

# Tighter Bounds for Graph Steiner Tree Approximation\*

Gabriel Robins<sup>†</sup> and Alex Zelikovsky<sup>‡</sup>

<sup>†</sup>Department of Computer Science, University of Virginia, Charlottesville, VA 22904  
robins@cs.virginia.edu, www.cs.virginia.edu/robins

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

## Abstract

The classical 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 that achieves a best-known approximation ratio of  $1 + \frac{\ln 3}{2} \approx 1.55$  for general graphs, and best-known approximation ratios of  $\approx 1.28$  for quasi-bipartite graphs (i.e., where no two non-terminals are adjacent) and for complete graphs with edge weights 1 and 2. Our method is considerably simpler and easier to implement than previous approaches. We also prove the first known non-trivial performance bound ( $1.5 \cdot \text{OPT}$ ) for the Iterated 1-Steiner heuristic of Kahng and Robins in quasi-bipartite graphs.

## 1 Introduction

Given an arbitrary weighted graph with a distinguished vertex subset, the *Steiner Tree Problem* seeks a minimum-cost subtree spanning the distinguished vertices. Steiner trees are important in various applications such as VLSI routing [14], wirelength estimation [7], 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], and so efficient approximation heuristics are sought instead of exact algorithms.

Arora established that Euclidean and rectilinear minimum-cost Steiner trees can be efficiently approximated arbitrarily close to optimal [2]. 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  $\epsilon > 0$  [5]. For arbitrary weighted graphs, the best Steiner approximation ratio achievable within polynomial time was gradually improved from 2 to 1.59 in a series of papers [21, 22, 3, 23, 18, 15, 10].

---

\*This work was supported by a Packard Foundation Fellowship, by National Science Foundation Young Investigator Award MIP-9457412, by NSF grant CCR-9988331, and by a GSU Research Initiation Grant. The corresponding author is Alex Zelikovsky, Department of Computer Science, Georgia State University, Atlanta, GA 30303, alexz@cs.gsu.edu, (404) 651-0676.

In this paper we address the graph Steiner Tree Problem by presenting a polynomial-time approximation scheme with a best-known performance ratio approaching  $1 + \frac{\ln 3}{2} \approx 1.55$ . This improves upon the previously best-known ratio of 1.59 due to Hougardy and Prömel [10]. We apply our heuristic for the Steiner Tree Problem to quasi-bipartite graphs (i.e., graphs, where no two non-terminals are adjacent), where our heuristic achieves an approximation ratio of  $\approx 1.28$  within time  $O(mn^2)$  ( $m$  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 exceeds 1.5.

We also show that the well-known Iterated 1-Steiner heuristic of Kahng and Robins [13, 9, 14] achieves an approximation ratio of 1.5 in 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, from the best previously known bound of  $\frac{4}{3}$  [5], to less than 1.28 for our algorithm in this paper.

The remainder of this 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. The basic approximation result for  $k$ -LCA is proved in Section 4. In Sections 5 and 6 we prove an approximation ratio of the  $k$ -LCA algorithm in general graphs, and estimate the performance of the Iterated 1-Steiner and  $k$ -LCA heuristics in quasi-bipartite graphs and in complete graphs with weights 1 and 2. We conclude in Section 7 with possible future research directions. A preliminary version of this paper appeared in [20].

## 2 Definitions, Notation and Basic Properties

Let  $G = (V, E, cost)$  be a graph with non-negative edge costs. 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* seeks a minimum-cost Steiner tree for a given terminal set  $S$ . Any non-terminal vertices contained in a Steiner tree are referred to as *Steiner points*. We can assume that the graph edge cost function 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  $S$ , let  $G_S$  be the complete graph with vertex set  $S$ .

Let  $MST(G_S)$  be a minimum spanning tree of  $G_S$ . For any graph  $H$ , let  $cost(H)$  be 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.,  $cost(MST(G_S)) = mst(G_S)m$ . For brevity, we use  $mst$  to denote  $mst(G_S)$ .

A Steiner tree over a terminal subset  $S' \subset S$  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 a “shrinking” terminal-spanning tree which initially coincides with  $MST(G_S)$ .

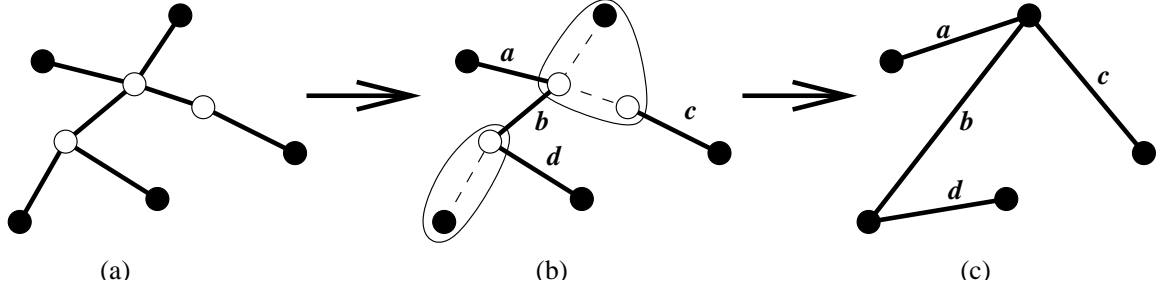


Figure 1: (a) A full component  $K$ : filled circles denote terminals and hollow circles denote Steiner points. (b) Connected components of  $Loss(K)$  to be collapsed; dashed edges belong to  $Loss(K)$ . (c) The corresponding terminal-spanning tree  $C[K]$  with the contracted  $Loss(K)$ .

The relative cost saving of a full component is quantified by the ratio of how much that 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 in the graph obtained by augmenting  $H$  with the tree  $T$ . 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  $gain_T(H) \leq cost(T) - mst(T \cup H)$  since  $T[H]$  cannot cost less than  $MST(T \cup H)$ . In fact, the gain of a full component  $K$  can be also defined as

$$gain_T(K) = cost(T) - mst(T \cup E_0(K)) - cost(K)$$

where  $E_0(K)$  are 0-cost edges between all pairs of terminals of  $K$ . For brevity, the minimum spanning tree of  $T \cup E_0$  will be referred as  $T/E_0$  for any set of 0-cost edges between pairs of terminals in  $S$ .

We will use the following property of *gain* (see Lemma 3.3-4, p.465 in [22] and Lemma 3.14, p.391 in [3]). Let  $E_0$  be an arbitrary set of 0-cost edges between pairs of terminals, and let  $K$  be a full component, then

$$gain_{T/E_0}(K) \leq gain_T(K)$$

This property implies the following key property of *gain*.

**Lemma 1** For any terminal-spanning tree  $T$  and full components  $K_1, K_2, \dots, K_n$ ,

$$gain_T\left(\bigcup_{i=1}^n K_i\right) \leq \sum_{i=1}^n gain_T(K_i)$$

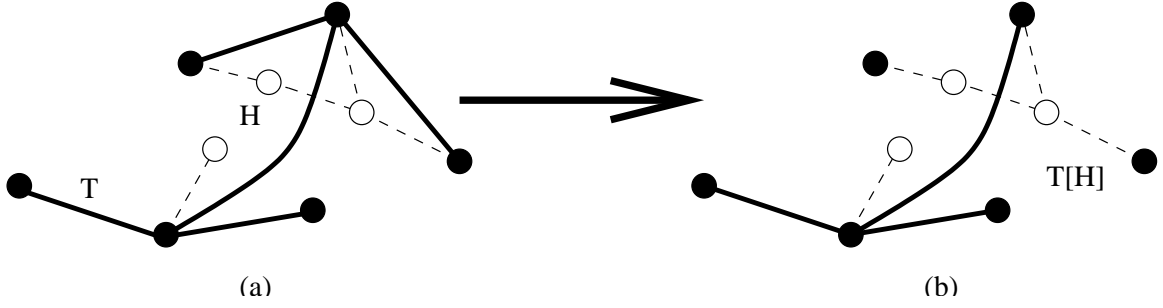


Figure 2: (a) A graph  $H$  (dashed edges) and a terminal-spanning tree  $T$  (solid edges). (b) The corresponding graph  $T[H]$  contains  $H$  and spans all of the terminals.

**Proof:** The proof follows from the following chain of inequalities.

$$\begin{aligned}
\text{gain}_T \left( \bigcup_{i=1}^n K_i \right) &= \text{cost}(T) - \text{cost} \left( T / \bigcup_{i=1}^n E_0(K_i) \right) - \sum_{i=1}^n \text{cost}(K_i) \\
&= \text{cost}(T) - \text{cost}(T \cup E_0(K_1)) - \text{cost}(K_1) + \\
&\quad \text{cost}(T/E_0(K_1)) - \text{cost}(T/E_0(K_1) \cup E_0(K_2)) - \text{cost}(K_2) + \\
&\quad \dots \\
&\quad \text{cost} \left( T / \bigcup_{i=1}^{n-1} E_0(K_i) \right) - \text{cost} \left( T / \bigcup_{i=1}^n E_0(K_i) \right) - \text{cost}(K_n) \\
&= \sum_{i=1}^n \text{gain}_{T/\bigcup_{j \leq i-1} E_0(K_j)}(K_i) \\
&\leq \sum_{i=1}^n \text{gain}_T(K_i)
\end{aligned}$$

□

The minimum-cost connection of the Steiner points of a full component  $K$  to its terminals is denoted  $\text{Loss}(K)$ . Formally,  $\text{Loss}(K)$  is a minimum-cost subgraph of  $K$  containing a path from each Steiner point of  $K$  to one of the terminals of  $K$  (see Figure 1(b)). The following lemma gives a simple method of computing  $\text{Loss}(K)$ .

**Lemma 2** *For any full component  $K$ ,  $\text{Loss}(K) = \text{MST}(K \cup E_0) \setminus E_0$ , where  $K \cup E_0$  is  $K$  combined with zero-cost edges  $E_0$  between all pairs of terminals in  $K$ .*

**Proof:** The forest  $F = \text{MST}(K \cup E_0) \setminus E_0$  connects all Steiner points of  $K$  to the terminals of  $K$  and has cost of  $\text{MST}(K \cup E_0)$ . Note that  $F$  has the minimum possible cost since  $\text{Loss}(K) \cup E_0$  spans all the vertices of  $K$  and therefore cannot cost more than  $\text{MST}(K \cup E_0)$ . □

Intuitively,  $\text{Loss}$  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  $\text{Loss}(K)$  by  $\text{loss}(K)$ . The loss of a union of full components is the sum of their individual losses.

As soon as our algorithm selects a full component  $K$  it *contracts* its  $Loss(K)$ , i.e. “collapses” each connected component of  $Loss(K)$  into a single node (see Figure 1(c)). Formally, a *loss-contracted* full component  $C[K]$  is a terminal-spanning tree over the 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.

The following lemma shows that if no  $k$ -restricted full component can improve a Steiner tree  $H$ , then  $H$  cannot be very expensive, i.e., if we contract the loss of each full component of  $H$ , then the resulting tree costs no more than an optimal  $k$ -restricted Steiner tree.

**Lemma 3** *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  $K_1, \dots, K_p$  be full components of  $Opt_k$ .

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

□

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

### 3 The Algorithm

All previous heuristics (except Berman and Ramaiyer’s [3] approach) with provably good approximation ratios repeatedly choose appropriate full components and then contract them in order to form the overall solution. However, this strategy does not allow us to discard an already-accepted full component even if later we would find out that a better full component *conflicts* with a previously accepted component (two components conflict 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 in 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  $K$  into the overall solution, and (ii) we decrease the gain of full components which conflict with  $K$  by a small value (e.g., less than in the Berman-Ramaiyer algorithm for large  $k$ , and much smaller than in [15] for any  $k$ ).

Our algorithm iteratively modifies a terminal-spanning tree  $T$ , which is initially  $MST(G_S)$ , by incorporating into  $T$  loss-contracted full components greedily chosen from  $G$ . Each such component  $K$  has positive gain, and therefore contains at least 3 terminals and has non-zero loss. 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 the 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 compensate 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. Maximizing the ratio of gain over loss is equivalent to maximizing the profit per unit expense.

<b>Loss-Contracting Algorithm (<math>k</math>-LCA) for Steiner Trees in Graphs</b>
<b>Input:</b> A complete graph $G = (V, E, cost)$ with edge costs satisfying the triangle inequality, a set of terminals $S \subset V$ , and an integer $k$ , $3 \leq k \leq  S $ <b>Output:</b> A $k$ -restricted Steiner tree in $G$ connecting all the terminals in $S$
$T = MST(G_S)$ $H = G_S$ <b>Repeat forever</b> Find a $k$ -restricted full component $K$ with at least 3 terminals maximizing $r = gain_T(K)/loss(K)$ <b>If</b> $r \leq 0$ <b>then exit repeat</b> $H = H \cup K$ $T = MST(T \cup C[K])$ <b>Output</b> the tree $MST(H)$

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

We now describe a polynomial-time implementation of  $k$ -LCA. We first find the all-pairs distances in the graph  $G$ . Then, for each  $k$ -tuple of terminals (there are  $|S|^k$  of them) it is sufficient to try all possible choices of  $k - 2$  Steiner points chosen from the non-terminal nodes of  $V - S$ , because every  $k$ -restricted full component  $K$  is uniquely defined by its Steiner points of degree at least 3. The loss of  $K$  can be determined in time  $O(k^2)$  by finding the minimum spanning tree of  $K \cup E_0$  (see Lemma 2). Thus, we can find all full Steiner components in time  $O(|S|^k \cdot |V - S|^{k-2})$ . Note that the cost and loss do not change in the iterations of  $k$ -LCA.

The number of iterations of  $k$ -LCA cannot exceed the number of full Steiner components  $O(S^k)$ , since  $gain_T(K) = 0$  after contracting the loss of a full component  $K$ . The gain of a full

component  $K$  can be found in time  $O(k)$  after precomputing the longest edges between any pair of nodes in the current minimum spanning tree, which may be accomplished in time  $O(S \log S)$ . Thus, the runtime of all the iterations can be bounded by  $O(k \cdot S^{2k+1} \log S)$ . The total runtime is thus  $O(|S|^k \cdot |V - S|^{k-2} + k \cdot S^{2k+1} \log S)$ .

## 4 Approximation Ratio of Algorithm $k$ -LCA

This section proves the basic approximation result of this paper.

**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:*

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

**Proof:** Let  $K_1, \dots, K_{last}$  be full components chosen by  $k$ -LCA. Let  $T_0 = MST(G_S)$  and let  $T_i$ ,  $i = 1, \dots, last$  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,  $gain_{T_{i-1}}(K_i) < gain_{T_{i-1}}(A \cup B) \leq gain_{T_{i-1}}(A) + gain_{T_{i-1}}(B)$ . Since  $e \notin MST(T_{i-1} \cup K_i)$ , we conclude that  $e \notin MST(K_i \cup E_0)$ , where  $E_0$  are zero cost edges between all terminals of  $K_i$ , and by Lemma 2,  $e \notin Loss(K_i)$ . Thus  $Loss(K_i) = Loss(A) \cup Loss(B)$  and  $loss(K_i) = loss(A) + loss(B)$ . Finally,

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

□

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:

$$\begin{aligned} supergain_{T_{i-1}}(K_i) &= gain_{T_{i-1}}(K_i) + loss(K_i) \\ &= cost(T_{i-1}) - mst(T_{i-1} \cup K_i) + mst(T_{i-1} \cup K_i) - cost(T_i) \\ &= cost(T_{i-1}) - cost(T_i) \end{aligned} \quad (2)$$

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

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

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

$$\frac{G_0}{\text{loss}_k} \leq \frac{\sum_{X_j \in \text{Opt}_k} \text{supergain}_{T_0}(X_j)}{\sum_{X_j \in \text{Opt}_k} \text{loss}(X_j)} \leq 1 + \max_{X_j \in \text{Opt}_k} \left\{ \frac{\text{gain}_{T_0}(X_j)}{\text{loss}(X_j)} \right\} \leq 1 + \frac{\text{gain}_{T_0}(K_1)}{\text{loss}(K_1)} = \frac{g_1}{l_1}$$

Inductively, for  $i = 1, 2, \dots, n$ ,  $\frac{G_{i-1}}{\text{loss}_k} \leq \frac{g_i}{l_i}$ . Therefore,

$$g_i \geq \frac{l_i}{\text{loss}_k} G_{i-1} \quad (3)$$

Each time algorithm  $k$ -LCA accepts a full tree  $K_i$ , it decreases the cost of  $T_i$  by the supergain of  $K_i$ , which results in a decrease of the supergain of  $\text{Opt}_k$  by the same value. Equality (2) yields  $G_i = \text{cost}(T_i) - \text{cost}(\text{Opt}_k) + \text{loss}_k$ . Therefore,  $G_{i-1} - G_i = \text{cost}(T_{i-1}) - \text{cost}(T_i) = g_i$ .

Inequality (3) implies that  $G_i = G_{i-1} - g_i \leq G_{i-1} \left(1 - \frac{l_i}{\text{loss}_k}\right)$ . Since  $G_n > 0$ , unravelling the last inequality yields:

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

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

$$\ln \frac{G_0}{G_n} \geq \sum_{i=1}^n \frac{l_i}{\text{loss}_k} = \frac{\text{loss}(n)}{\text{loss}_k} \quad (4)$$

□

By Lemma 3, after the algorithm stops iterating, the cost of the last tree  $T_{\text{last}}$  will be at most  $\text{opt}_k$ . We stop iterating when  $\text{cost}(T_{n+1}) < \text{opt}_k \leq \text{cost}(T_n)$  for some  $n$ .

We now show how iteration  $n+1$  can be “partially” performed so that  $\text{cost}(T_{n+1})$  will coincide with  $\text{opt}_k$ . We split  $g_{n+1} = \text{supergain}_{T_n}(K_{n+1})$  into two values  $g_{n+1}^1$  and  $g_{n+1}^2$  (i.e.,  $g_{n+1} = g_{n+1}^1 + g_{n+1}^2$ ) such that  $\text{cost}(T_n) - g_{n+1}^1 = \text{opt}_k$  and, therefore,

$$g_{n+1}^1 = \text{cost}(T_n) - \text{opt}_k \quad (5)$$

$$G_n - g_{n+1}^1 = \text{cost}(T_n) - \text{opt}_k + \text{loss}_k - (\text{cost}(T_n) - \text{opt}_k) = \text{loss}_k \quad (6)$$

We split  $l_{n+1} = \text{loss}(K_{n+1})$  into  $l_{n+1}^1$  and  $l_{n+1}^2$  such that  $\frac{g_{n+1}}{l_{n+1}} = \frac{g_{n+1}^1}{l_{n+1}^1}$ . Finally, we set  $\text{loss}^1(n+1) = \text{loss}(n) + l_{n+1}^1$  and

$$G_{n+1}^1 = G_n - g_{n+1}^1 > 0 \quad (7)$$

Since  $\frac{g_{n+1}}{l_{n+1}} = \frac{g_{n+1}^1}{l_{n+1}^1}$ , inequality (4) implies that

$$\ln \frac{G_0}{G_{n+1}^1} \geq \frac{\text{loss}^1(n+1)}{\text{loss}_k} \quad (8)$$

Since  $g_i = \text{gain}_{T_i}(K_i) + \text{loss}(K_i) \geq \text{loss}(K_i) = l_i$ , we have  $\frac{g_{n+1}^2}{l_{n+1}^2} = \frac{g_{n+1}}{l_{n+1}} \geq 1$ , and thus obtain:

$$g_{n+1}^2 \geq l_{n+1}^2 \quad (9)$$

The cost of the approximate Steiner tree after  $n+1$  iterations is at most:

$$\begin{aligned} \text{Approx}(n+1) &= \text{mst}(T_0 \cup K_1 \cup \dots \cup K_{n+1}) \\ &\leq \text{mst}(\text{MST}(T_0 \cup K_1) \cup K_2 \cup \dots \cup K_{n+1}) + \text{loss}(K_1) \\ &\leq \text{mst}(T_1 \cup K_2 \cup \dots \cup K_{n+1}) + \text{loss}(K_1) \\ &\dots \\ &\leq \text{mst}(T_n \cup K_{n+1}) + \sum_{i=1}^n \text{loss}(K_i) \\ &\leq \text{cost}(T_{n+1}) + \text{loss}(n+1) \end{aligned} \quad (10)$$

Since  $\text{Approx}(n)$  decreases with  $n$ , the upper bound on  $\text{Approx}(n+1)$  also bounds  $\text{Approx} = \text{Approx}(\text{last})$ , the output of  $k$ -LCA. We complete the proof of inequality (1) with the following chain of inequalities.

$$\begin{aligned} \text{Approx} &\leq \text{Approx}(n+1) \\ &\stackrel{(10)}{\leq} \text{loss}(n+1) + \text{cost}(T_{n+1}) \\ &= \text{loss}(n) + l_{n+1}^1 + l_{n+1}^2 + \text{cost}(T_n) - g_{n+1}^1 - g_{n+1}^2 \\ &\stackrel{(9)}{\leq} \text{loss}(n) + l_{n+1}^1 + \text{cost}(T_n) - g_{n+1}^1 \\ &\stackrel{(5)}{=} \text{loss}(n) + l_{n+1}^1 + \text{opt}_k \\ &\stackrel{(8)}{\leq} \text{loss}_k \cdot \ln \frac{G_0}{G_{n+1}^1} + \text{opt}_k \end{aligned}$$

$$\begin{aligned}
& \stackrel{(7)}{=} \text{loss}_k \cdot \ln \frac{\text{mst} - \text{opt}_k + \text{loss}_k}{G_n - g_{n+1}^1} + \text{opt}_k \\
& \stackrel{(6)}{=} \text{loss}_k \cdot \ln \frac{\text{mst} - \text{opt}_k + \text{loss}_k}{\text{loss}_k} + \text{opt}_k \\
& = \text{loss}_k \cdot \ln \left( 1 + \frac{\text{mst} - \text{opt}_k}{\text{loss}_k} \right) + \text{opt}_k
\end{aligned}$$

□

## 5 Performance of Algorithm $k$ -LCA in General Graphs

Our estimate of the performance ratio of algorithm  $k$ -LCA in arbitrary graphs is based on estimating optimal  $k$ -restricted Steiner trees. Let  $\rho_k$  be the worst-case ratio of  $\frac{\text{opt}_k}{\text{opt}}$ . It was shown in [6] that  $\rho_k \leq 1 + \lfloor \log_2 k \rfloor^{-1}$ . 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 algorithm  $k$ -LCA converges to  $1 + \frac{\ln 3}{2} < 1.55$  when  $k \rightarrow \infty$  since  $\lim_{k \rightarrow \infty} \rho_k = 1$ . This is an improvement over the algorithm given by Hougardy and Prömel [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  $\text{mst} \leq 2 \cdot \text{opt}$  (see [21]), inequality (1) yields the following upper bound on the output tree cost of  $k$ -LCA:

$$\text{Approx} \leq \text{loss}_k \cdot \ln \left( 1 + \frac{2 \cdot \text{opt} - \text{opt}_k}{\text{loss}_k} \right) + \text{opt}_k$$

Following [15], we show that for any Steiner tree  $T$ ,  $\text{loss}(T) \leq \frac{1}{2} \text{cost}(T)$ . Without loss of generality, we can assume that  $T$  is a rooted tree where all Steiner points have degree at least 3 (degree-2 Steiner points can be disregarded since the graph is complete). For each Steiner point in  $T$ , choose the shortest outgoing edge; then, all chosen edges (i) connect all Steiner points to terminals (thus having cost of at least  $\text{loss}(T)$ ), and (ii) have total cost at most half the cost of  $T$ . Therefore:

$$\text{loss}_k \leq \frac{1}{2} \text{opt}_k$$

The partial derivative  $(\text{loss}_k \cdot \ln(1 + \frac{2 \cdot \text{opt} - \text{opt}_k}{\text{loss}_k}))'_{\text{loss}_k}$  is always positive; the upper bound on  $\text{Approx}$  is therefore maximized when  $\text{loss}_k = \frac{1}{2} \text{opt}_k$ . We thus obtain:

$$\frac{\text{Approx}}{\text{opt}} \leq \frac{\text{opt}_k}{\text{opt}} \cdot \left( 1 + \frac{\ln(\frac{4 \text{opt}}{\text{opt}_k} - 1)}{2} \right)$$

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

## 6 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 no non-terminals are adjacent. 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 [5] proved that this problem is MAX SNP-hard and gave a  $\frac{4}{3} \cdot OPT$  approximation algorithm. Applying Lovász's algorithm for the parity matroid problem (see [16]), an 1.2875-approximation algorithm was given in [4]. We will show that the performance ratio of algorithm  $k$ -LCA approaches 1.28 for such graphs, improving on previously achievable bounds.

### 6.1 The Iterated 1-Steiner Heuristic

The Iterated 1-Steiner heuristic (IIS) (see [13, 9, 14]) repeatedly adds Steiner points to the terminal set, as long as so decreases the cost of the minimum spanning tree over the terminals. Accepted Steiner points are deleted if they become useless, i.e., if their degree become 1 or 2 in the MST over the terminals. A generalization of the Iterated 1-Steiner heuristic to arbitrary graphs, along with a polynomial-time implementation, is given in [1].

Although IIS 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.,  $mst(H \cup v) \geq cost(H)$  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 only a single Steiner point. Therefore, the loss of any full component equals the cost of the cheapest 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,  $loss(H) \leq \frac{1}{3} \cdot cost(H)$ .

We now show that  $gain_{C[H]}(K) \leq 0$  for any full component  $K$ . Condition (ii) implies that  $mst(H \cup K) \geq cost(H)$ . 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,  $mst(C[H] \cup K) \geq mst(H \cup K) - loss(H)$  and  $mst(C[H] \cup K) \geq cost(H) - loss(H) = cost(C[H])$ . Thus,  $gain_{C[H]}(K) \leq cost(C[H]) - mst(C[H] \cup K) \leq 0$ . By Lemma 3,  $cost(H) - loss(H) \leq opt$ , and since  $loss(H) \leq \frac{1}{3} \cdot cost(H)$ , we obtain  $cost(H) \leq \frac{3}{2} \cdot opt$ .  $\square$

The above result helps to 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]).

## 6.2 Runtime of 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 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 algorithm  $k$ -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 into  $H$ . Each iteration consists of  $O(n)$  gain evaluations, each of which can be computed within  $O(m)$  time. Using the appropriate data structures, the  $k$ -LCA algorithm can be implemented within a total runtime of  $O(n^2 \cdot m)$ , where  $m$  is the number of terminals and  $n$  is the number of non-terminals.

## 6.3 Performance Bound of Algorithm $k$ -LCA for Special Graphs

We first estimate the loss of a Steiner trees in quasi-bipartite graphs and in complete graphs with edge weights 1 and 2.

**Lemma 6** *For the Steiner Tree Problem in quasi-bipartite graphs and in complete graphs with edge weights 1 and 2,*

$$mst \leq 2(opt_k - loss_k) \tag{11}$$

**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  $d_0, d_1, \dots, d_{p-1}$ . Assume that  $loss(K) = d_0 = \min\{d_i\}$ . Let  $mst(K)$  be the cost of a minimum spanning tree of  $G_{S'}$ , where  $S'$  is the set of terminals in  $K$ . By the triangle inequality, we have:

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

The bound (11) follows from the fact that  $mst$ , the cost of a minimum spanning tree over  $S$ , 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  $mst \leq 2m - 2$  since all edge weights are at most 2 and  $opt_k \geq m + n - 1$  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,  $mst \leq 2m - 2 = 2(m + n - 1 - n) \leq 2(opt_k - loss_k)$ .  $\square$

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

**Proof:** After substituting the MST bound (11) into inequality (1), we obtain:

$$Approx \leq loss_k \cdot \ln \left( \frac{opt_k}{loss_k} - 1 \right) + opt_k \quad (12)$$

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 (12) 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 (12), yielding:

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

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  $k \rightarrow \infty$ .  $\square$

## 7 Conclusions and Open Problems

We presented a new best-performing polynomial-time heuristic for the classical graph Steiner tree problem. This heuristic, called the  $k$ -Restricted Loss-Contracting Algorithm, can be applied to arbitrary metric spaces. The worst-case performance for  $k$ -LCA depends on the Steiner ratio and the loss of the optimal Steiner tree (i.e., the cost of connecting Steiner points to terminals). We proved that algorithm  $k$ -LCA is currently the best approximation heuristic for the Steiner tree problem in graphs: its approximation ratio is  $\approx 1.55$  for general graphs, and  $\approx 1.28$  for quasi-bipartite graphs and for graphs with edge costs 1 and 2. We also used our techniques to derive the first known non-trivial performance ratio (1.5·OPT) for the Iterated 1-Steiner heuristic of Kahng and Robins [13, 9, 14, 1] in quasi-bipartite graphs.

Chief among the remaining open problems is finding heuristics for the classical graph Steiner problem with improved performance bounds. Other special cases of the Steiner problem for

special metrics, cost functions, and graph types may be explored separately, where it may be possible to exploit the specific underlying structure to further improve the performance bounds. Interestingly, our  $k$ -LCA algorithm is the first (and so far only) heuristic that works provably well for all of the special graph types discussed above.

From a practical perspective, for any given fixed performance bound it would be useful to minimize the running times of the associated heuristics, and to quantify and explore various tradeoffs between running times and solution quality. Finally, it would be useful to implement the various heuristics and explore their practical runtime and empirical solution quality, by comparing the performance of these implementations side-by-side on various classes and sizes of inputs.

## Acknowledgments

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

## References

- [1] ALEXANDER, M. J., AND ROBINS, G., “New Performance-Driven FPGA Routing Algorithms”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15 (1996), pp. 1505–1517.
- [2] S. ARORA, “Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems”, *Journal of ACM*, 45 (1998), pp. 501-555.
- [3] P. BERMAN AND V. RAMAIYER, “Improved Approximations for the Steiner Tree Problem”, *Journal of Algorithms*, 17 (1994), pp. 381–408.
- [4] P. BERMAN, M. FÜRER AND A. ZELIKOVSKY, “Applications of the Matroid Parity Problem to Approximating Steiner Trees”, *Tech. Rep. 980021, Computer Science Dept., UCLA*, Los Angeles, 1998.
- [5] M. BERN AND P. PLASSMANN, “The Steiner Tree Problem with Edge Lengths 1 and 2”, *Information Processing Letters* 32 (1989), pp. 171–176.
- [6] A. BORCHERS AND D.-Z. DU, “The  $k$ -Steiner Ratio in Graphs”, *SIAM Journal on Computing* 26 (1997), pp. 857–869.
- [7] 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.
- [8] M. R. Garey, D. S. Johnson, “The Rectilinear Steiner Tree Problem is NP-Complete”, *SIAM Journal on Applied Mathematics*, 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. PRÖMEL, “A 1.598 Approximation Algorithm for the Steiner Problem in Graphs”, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 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), pp. 47–65.
- [16] L. LOVÁSZ 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”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19 (2000), pp. 1129-1139.
- [18] H. J. PRÖMEL AND A. STEGER, “RNC-Approximation Algorithms for the Steiner Problem”, *Proceedings 14th Annual Symposium on Theoretical Aspects of Computer Science* (1997), pp. 559–570.
- [19] S. RAJAGOPALAN AND V. V. VAZIRANI, “On the Bidirected Cut Relaxation for the Metric Steiner Tree Problem”, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 742–757.
- [20] G. ROBINS AND A. ZELIKOVSKY, “Improved Steiner Tree Approximation in Graphs”, *Proceedings of SIAM-ACM Symposium on Discrete Algorithms* (2000), pp. 770–779.
- [21] H. TAKAHASHI AND A. MATSUYAMA, “An Approximate Solution for the Steiner Problem in Graphs”, *Math. Jap.* 24 (1980), pp. 573–577.
- [22] A. ZELIKOVSKY, “An 11/6-Approximation Algorithm for the Network Steiner Problem”, *Algorithmica* 9 (1993), pp. 463–470.
- [23] A. ZELIKOVSKY, “Better Approximation Bounds for the Network and Euclidean Steiner Tree Problems”, *Technical report CS-96-06, University of Virginia, 1996*.