Improved Steiner Tree Approximation in Graphs*

Gabriel Robins† and Alexander Zelikovsky‡

†Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442
robins@cs.virginia.edu, www.cs.virginia.edu/robins

‡Department of Computer Science, Georgia State University, Atlanta, GA 30303
alexz@cs.gsu.edu, www.cs.gsu.edu/~matazz

Abstract

The Steiner tree problem in weighted graphs seeks a minimum weight connected subgraph containing a given subset of the vertices (terminals). We present a new polynomial-time heuristic with an approximation ratio approaching $1 + \frac{\ln 3}{2} \approx 1.55$, which improves upon the previously best-known approximation algorithm of [10] with performance ratio ≈ 1.59. In quasi-bipartite graphs (i.e., in graphs where all non-terminals are pairwise disjoint), our algorithm achieves an approximation ratio of ≈ 1.28, whereas the previously best method achieves an approximation ratio approaching 1.5 [19]. For complete graphs with edge weights 1 and 2, we show that our heuristic has an approximation ratio approaching ≈ 1.28, which improves upon the previously best-known ratio of $\frac{4}{3}$ [4]. Our method is considerably simpler and easier to implement than previous approaches. Our techniques can also be used to prove that the Iterated 1-Steiner heuristic [14] achieves an approximation ratio of 1.5 in quasi-bipartite graphs, thus providing the first known non-trivial performance ratio of this well-known method.

1 Introduction

Given an arbitrary weighted graph with a distinguished vertex subset, the *Steiner Tree Problem* asks for a minimum-cost subtree spanning the distinguished vertices. Steiner trees are important in various applications such as VLSI routing [14], wirelength estimation [6], phylogenetic tree reconstruction in biology [11], and network routing [12]. The Steiner Tree Problem is NP-hard even in the Euclidean or rectilinear metrics [8].

Arora established that Euclidean and rectilinear minimum-cost Steiner trees can be efficiently approximated arbitrarily close to optimal [1]. On the other hand, unless $P = NP$, the Steiner Tree Problem in general graphs cannot be approximated within a factor of $1 + \epsilon$ for sufficiently small $\epsilon > 0$ [4, 7]. For arbitrary weighted graphs, the best Steiner approximation ratio achievable within polynomial time was gradually decreased from 2 to 1.59 in a series of works [20, 21, 2, 22, 18, 15, 10].

In this paper we present a polynomial-time approximation scheme with a performance ratio approaching $1 + \frac{\ln 3}{2} \approx 1.55$ which improves upon the previously best-known ratio of 1.59 due to

*This work was supported by a Packard Foundation Fellowship, by National Science Foundation Young Investigator Award (MIP-9457412), and by a GSU Research Initiation Grant.

Hougardy and Prömel [10]. We apply our heuristic to the Steiner Tree Problem in quasi-bipartite graphs (i.e., where all non-terminals are pairwise disjoint). In quasi-bipartite graphs our heuristic achieves an approximation ratio of ≈ 1.28 within time $O(mn^2)$, where m and n are the numbers of terminals and non-terminals in the graph, respectively. This is an improvement over the primal-dual algorithm by Rajagopalan and Vazirani [19] where the bound is more than 1.5. We also show that a well-known Iterated 1-Steiner heuristic [13, 9, 14] achieves an approximation ratio of 1.5 for quasi-bipartite graphs; previously, no non-trivial bounds were known for the Iterated 1-Steiner heuristic. Finally, we improve the approximation ratio achievable for the Steiner Tree Problem in complete graphs with edge weights 1 and 2, by decreasing it from the previously known $\frac{4}{3}$ [4] to less than 1.28 for our algorithm.

The remainder of the paper is organized as follows. In the next section we introduce basic definitions, notation and properties. In Section 3 we present our main algorithm (called k-LCA) and formulate the basic approximation result. In Sections 4 and 5 we prove the approximation ratio of the algorithm k-LCA in general graphs and estimate the performance of the Iterated 1-Steiner heuristic and k-LCA in quasi-bipartite graphs and in complete graphs with weights 1 and 2. We conclude by proving in Section 6 the basic approximation result for k-LCA.

2 Definitions, Notations and Basic Properties

Let $G = (V, E, cost)$ be a graph with a non-negative cost function on its edges. Any tree in G spanning a given set of *terminals* $S \subseteq V$ is called a *Steiner tree*, and the cost of a tree is defined to be the sum of its edge costs. The *Steiner Tree Problem* (STP) seeks a minimum-cost Steiner tree. Note that a Steiner tree may contain non-terminal vertices and these are referred to as *Steiner points*. We can assume that the cost function over $G = (V, E, cost)$ is metric (i.e., the triangle inequality holds) since we can replace any edge $e \in E$ with the shortest path connecting the ends of e. Henceforth we will therefore assume that G is a complete graph. Similarly, for the subgraph G_S induced by the terminal set S, G_S is a complete graph with vertex set S.

Let $MST(G_S)$ be the minimum spanning tree of G_S. For any graph H, $cost(H)$ is the sum of the costs of all edges in H. We thus denote the cost of a minimum spanning tree of H by $mst(H)$, e.g., $cost(MST(G_S)) = mst(G_S)$. For brevity, we use mst to denote $mst(G_S)$. In order to simplify our analyses, we further assume that all edge costs in G are unique (this ensures that the optimal Steiner tree and minimum spanning tree are unique).

A Steiner tree over a subset of the terminals $S' \subset S$ in which all terminals S' are leaves is called a *full component* (see Figure 1(a)). Any Steiner tree can be decomposed into full components by splitting all the non-leaf terminals. Our algorithm will proceed by adding full components to a growing solution, based on their "relative cost savings" (this notion will be made precise below). We assume that any full component has its own copy of each Steiner point so that full components chosen by our algorithm do not share Steiner points.

A Steiner tree which does not contain any Steiner points (i.e., where each full component consists of a single edge), will be henceforth called a *terminal-spanning tree*. Our algorithm will compute relative cost savings with respect to the "shrinking" terminal-spanning tree which initially coincides with $MST(G_S)$.

The relative cost savings of full components are represented by a ratio of how much a full component decreases the cost of the current terminal-spanning tree over the cost of connecting its Steiner points to terminals. The cost savings of an arbitrary graph H with respect to a terminal-spanning tree T is the difference between the cost of T and the cost of the Steiner tree

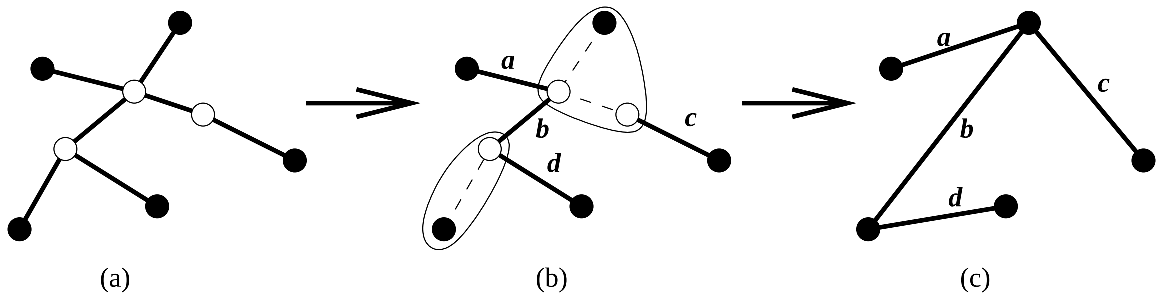

Figure 1: (a) A full component K: filled circles denote terminals and empty circles denote Steiner points. (b) Connected components of $Loss(K)$ to be collapsed, dashed edges belong to $Loss(K)$. (c) The corresponding terminal-spanning tree $C[K]$ with the contracted $Loss(K)$.

obtained by augmenting H with the edges of T. Formally, let $T[H]$ be the minimum cost graph in $H \cup T$ which contains H and spans all the terminals of S (see Figure 2). The *gain* of H with respect to T is defined as $gain_T(H) = cost(T) - cost(T[H])$. If H is a Steiner tree, then $gain_T(H) = cost(T) - cost(H)$. Note that $gain_T(H) \leq cost(T) - mst(T \cup H)$ because $T[H]$ cannot cost less than $MST(T \cup H)$. We will use the following property of *gain* proved in [21, 2].

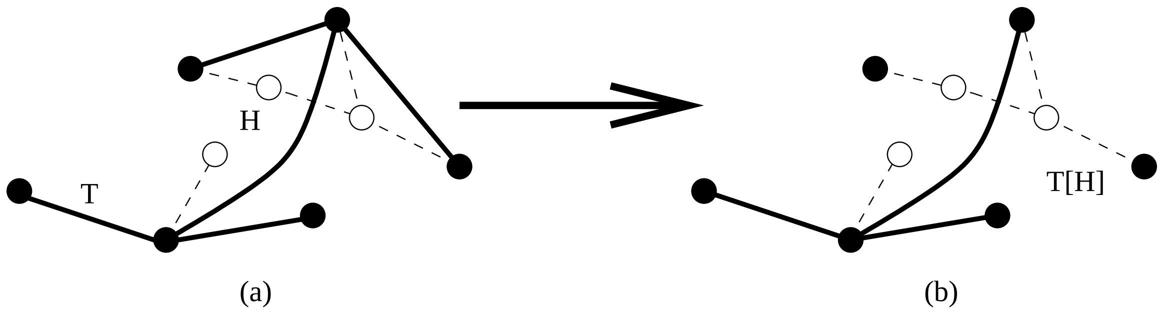

Figure 2: (a) A graph H (dashed edges) and a terminal-spanning tree T (solid edges). (b) The corresponding graph $T[H]$ contains H and spans all the terminals.

Lemma 1 *For any terminal-spanning tree T and graphs H and H',*

$$gain_T(H \cup H') \leq gain_T(H) + gain_T(H')$$

The minimum-cost connection of Steiner points of a full component K to its terminals is denoted $Loss(K)$. Formally, $Loss(K)$ is a minimum-cost forest spanning the Steiner nodes of a full component K such that each connected component contains at least one terminal (see Figure 1(b)). Intuitively, *Loss* will serve as an upper bound on the optimal solution cost increase during our algorithm's execution (as will be elaborated below). We will denote the cost of $Loss(K)$ by $loss(K)$. The loss of a union of full components is the sum of their individual losses.

As soon as our algorithm accepts a full component K it *contracts* its $Loss(K)$, i.e. "collapses" each connected component of *Loss* into a single node (see Figure 1(c)). Formally, a *loss-contracted* full component $C[K]$ is a terminal-spanning tree over terminals of K in which two terminals are connected if there is an edge between the corresponding two connected components in the forest $Loss(K)$. The cost of any edge in $C[K]$ coincides with the cost of the corresponding edge in K. The 1-to-1 correspondence between edges of $K \setminus Loss(K)$ and $C[K]$ implies that $cost(H) - loss(H) = cost(C[H])$. Similarly, for any Steiner tree H, $C[H]$ is the terminal-spanning tree in which the losses of all full components of H are contracted.

Our algorithm constructs a *k-restricted* Steiner tree, i.e., a Steiner tree in which each full component has at most k terminals. Let Opt_k be an optimal k-restricted Steiner tree, and let opt_k and $loss_k$ be the cost and loss of Opt_k, respectively. Let opt and $loss$ be the cost and loss of the optimal Steiner tree, respectively.

We now prove the following lower bound on the cost of the optimal k-restricted Steiner tree.

Lemma 2 *Let H be a Steiner tree; if $gain_{C[H]}(K) \leq 0$ for any k-restricted full component K, then*

$$cost(H) - loss(H) = cost(C[H]) \leq opt_k$$

Proof. Let $K_1, \ldots, K_p$ be full components of Opt_k.

$$\begin{aligned} cost(C[H]) - opt_k &= gain_{C[H]}(Opt_k) \\ &= gain_{C[H]}(K_1 \cup \ldots \cup K_p) \\ &\leq gain_{C[H]}(K_1) + \ldots + gain_{C[H]}(K_p) \\ &\leq 0 \end{aligned}$$

□

An *approximation ratio* of an algorithm is an upper bound on the ratio of the cost of the found solution over the cost of the optimal solution. In the next section we will propose a new algorithm for the Steiner Tree Problem, and then prove a (best-to-date) approximation ratio for it.

3 The Algorithm

All previous heuristics (except Berman and Ramayer's [2] approach) with provably good approximation ratios choose appropriate full components and then contract them in order to keep them for the overall solution. This does not allow us to give up an already-accepted full component even if later we would find out that a better full component *disagrees* with a previously accepted (two components disagree if they share at least two terminals).

The main idea behind the Loss-Contracting Algorithm (see Figure 3) is to contract as little as possible so that (i) a chosen full component may still participate into the overall solution but (ii) not many other full components would be rejected. In particular, if we contract $Loss(K)$, i.e., replace a full component K with $C[K]$, then (i) it will not cost anything to add a full component K in the overall solution and (ii) we decrease the gain of full components which disagree with K by a small value (e.g., less than in the Berman-Ramayer algorithm for large k, and much smaller than in [15] for any k).

Our algorithm iteratively modifies a terminal-spanning tree T, which is originally $MST(G_S)$, by incorporating into T loss-contracted full components greedily chosen from G. The intuition behind the gain-over-loss objective ratio is as follows. The cost of the approximate solution lies between $mst = mst(G_S)$ and opt_k. If we accept a component K, then it increases (by a gain of K) the gap between mst and the cost of the approximation. Thus the gain of K is our clear profit. On the other hand, if K does not belong to OPT_k, then after accepting K we would no longer be able to reach Opt_k because we would need to pay for the connection of incorrectly chosen Steiner points. Therefore, the value of $loss(K)$, which is the connection cost of Steiner points of K to terminals, is an upper bound on the increase in the cost gap between opt_k and the best achievable solution after accepting K. Thus $loss(K)$ is an estimate of our connection expense. Finally, maximizing the ratio of gain over loss is equivalent to maximizing the profit per unit expense.

Loss-Contracting Algorithm (k-LCA) for Steiner Trees in Graphs

Input: A complete graph $G = (V, E, cost)$ with edge costs satisfying the triangle inequality, a set of terminals $S \subset V$, and an integer $k \leq |S|$
Output: A k-restricted Steiner tree in G connecting all the terminals in S

$T = MST(G_S)$
$H = G_S$

Repeat forever
  Find a k-restricted full component K with the maximum $r = gain_T(K)/loss(K)$
  If $r \leq 0$ **then exit repeat**
  $H = H \cup K$
  $T = MST(T \cup C[K])$

Output the tree $MST(H)$

Figure 3: The k-restricted Loss-Contracting Algorithm (k-LCA).

In Section 6 we will show that $cost(T) - mst(T \cup K) = gain_T(K)$. Therefore, each time the algorithm chooses a full component K, the cost of T decreases by $gain_T(K) + loss(K)$. This will imply the basic approximation result proved in Section 6.

Theorem 1 *For any instance of the Steiner Tree Problem, the cost Approx of the Steiner tree produced by algorithm k-LCA is at most:*

$$Approx \leq loss_k \cdot \ln\left(1 + \frac{mst - opt_k}{loss_k}\right) + opt_k \tag{1}$$

4 Performance of k-LCA in General Graphs

Our estimate of the performance ratio of algorithm k-LCA in arbitrary graphs is based on the estimates of optimal k-restricted Steiner trees. Let ρ_k be the worst-case ratio of $\frac{opt_k}{opt}$. It was shown in [5] that $\rho_k \leq 1 + (\lfloor \log_2 k \rfloor + 1)^{-1}$. We will show below that the approximation ratio of k-LCA is at most $\rho_k(1 + \frac{1}{2}\ln(\frac{4}{\rho_k} - 1))$. Therefore, the approximation ratio of k-LCA converges to $1 + \frac{\ln 3}{2} < 1.55$ when $k \to \infty$. This is an improvement over the algorithm given by Hougrady and Prommel [10], where the approximation ratio approaches 1.59.

Theorem 2 *The k-LCA algorithm has an approximation ratio of at most* $(1+\frac{1}{2}\ln(\frac{4}{\rho_k}-1))\rho_k$

Proof. Since $mst \le 2opt$ (see [20]), the inequality (1) yields the following upper bound on the output tree cost of k-LCA.

$$Approx \le loss_k \cdot \ln\left(1+\frac{2opt-opt_k}{loss_k}\right)+opt_k$$

It was proved in [15] that for any Steiner tree T, $loss(T) \le \frac{1}{2}cost(T)$. Therefore, $loss_k \le \frac{1}{2}opt_k$.

The partial derivative $(loss_k \cdot \ln(1+\frac{2opt-opt_k}{loss_k}))'_{loss_k}$ is always positive; therefore, the the upper bound on *Approx* achieves maximum when $loss_k = \frac{1}{2}opt_k$. Thus, we obtain

$$\frac{Approx}{opt} \le \frac{opt_k}{opt}\cdot\left(1+\frac{\ln(\frac{4opt}{opt_k}-1)}{2}\right)$$

Since the upper bound above grows when opt_k increases, we can replace $\frac{opt_k}{opt}$ with the maximum value of ρ_k. □

5 Steiner Trees in Quasi-Bipartite Graphs and Complete Graphs with Edge Weights 1 and 2

Recently Rajagopalan and Vazirani [19] suggested a primal-dual -based algorithm for approximating Steiner trees. They show that their algorithm has an approximation ratio of $1.5+\epsilon$ for quasi-bipartite graphs, i.e., the graphs where all non-terminals are pairwise disjoint. We first show that the well-known Iterated 1-Steiner heuristic [13, 9, 14] has an approximation ratio of 1.5. Next, we apply algorithm k-LCA to quasi-bipartite graphs and estimate its runtime. Finally we prove that the performance ratio of k-LCA for quasi-bipartite graphs is below 1.28. We also apply k-LCA to the Steiner Tree Problem in complete graphs with edge weights 1 and 2. Bern and Plassmann [4] proved that this problem is MAX SNP-hard and gave a $\frac{4}{3}$-approximation algorithm. Applying Lovasz's algorithm for the parity matroid problem (see [16]), an 1.2875-approximation algorithm was given in [3]. We will show that the performance ratio of algorithm k-LCA approaches 1.28 for such graphs, improving on previously achievable bounds.

The Iterated 1-Steiner heuristic. The Iterated 1-Steiner heuristic (I1S) (see [13, 9, 14]) repeatedly (while it is possible) adds Steiner points to the terminal set, which decreases the cost of the minimum spanning tree over terminals. Accepted Steiner nodes are deleted if they become useless, i.e., if their degree become 1 or 2 in the MST over the terminals. Although I1S decreases the MST cost by the maximum possible value at each iteration, we will estimate the cost of the output Steiner tree regardless of how it was obtained. The following theorem will also enable us to estimate the performance ratio of a faster *Batched* variant of the Iterated 1-Steiner heuristic [13, 9, 14].

Theorem 3 *Given an instance of the Steiner Tree Problem in a quasi-bipartite graph G, let H be a Steiner tree in G such that (i) any Steiner point has degree at least 3 and (ii) H cannot be improved by adding any other Steiner point, i.e., $mst(H\cup v)\ge cost(H)$ for any vertex v in G. Then the cost of H is at most 1.5 times the optimal.*

Proof. Any full component in quasi-bipartite graphs has a single Steiner point. Therefore, the loss of any full component equals the cost of the least-cost edge connecting its single Steiner point to a terminal. Since any Steiner point has degree at least 3 (condition (i)), the loss of any full component in H is at most one third of its cost. Thus, $loss(H) \le \frac{1}{3}cost(H)$.

We now show that $gain_{C[H]}(K) \le 0$ for any full component K. Indeed, condition (ii) implies that $mst(H \cup K) \ge cost(H)$. If we contract the loss of H, then we can decrease $MST(H \cup K)$ by at most $loss(H)$ since reduction by $loss(H)$ happens only if all edges of $Loss(H)$ belong to $MST(H \cup K)$. Therefore, $mst(C[H] \cup K) \ge mst(H \cup K) - loss(H)$ and $mst(C[H] \cup K) \ge cost(H) - loss(H) = cost(C[H])$. Thus, $gain_{C[H]}(K) \le cost(C[H]) - mst(C[H] \cup K) \le 0$. By Lemma 2, $cost(H) - loss(H) \le opt$ and since $loss(H) \le \frac{1}{3}cost(H)$, we obtain $cost(H) \le \frac{3}{2}opt$. □

The above result helps explain why the Iterated 1-Steiner and Rajagopalan-Vazirani heuristics perform similarly when applied to instances of the Steiner Tree Problem restricted to the rectilinear plane (see [17]).

Runtime of the algorithm k-LCA in quasi-bipartite graphs. For a given Steiner point v, algorithm k-LCA adds only a full component with the largest gain since the loss is determined by v. We can find a full tree with the maximum gain with respect to a terminal-spanning tree T among *all* possible full components with Steiner point v by merely finding all neighbors of v in $MST(T \cup v)$. Therefore a full component maximizing the gain-over-loss ratio over all k can be found within polynomial time.

We estimate the runtime of k-LCA for quasi-bipartite graphs as follows. Let m and n be the number of terminals and non-terminals, respectively. The number of iterations is $O(n)$ since a Steiner point can be added only once to H. Each iteration consists of $O(n)$ gain evaluations, each of which can be computed within $O(m)$ time. Finally, using the appropriate data structures, the k-LCA algorithm can be implemented within a total runtime of $O(n^2m)$, where m is the number of terminals.

Performance of the algorithm k-LCA. We first estimate the loss of a Steiner tree in the cases of quasi-bipartite graphs and complete graphs with edge weights 1 and 2.

Lemma 3 *For the Steiner Tree Problem in quasi-bipartite graphs and complete graphs with edge weights 1 and 2,*

$$mst \le 2(opt_k - loss_k) \tag{2}$$

Proof. For quasi-bipartite graphs, let K be an arbitrary full component of a Steiner tree T with p terminals connected by a single Steiner point with edges of lengths $d_0, d_1, \ldots, d_{p-1}$. Assume that $loss(K) = d_0 = min\{d_i\}$. Let $mst(K)$ be the cost of the minimum spanning tree of $G_{S'}$, where S' is the set of terminals in K. By the triangle inequality,

$$mst(K) \le \sum_{i=1}^{p-1}(d_0 + d_i) = p \cdot d_0 + cost(K) - 2d_0 \le 2cost(K) - 2loss(K)$$

The bound (2) follows from the fact that mst, the minimum spanning tree cost of S, does not exceed the sum of mst-costs for terminals in each of the full components in Opt_k.

Now we prove the lemma for the case of complete graphs with edge weights 1 and 2. Let m and n respectively be the number of terminals and Steiner points in the optimal k-restricted Steiner tree Opt_k. Then $mst \le 2m - 2$ since all edge weights are at most 2 and $opt_k \ge m + n - 1$ since

Opt_k contains $m+n$ nodes. We may assume that full components of Opt_k contain only edges of weight 1, and therefore $loss_k = n$. Thus, $mst \le 2m-2 = 2(m+n-1-n) \le 2(opt_k - loss_k)$. $\square$

Theorem 4 *Algorithm k-LCA has an approximation ratio of at most* ≈ 1.279 *for quasi-bipartite graphs and an approximation ratio approaching* ≈ 1.279 *for complete graphs with edge weights 1 and 2.*

Proof. After substituting the MST bound (2) into inequality (1), we obtain

$$Approx \le loss_k \cdot \ln\left(\frac{opt_k}{loss_k} - 1\right) + opt_k \tag{3}$$

Taking the partial derivative of $(loss \cdot \ln(\frac{opt_k}{loss_k} - 1))'_{loss_k}$, we see that the single maximum of the upper bound (3) occurs when $x = \frac{loss_k}{opt_k - loss_k}$ is the root of the equation $1 + \ln x + x = 0$. Solving this equation numerically we obtain $x \approx 0.279$. Finally, we substitute x into (3)

$$Approx \le \frac{x}{1+x} \cdot opt_k \cdot \ln\frac{1}{x} + opt_k = (x+1) \cdot opt_k \approx 1.279 \cdot opt_k$$

The bound above is valid for the output of algorithm k-LCA for quasi-bipartite graphs if we set $k = |S|$, i.e., if we omit the index k. For complete graphs with edge weights 1 and 2, opt_k converges to opt, and the approximation ratio of algorithm k-LCA therefore converges to 1.279 when $k \to \infty$. $\square$

6 Approximation Ratio of Algorithm k-LCA

This section is devoted to the proof of the Theorem 1. Let $K_1, \ldots, K_{last}$ be full components chosen by k-LCA. Let $T_0 = MST(G_S)$ and let T_i, $i = 1, \ldots, last$ be the tree T produced by k-LCA after i iterations. Let $cost(T_i)$ be the cost of T_i after the i-th iteration of k-LCA.

Lemma 4 $gain_{T_{i-1}}(K_i) = cost(T_{i-1}) - mst(T_{i-1} \cup K_i)$

Proof. It is sufficient to show that $T_{i-1}[K_i] = MST(T_{i-1} \cup K_i)$. Assume that $MST(T_{i-1} \cup K_i)$ does not contain some edge $e \in K_i$ and let A and B be two connected components of $K_i - \{e\}$. We will show that either A or B has a larger gain-over-loss ratio, which contradicts the choice of K_i.

Since e does not belong to $MST(T_{i-1} \cup K_i)$, we have $cost(T_{i-1}[A \cup B]) < cost(T_{i-1}[K_i])$. By Lemma 1, $gain_{T_{i-1}}(K_i) < gain_{T_{i-1}}(A \cup B) \le gain_{T_{i-1}}(A) + gain_{T_{i-1}}(B)$. Note that e is the longest edge on a K_i-path between some pair of terminals, and therefore cannot belong to $Loss(K_i)$. Thus $Loss(K_i) = Loss(A) \cup Loss(B)$ and $loss(K_i) = loss(A) + loss(B)$. Finally,

$$\frac{gain_{T_{i-1}}(K_i)}{loss(K_i)} < \frac{gain_{T_{i-1}}(A) + gain_{T_{i-1}}(B)}{loss(A) + loss(B)} \le \max\left\{\frac{gain_{T_{i-1}}(A)}{loss(A)}, \frac{gain_{T_{i-1}}(B)}{loss(B)}\right\}$$

$\square$

We define the *supergain* of a graph H with respect to a Steiner tree T as $supergain_T(H) = gain_T(H) + loss(H)$. By Lemma 4, the supergain of K_i with respect to T_{i-1} is

$$\begin{aligned} supergain_{T_{i-1}}(K_i) &= gain_{T_{i-1}}(K_i) + loss(K_i) \\ &= cost(T_{i-1}) - mst(T_{i-1} \cup K_i) + mst(T_{i-1} \cup K_i) - cost(T_i) \\ &= cost(T_{i-1}) - cost(T_i) \end{aligned} \tag{4}$$

Let $G_i = supergain_{T_i}(OPT_k)$ be the supergain of the optimal k-restricted Steiner tree OPT_k with respect to T_i $i = 0, 1, \ldots, last$. Let $loss(n)$ be the loss of the first n accepted full trees $K_1, \ldots, K_n$. We will show that the loss of full components accepted by k-LCA does not grow too fast.

Lemma 5 *If G_n is positive, then* $\frac{loss(n)}{loss_k} \leq \ln \frac{G_0}{G_n}$

Proof. Let $l_i = loss(K_i)$ and $g_i = supergain_{T_{i-1}}(K_i)$ be respectively the loss and supergain of the i-th full Steiner tree accepted by algorithm k-LCA. Let Opt_k consist of full components X_j. By Lemma 1,

$$\frac{G_0}{loss_k} \leq \frac{\sum_{X_j \in Opt_k} supergain_{T_0}(X_j)}{\sum_{X_j \in Opt_k} loss(X_j)} \leq 1 + max_{X_j \in Opt_k} \left\{ \frac{gain_{T_0}(X_j)}{loss(X_j)} \right\} \leq 1 + \frac{gain_{T_0}(K_1)}{loss(K_1)} = \frac{g_1}{l_1}$$

Inductively, for $i = 1, 2, \ldots, n$, $\frac{G_{i-1}}{loss_k} \leq \frac{g_i}{l_i}$. Therefore,

$$g_i \geq \frac{l_i}{loss_k} G_{i-1} \tag{5}$$

Each time k-LCA accepts a full tree K_i, it decreases the cost of T_i by the supergain of K_i, which results in decrease of the supergain of Opt_k by the same value. The equality (4) yields $G_i = cost(T_i) - cost(OPT_k) + loss_k$. Therefore, $G_{i-1} - G_i = cost(T_{i-1}) - cost(T_i) = g_i$.

The inequality (5) implies that $G_i = G_{i-1} - g_i \leq G_{i-1}\left(1 - \frac{l_i}{loss_k}\right)$. Since $G_n > 0$, unraveling the last inequality yields

$$\frac{G_n}{G_0} \leq \prod_{i=1}^{n} \left(1 - \frac{l_i}{loss_k}\right)$$

Taking the natural logarithms of both sides and using inequality $x \geq \ln(1 + x)$ we finally obtain

$$\ln \frac{G_0}{G_n} \geq \sum_{i=1}^{n} \frac{l_i}{loss_k} = \frac{loss(n)}{loss_k} \tag{6}$$

□

By Lemma 2, after all iterations terminate, the cost of the last tree T_{last} will be at most opt_k. We stop iterating when $cost(T_{n+1}) < opt_k \leq cost(T_n)$ for some n. It can be show that we can "partially" perform the $(n + 1)$-st iteration so that $cost(T_{n+1})$ will *coincide* with opt_k. Then $G_0 = mst - opt_k + loss_k$ and $G_{n+1} = opt_k - opt_k + loss_k = loss_k$. Finally,

$$Approx \leq cost(T_{n+1}) + loss(n + 1) \leq opt_k + opt_k \cdot \ln \frac{mst - opt_k + loss_k}{loss_k}$$

Acknowledgments

We thank Gruia Calinescu for reading earlier drafts of this paper and giving numerous helpful suggestions.

References

[1] S. Arora, "Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems", *Proceedings 37th Annual Symposium on Foundations of Computer Science* (1996), 2–11.

[2] P. Berman and V. Ramaiyer, "Improved Approximations for the Steiner Tree Problem", *J. of Algorithms*, 17 (1994), 381–408.

[3] P. Berman, M. Furer and A. Zelikovsky, "Applications of the Matroid Parity Problem to Approximating Steiner Trees", *Tech. Rep. 980021, Computer Science Dept., UCLA*, Los Angeles, 1998.

[4] M. Bern and P. Plassmann, "The Steiner Tree Problem with Edge Lengths 1 and 2", *Information Processing letters* 32 (1989), 171–176.

[5] A. Borchers and D.-Z. Du, "The k-Steiner Ratio in Graphs", *SIAM J. Computing* 26 (1997), 857–869.

[6] A. Caldwell, A. Kahng, S. Mantik, I. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-Based Placement", *Proceedings of the International Symposium on Physical Design, Monterey, California* (1998), pp. 4–11.

[7] A. E. F. Clementi and L. Trevisan, "Improved Non-Approximability Results for Minimum Vertex Cover with Density Constraints", *Electronic Colloquium on Computational Complexity*, TR96-016 (1996).

[8] M. R. Garey, D. S. Johnson. "The Rectilinear Steiner Problem is NP-Complete", SIAM J. Appl. Math., 32, 826-834, 1977.

[9] J. Griffith, G. Robins, J. S. Salowe, and T. Zhang, *Closing the Gap: Near-Optimal Steiner Trees in Polynomial Time*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 13 (1994), pp. 1351–1365.

[10] S. Hougardy and H. J. Prömmel, "A 1.598 Approximation Algorithm for the Steiner Problem in Graphs", *Proceedings of ACM-SIAM Symposium on Discrete Algorithms* (1999), 448–453.

[11] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*, North-Holland, 1992.

[12] B. Korte, H. J. Prömel, A. Steger. "Steiner Trees in VLSI-Layouts", In Korte et al.: Paths, Flows and VLSI-Layout, Springer, 1990.

[13] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics With Good Performance", *IEEE Transactions on Computer-Aided Design*, 11 (7), 1992, pp. 893-902.

[14] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Publishers, 1995.

[15] M. Karpinski and A. Zelikovsky, "New Approximation Algorithms for the Steiner Tree Problem", *Journal of Combinatorial Optimization*, 1 (1997), 47–65.

[16] L. Lovasz and M. D. Plummer, *Matching Theory.* Elsevier Science, Amsterdam, 1986.

[17] I. I. Mandoiu, V. V. Vazirani and J. L. Ganley, "A New Heuristic for Rectilinear Steiner Trees", *manuscript.*

[18] H. J. Prömmel and A. Steger, "RNC-Approximation Algorithms for the Steiner Problem", *Proceedings 14th Annual Symposium on Theoretical Aspects of Computer Science* (1997), 559–570.

[19] S. Rajagopalan and V. V. Vazirani, "On the Bidirected Cut Relaxation for Metric Steiner Problem", *Proceedings of ACM-SIAM Symposium on Discrete Algorithms* (1999), 742–757.

[20] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs", *Math. Jap. 24 (1980)*, 573–577.

[21] A. Zelikovsky, "An 11/6-Approximation Algorithm for the Network Steiner Problem", *Algorithmica 9 (1993)*, 463–470.

[22] A. Zelikovsky, "Better Approximation Bounds for the Network and Euclidean Steiner Tree Problems", *Technical report CS-96-06, University of Virginia, 1996.*