.6103 w(OPT) \ln \frac{n_i}{n_{i+1}} > C_i$. Summing over the iterations,

$$1.6103\ w(OPT) \sum_{i=1}^{z-1} \ln \frac{n_i}{n_{i+1}} = 1.6103\ w(OPT) \ln k > \sum_{i=1}^{z-1} C_i.$$

Since our solution has cost $\sum_{i=1}^{z-1} C_i$, the theorem follows. □

Proof: (Of Lemma 4.3) We can view each path P_j from terminal j to its closest terminal x, as a directed edge from j to x. By imposing a lexical ordering, we can ensure that the only directed cycles we get are 2-cycles. Note that the induced spanning forest has at most $|S|$ edges, and can be a lot smaller (as low as $\frac{|S|}{2}$, if the terminals form pairs). The collection of edges form a spanning forest of at least $|S|$ terminals.

Let $P^* = 2 \min \left[\frac{4\gamma_m}{3}, \gamma_{3+}\right]$. By the definition of Set S, for $j \in S$, $P_j \leq P^*$.

We wish to upper-bound the total cost of the forest produced. Let each vertex j in S have degree d_j in the forest. We charge the cost of the path to the two end vertices equally. Node j gets a charge of at most $(d_j - 1)\frac{P^*}{2} + \frac{P_j}{2}$. Summing over all the vertices the cost is at most

$$Cost(T) \leq \sum_{j \in S} (d_j - 1)\frac{P^*}{2} + \frac{P_j}{2} = (2l_i - |S|)\frac{P^*}{2} + \sum_{j \in S} \frac{P_j}{2} \ .$$

Using Lemma 2.1, OPT can be decomposed into spiders of total cost at most $w(OPT)$. By charging the terminals the spider ratio of the spider they belong to, we can obtain a lower-bound on $w(OPT)$. (This charging is the same as in Lemma 2.2.) Consider a terminal j. If j is in a 2 spider in the decomposition obtained by Lemma 2.1 then the spider ratio is at least $\frac{P_j}{2}$. Otherwise, j gets a charge of at least $\gamma_{3+} = \frac{P^*}{2}$. In any case, j gets a charge of at least $\min\left[\frac{P_j}{2}, \frac{P^*}{2}\right]$. If $j \notin S$, then $\min\left[\frac{P_j}{2}, \frac{P^*}{2}\right] = \frac{P^*}{2}$. Thus we get,

$$w(OPT) \geq (n_i - |S|)\frac{P^*}{2} + \sum_{j \in S} \frac{P_j}{2} \ .$$

Multiplying the above with -1 and adding to the equation before, we get (taking $-w(OPT)$ to RHS.)

$$Cost(T) \leq (2\ell_i - n_i)\frac{P^*}{2} + w(OPT) \ .$$

Setting $y = \frac{n_i P^*}{2w(OPT)}$, (note $\frac{P^*}{2} \leq \frac{4\gamma_m}{3} \leq \frac{4w(OPT)}{3n_i}$ and hence $y \leq \frac{4}{3}$),

$$\frac{Cost(T)}{-\ln(1 - \frac{\ell_i}{n_i})} \leq \frac{(2\frac{\ell_i}{n_i} - 1)y + 1}{-\ln(1 - \frac{\ell_i}{n_i})} w(OPT) \ .$$

From definition of y,

$$\frac{3y}{2} w(OPT) = \frac{3n_i}{2}\frac{P^*}{2} = \frac{3n_i}{2} \min\left[\frac{4\gamma_m}{3}, \gamma_{3+}\right] = \min\left[2n_i\gamma_m, \frac{3n_i}{2}\gamma_{3+}\right]$$

Fact: For all values of $0 < x \leq 1$ and $y \leq \frac{4}{3}$,

$$\min\left[\frac{(2x-1)y+1}{-\ln(1-x)}, \frac{3y}{2}\right] < 1.6103$$

Thus

$$\min\left[\frac{Cost(T)}{-\ln(1-\frac{\ell_i}{n_i})}, 2n_i\gamma_m, \frac{3n_i}{2}\gamma_{3+}\right] = \min\left[\frac{Cost(T)}{-\ln(1-\frac{\ell_i}{n_i})}, \min\left[2n_i\gamma_m, \frac{3n_i}{2}\gamma_{3+}\right]\right]$$

$$= \min\left[\frac{(2x-1)y+1}{-\ln(1-x)}w(OPT), \frac{3y}{2}w(OPT)\right] < 1.6103\, w(OPT)\ .$$

Which proves the lemma. □

5. Algorithm for Unweighted Case

For the case where each node has the same cost, and we are counting the number of nodes in our final solution, we can show that an approximation guarantee of $\ln k + \theta(1)$ is possible. We can also show that if the value of the optimal solution is at least a fixed constant then we can achieve an approximation factor of $\ln k$. The reason behind the improvement lies in the fact that we can provide better lower bounds on the optimal solution if we cannot find good ratio spiders.

In this section we present a simple algorithm with an approximation ratio of $\ln k$ when all the vertices have unit weight.

We have k required vertices (terminals) in a graph $G = (V, E)$, that we want to connect using the least number of non-terminals. We assume that the non-terminals have weight 1, and the terminals have weight 0.

We first note that connected components induced by terminals can always be shrunk to single terminals. Our algorithm runs in two phases. In the first phase, the algorithm greedily picks high degree stars (a star is a vertex that has at least two required vertices as neighbors) and merges them, until very few components are left. In the second phase, the algorithm runs a Steiner tree (edge) approximation algorithm with each edge having unit weight.

We pick $\lambda = 2c_s + 1$ where c_s is the best approximation ratio for the unweighted Steiner tree problem.

Algorithm A

Step 1. In each iteration choose a vertex that merges the largest number of required vertices until we reach a stage that the number of components left to merge is less than $\frac{\text{iteration count}}{\ln k - \lambda} + e^{\lambda}$ or no merging is possible.

Step 2. Apply an (edge weighted) Steiner tree approximation algorithm, with each edge having unit weight.

Theorem 5.1: *The above algorithm finds a solution to the unit node weighted Steiner tree (UNST) problem with an approximation factor of* $\ln k$ *(which is best possible), when the optimal solution is greater than* $c_s \cdot e^{\lambda}$.

Proof: Assume that the set of components remaining after the first phase is A'. We claim that there is a Steiner tree with at most $|A'| + |OPT|$ edges. Thus when we apply an (edge weighted) Steiner tree approximation, we get a tree with at most $c_s \cdot (|A'| + |OPT|)$ edges.

If the number of iterations in the first phase is r, the final solution has a cost $r + c_s \cdot (|A'| + |OPT|)$. We now proceed to give a bound on r.

Let a_i components be left after the i^{th} iteration. Since $|OPT|$ nodes are capable of merging these components, for each i, in the i^{th} iteration, there must be a node that merges $\left\lceil \frac{a_{i-1}}{|OPT|} \right\rceil$ components. This gives a bound on a_i,

$$a_i \leq a_{i-1} - \left\lceil \frac{a_{i-1}}{|OPT|} \right\rceil + 1 \leq a_{i-1}(1 - \frac{1}{|OPT|}) + 1.$$

We can easily verify that $a_i \leq a_0 \cdot (1 - \frac{1}{|OPT|})^i + \sum_{j=0}^{i-1}(1 - \frac{1}{|OPT|})^j$. The second term is a geometric series that sums to at most $|OPT|$. Thus when $i = (\ln k - \lambda) \cdot |OPT|$ the first term is at most e^{λ}, and the number of components $a_i \leq |OPT| + e^{\lambda} \leq \frac{i}{\ln k - \lambda} + e^{\lambda}$. This guarantees that the number of iterations, $r \leq (\ln k - \lambda) \cdot |OPT|$.

If we stop because no merging by stars is possible, then the components have disjoint neighborhoods, and OPT has to pick at least one vertex from each neighborhood. Thus $|A'| \leq |OPT|$. If we stop because the number of components is small, then $|A'| \leq |OPT| + e^{\lambda}$. In any case, $|A'| \leq |OPT| + e^{\lambda}$ and this yields a solution of cost at most $\ln k \cdot |OPT| + c_s \cdot e^{\lambda} + (2c_s - \lambda)|OPT|$. Putting $\lambda = 2c_s + 1$ gives at most $\ln k \cdot |OPT|$ vertices in our solution when $|OPT| \geq c_s \cdot e^{2c_s+1}$. □

The optimality of this approximation ratio was established by Berman (see [9]).

We can modify the above algorithm, to run until no further merging is possible.

Algorithm B

Step 1. In each iteration choose a vertex that merges the largest number of required vertices (at least two).

Step 2. Apply an (edge weighted) Steiner tree approximation algorithm, with each edge having unit weight.

Theorem 5.2: *The above algorithm finds a solution to the unit node weighted Steiner tree (UNST) problem with an approximation factor of* $\ln \Delta + 2c_s + 1$, *where* Δ *is the maximum degree.*

Proof: As before, let a_i denote the number of vertices left after the i^{th} iteration and $a_0 = k$. Then after $|OPT| \cdot \ln \frac{a_0}{|OPT|}$, there are at most $2 \cdot |OPT|$ components to connect. Hence we will continue to merge by stars for $|OPT|$ more iterations then the number of components will be definitely less than $|OPT|$.

Since each Steiner vertex can be adjacent to at most Δ required vertices, $|OPT| \geq \frac{a_0}{\Delta}$.

If at this stage we use a_f more iterations before invoking the edge weighted Steiner tree algorithm, there is a tree with $|OPT| - a_f + |OPT|$ edges. So we find a solution of cost at most $c_s \cdot (|OPT| - a_f + |OPT|)$. The final solution has at most $|OPT| \cdot \ln \frac{a_0}{|OPT|} + |OPT| + a_f + c_s \cdot (|OPT| - a_f + |OPT|)$ nodes. Since $|OPT| \geq \frac{a_0}{\Delta}$, we get a performance guarantee of $\ln \Delta + 2c_s + 1$ for the algorithm. □

6. Application to Connected Dominating Sets

The *connected dominating set* (CDS) problem is defined as follows: given a node weighted graph G, find a subset S of vertices of minimum weight, such that S induces a connected subgraph and the vertices in S form a dominating set in G. See [7] for applications and other results for the CDS problem on unweighted graphs.

We can develop a similar approximation scheme for connected dominating sets. As the algorithm proceeds, certain vertices are added to the current CDS. Initially, the CDS is empty, and finally it forms a valid CDS when the algorithm terminates. We use the following color notation – each vertex in the CDS is black. All vertices which are adjacent to a black vertex are colored gray. The remaining vertices are colored white.

Define a *piece* as a black connected component or a white vertex. Treat each piece as a terminal. Define spiders as in the previous section, but include only the weight of non-leaf gray and white nodes when computing the weight of a spider. In this algorithm, we only shrink black connected components, and not complete spiders.

It is easy to observe that a spider connecting ℓ pieces, reduces the number of pieces by $\ell - 1$. And ultimately when we have found a solution only one piece should be remaining. Notice that all the decomposition claims in the previous section follow. We can proceed analyzing in a similar way to achieve an approximation ratio of $(1.35 + \epsilon) \ln k$.

References

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

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

[3] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, The MIT Press, 1989.

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

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

[6] M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems", *SIAM Journal on Computing*, 24:296–317, (1995).

[7] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets", *Proc. of 4th Annual European Symposium on Algorithms*, pages 179–193, (1996). To appear in *Algorithmica*.

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

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

[10] M. Karpinsky and A. Zelikovsky, "New approximation algorithms for the Steiner tree problem", *Journal of Combinatorial Optimization*, 1(1):47–66, (1997).

[11] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems", *Journal of the ACM*, 41(5): 960–981, (1994).

[12] V. J. Rayward-Smith, "The computation of nearly minimal Steiner trees in graphs", *Internat. J. Math. Educ. Sci. Tech.*, 14: 15–23, (1983).

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

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