A 1.598 Approximation Algorithm for the Steiner Problem in Graphs

Stefan Hougardy [*]   Hans Jürgen Prömel [†]

Abstract

We present a general iterative framework for improving the performance ratio of Steiner tree approximation algorithms. By applying this framework to one specific algorithm we obtain a new polynomial time approximation algorithm for the Steiner tree problem in graphs that achieves a performance ratio of 1.598 after 11 iterations. This beats the so far best known factor of 1.644 due to Karpinski and Zelikovsky [10]. With the help of a computer program we estimate the limit performance of our algorithm to be 1.588.

1 Introduction

Given a graph $G = (V, E)$, a subset $T \subset V$ of *terminals* and a length function $c : E \to \mathbb{R}$ on the edges of G, then the *Steiner Tree Problem* asks for a shortest network connecting the vertices of T. The Steiner tree problem appears in many different kinds of applications. For example in the *network routing problem* a communication server has to distribute the same data to several nodes in a network by selecting a minimum cost set of links that connect the server to all nodes. Other examples of applications of Steiner tree problems are the computation of phylogenetic trees in biology or the routing phase in VLSI-design.

Two famous special cases of the Steiner tree problem in graphs are the *Euclidean* Steiner tree problem and the *rectilinear* Steiner tree problem. In both problems the task is to find a shortest network connecting given points in the plane. The only difference in these special cases is the metric used to measure distances. In the Euclidean Steiner tree problem distances are measured by the L_2, i.e. the Euclidean metric, while in the rectilinear Steiner tree problem distances are measured by the L_1 metric. These two special cases of the Steiner tree problem have been studied intensely. However, Steiner tree problems arising in practical applications usually involve cost functions that do not satisfy the L_1 or L_2 metric. This motivates the study of the Steiner tree Problem in graphs. Since in the Steiner

tree problem in graphs we do not have any restrictions on the length function for the edges in the graph, we can model any Steiner tree problem in any metric by a Steiner tree problem in graphs.

The Steiner tree problem is a well known NP-complete problem even in the very special cases of Euclidean or rectilinear metric. This fact rises the question for provably good heuristics for these problems. Arora [1] has shown that Euclidean and rectilinear Steiner tree problems admit a polynomial time approximation scheme, i.e., they can be approximated in polynomial time up to a factor of $1 + \epsilon$ for any constant $\epsilon > 0$.

In contrast to these two special cases the Steiner problem in graphs is known to be APX-complete [6][2], which means unless P=NP there cannot exist a polynomial time approximation scheme for this problem. Here we will present a polynomial time approximation algorithm for the Steiner tree problem in graphs which achieves a performance ratio of 1.598. This beats the so far best known factor of 1.644 due to Karpinski and Zelikovsky [10]. The new idea of our algorithm is to iteratively use a parameterized Steiner tree algorithm to improve the solution found so far by the algorithm. This is done by successively adding in a certain way additional terminals to the set of terminals given in the beginning. While this clearly worsens the value of an optimal solution we prove that by choosing the optimal sequence of parameters for the algorithm the performance ratio decreases up to a certain point faster than the quality gets worse. After 11 such iterations we already get a performance ratio of 1.598. By a computer program we have numerically estimated the limit performance for k iterations for $k \to \infty$ and it turned out that this value is about 1.588.

2 Previous Results and our Contribution

There exist several different heuristics for the Steiner tree problem in graphs. But only few of them have provably good performance ratios. Table 1 gives a survey on such results. (The algorithm of Prömel and Steger [11] is a randomized algorithm, all other algorithms listed in the table are deterministic algorithms. The performance ratio 1.734 is the value that is obtained by applying the

---
[*]Humboldt-Universität zu Berlin, Institut für Informatik, 10099 Berlin, GERMANY, hougardy@informatik.hu-berlin.de

[†]Humboldt-Universität zu Berlin, Institut für Informatik, 10099 Berlin, GERMANY, proemel@informatik.hu-berlin.de

2

result of Borchers and Du [5] to [4]. The value given in [4] is 1.746). All performance ratios given in this paper are rounded up to the third digit.

Year	Performance Ratio	Authors
1980	2.000	Takahashi, Matsuyama [12]
1993	1.834	Zelikovsky [13]
1994	1.734	Berman, Ramaiyer [4]
1995	1.694	Zelikovsky [14]
1997	1.667	Prömel, Steger [11]
1997	1.644	Karpinski, Zelikovsky [10]
1998	1.598	Hougardy, Prömel [this paper]

Table 1: Steiner tree approximation algorithms

Takahashi and Matsuyama [12] were the first proving that the well known minimum spanning tree heuristic achieves a performance ratio of 2. The minimum spanning tree heuristic is based on the idea of greedily adding a shortest connection between a *pair* of terminals. This idea naturally extends to adding shortest connections between k-*tuples* of terminals, for fixed k. All approximation algorithms for the Steiner tree problem listed in Table 1 are based on this simple idea.

The present approach to get better performance ratios for the Steiner tree problem in graphs is to iteratively apply a series of algorithms to the output of its predecessor. This way we obtain the 1.598 record performance ratio for the Steiner tree problem in graphs. To achieve this ratio it is shown that only 11 iterations are necessary. This implies that the running time of the algorithm only increases by a factor of 11 compared to the running time of the base algorithm applied in each step.

This new approach provides a general framework that can be applied to any known heuristic for the Steiner tree problem in graphs to obtain a potentially better algorithm. Besides the presentation of this general framework the main contribution of our paper is to show that by applying the new framework to one specific heuristic we obtain a provably good performance ratio that outperforms all results known before.

3 Definitions and Notations

The Steiner tree problem in graphs is defined as follows: Given a graph $G = (V, E)$ with a length function

$c : E \to \mathbb{R}$ and a subset $T \subseteq V$ of *terminals* of the vertices of G, find a shortest network connecting the vertices in T. Any network connecting the vertices of T is called a *Steiner tree* for T. Without loss of generality we assume that G is a complete graph and the length of each edge between two vertices u and v equals the length of a shortest path between u and v. In general, a Steiner tree contains not only the vertices from T but also vertices from $V - T$. These vertices are called *Steiner points*. The *length* of a Steiner tree B is the sum of the lengths of all edges in the tree and is denoted by $d(B)$. The *loss* of a Steiner tree B is the length of a minimum spanning forest in B such that each component in the forest contains at least one vertex from T. The loss of a Steiner tree B is denoted by $l(B)$. Clearly, the loss of a given Steiner tree can be computed in polynomial time by using a minimum spanning tree algorithm.

For a given set T of terminals we denote by $MST(T)$ a minimum spanning tree for T and by $mst(T)$ its length. A *Steiner minimal tree* for a set T of terminals, denoted by $SMT(T)$ is a shortest possible Steiner tree for T. Its length is denoted by $smt(T)$.

A Steiner tree is called *full* if all terminals are leaves in the tree. If a Steiner tree is not full, it can be decomposed into *full components* by splitting off the terminals that are not leaves. A k-*restricted* Steiner tree is a full Steiner tree with at most k terminals. By the *contraction* of a set S of vertices we understand the setting of all lengths to zero for edges between vertices in S.

The *performance ratio* of an approximation algorithm for the Steiner tree problem is an upper bound on the length of the Steiner tree found by the algorithm divided by the length of an optimal solution.

4 The Algorithm

Our iterative algorithm is based on the generalization of the relative greedy heuristic which was suggested by Karpinski and Zelikovsky [10]. Their algorithm depending on k and on some constant α is denoted by k-RGH(α). It works as follows: among all k-restricted Steiner trees choose one tree B that minimizes $(d(B) + \alpha l(B))/(mst(T) - mst(T/B))$. Then put $T := T/B$. Here we denote by T/B the set of terminals that is obtained from T after contracting the terminals of B. The algorithm stops when $mst(T) = 0$. The union of all generated k-trees is the output of the algorithm. Karpinski and Zelikovsky have shown that for $k \to \infty$ the Steiner tree B generated by their algorithm satisfies

$$d(B) + \alpha l(B) \quad \leq \quad smt(T)\left(1 + \frac{\alpha}{2}\right) \cdot$$

$$(4.1) \qquad \cdot \left(1 + \ln \frac{mst(T)}{\left(1 + \frac{\alpha}{2}\right)smt(T)}\right)$$

We denote the algorithm k-RGH(α) in the limit $k \to \infty$ by RGH(α). In this paper we will only make statements about the limit performance of our algorithm. While the limit performance will not be reached in polynomial time, one can get for any $\epsilon > 0$ a polynomial approximation algorithm with a ratio that is only by a factor of $1+\epsilon$ away from the limit performance. All performance ratios in this paper that are given as actual numbers are rounded up. Therefore, such performance ratios can be reached in polynomial time.

Let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$ with $\alpha_1 \geq \ldots \geq \alpha_k = 0$. Then the *iterated relative greedy heuristic* for the vector $\vec{\alpha}$, IRGH$(\vec{\alpha})$ for short, works as follows:

Algorithm IRGH$(\vec{\alpha})$

$T_0 :=$ terminals of $G = (V, E)$

for $i := 1$ **to** k **do**

    apply RGH(α_i) to T_{i-1} to get a Steiner tree B_i;

    $T_i := T_{i-1} \cup \{$Steiner points of $B_i\}$

output: a minimum spanning tree for T_k

Note that this algorithm represents a general framework for improving the performance ratio of Steiner tree approximation algorithms. One can for example apply this framework to the algorithm IRGH$(\vec{\alpha})$ itself, to obtain an even better approximation algorithm for the Steiner tree problem. However, it is easy to see that the performance ratio of such an algorithm can always be beaten by IRGH$(\vec{\alpha})$ with a sufficiently large number of iterations.

In the next section we analyze the performance ratio of IRGH$(\vec{\alpha})$.

5 Analysis of the Algorithm

For a sequence $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_k = 0$ the algorithm IRGH$(\vec{\alpha})$ applies iteratively algorithm $RGH(\alpha_i)$ for $i = 1, \ldots k$ to a given instance $G = (V, E, c, T)$ of the Steiner tree problem. Let s_i denote the value of the optimum solution before the ith iteration of the algorithm. We may assume $s_1 = 1$. By d_i we denote the length of

the solution tree returned in the ith iteration. Its loss is denoted by l_i. We have $d_0 = mst$ and $l_0 = 0$. We can bound the length of the optimum solution after i iterations as follows:

$$(5.2) \qquad s_i \leq s_{i-1} + l_{i-1} \leq \sum_{j=1}^{i-1} l_j$$

The following lemma gives a simple bound on the length d_k of the Steiner tree returned after k iterations in terms of the lengths of the Steiner trees that are generated in all previous iterations.

LEMMA 5.1. *The length d_k of the Steiner tree generated by algorithm IRGH$(\vec{\alpha})$ after k iterations satisfies:*

$$(5.3) \qquad d_k \quad \leq \quad C_k \cdot \left(1 + \ln \frac{d_{k-1}}{C_k}\right)$$

with C_i defined for $i = 1, \ldots, k$ as follows:

$$C_i :=$$

$$(5.4) \qquad \left(1 + \sum_{j=1}^{i-1} \frac{C_j\left(1 + \ln \frac{d_{j-1}}{C_j}\right) - d_j}{\alpha_j}\right)\left(1 + \frac{\alpha_i}{2}\right)$$

Proof. First note that for any numbers $0 < x \leq y < c$ we have $x(1 + \ln \frac{c}{x}) \leq y(1 + \ln \frac{c}{y})$. Now by using (4.1) and (5.2) one can prove by induction (the details are left out in this extended abstract):

$$(5.5) \quad d_i + \alpha_i \cdot l_i \quad \leq \quad s_i \cdot \left(1 + \frac{\alpha_i}{2}\right) \cdot$$

$$\cdot \left(1 + \ln \frac{d_{i-1}}{s_i \cdot \left(1 + \frac{\alpha_i}{2}\right)}\right)$$

$$(5.6) \qquad \leq \quad C_i \cdot \left(1 + \ln \frac{d_{i-1}}{C_i}\right)$$

with the following bound for l_i:

$$(5.7) \qquad l_i \quad \leq \quad \frac{C_i \cdot \left(1 + \ln \frac{d_{i-1}}{C_i}\right) - d_i}{\alpha_i}$$

By making use of $\alpha_k = 0$ we get the desired result. $\square$

In the following we are going to compute a bound for d_k that only depends on the vector $\vec{\alpha}$. To do so we have to compute the maximum of the function $C_k \cdot \left(1 + \ln \frac{d_{k-1}}{C_k}\right)$.

4

The following fact will be used several times throughout the proofs. Let f be a function in x. Then

$$(5.8) \quad \frac{\partial}{\partial x}\left(\left(1+\ln\frac{x}{f}\right)\cdot f\right) = \frac{f}{x}+\ln\frac{x}{f}\cdot\frac{\partial}{\partial x}f$$

The next theorem now gives a bound on the length d_k of the Steiner tree generated by algorithm IRGH($\vec{\alpha}$) that only depends on $\vec{\alpha}$. Note that since we assume $s_1 = 1$ the value d_k is the desired approximation ratio of our algorithm IRGH($\vec{\alpha}$).

THEOREM 5.1. *The length d_k of the Steiner tree generated by algorithm IRGH($\vec{\alpha}$) after k iterations satisfies:*

$$(5.9) \quad d_k \le \left(\alpha_1 + \left(1+\frac{\alpha_1}{2}\right)\left(1+\ln\frac{2}{1+\frac{\alpha_1}{2}}\right)\right)\prod_{j=2}^{k} x_j$$

with x_i defined by

$$(5.10) \quad \alpha_{i-1}\cdot x_i - \alpha_i = \left(1+\frac{\alpha_i}{2}\right)\cdot\ln\frac{1}{x_i}$$

Proof. We have

$$(5.11) \quad \frac{\partial}{\partial d_{k-1}}C_k = -\frac{1}{\alpha_{k-1}}$$

Therefore a necessary condition for a maximum of the function $C_k\cdot\left(1+\ln\frac{d_{k-1}}{C_k}\right)$ is

$$(5.12) \quad \frac{C_k}{d_{k-1}} - \frac{1}{\alpha_{k-1}}\cdot\ln\frac{d_{k-1}}{C_k} = 0$$

Let x_k be the solution of the equation

$$(5.13) \quad \alpha_{k-1}\cdot x_k = \ln\frac{1}{x_k}$$

In the following we will make use of the fact that the quadratic form $x^T Q x$ with Q being the $(k-1)\times(k-1)$ matrix of the second partial derivations of (5.3) is negative definit at the critical point. This can be shown by proving that Q is a diagonal matrix at the critical point with all entries being negative. The details of these calculations are left out in this extended abstract. Now by substituting equation (5.12) in (5.3) and using $\alpha_k = 0$ we get

$$d_k \le C_k\cdot\left(1+\ln\frac{d_{k-1}}{C_k}\right)$$
$$\le C_k\cdot\left(1+\frac{\alpha_{k-1}\cdot C_k}{d_{k-1}}\right)$$

$$\le \frac{C_k}{d_{k-1}}\cdot(d_{k-1}+\alpha_{k-1}\cdot C_k)$$

$$\le x_k\alpha_{k-1}\left(1+\sum_{j=1}^{k-2}\frac{C_j\left(1+\ln\frac{d_{j-1}}{C_j}\right)-d_j}{\alpha_j}\right)+$$
$$(5.14) \quad + x_k C_{k-1}\left(1+\ln\frac{d_{k-2}}{C_{k-1}}\right)$$

Using (5.8) we get as a necessary condition for a maximum of (5.14):

$$(5.15) \quad 0 = -\frac{\alpha_{k-1}}{\alpha_{k-2}}+\frac{C_{k-1}}{d_{k-2}}+$$
$$+\ln\frac{d_{k-2}}{C_{k-1}}\cdot\left(-\frac{1}{\alpha_{k-2}}\right)\cdot\left(1+\frac{\alpha_{k-1}}{2}\right)$$

Let x_{k-1} be the solution of

$$(5.16) \quad \alpha_{k-2}\cdot x_{k-1}-\alpha_{k-1} = \left(1+\frac{\alpha_{k-1}}{2}\right)\cdot\ln\frac{1}{x_{k-1}}$$

Now (5.14) can be written as

$$d_k \le x_k\left(\alpha_{k-1}\left(1+\sum_{j=1}^{k-2}\frac{C_j\left(1+\ln\frac{d_{j-1}}{C_j}\right)-d_j}{\alpha_j}\right)+\right.$$
$$\left.+ C_{k-1}\left(1+\ln\frac{d_{k-2}}{C_{k-1}}\right)\right)$$

$$\le x_k\cdot\left(\alpha_{k-1}\cdot\frac{C_{k-1}}{1+\frac{\alpha_{k-1}}{2}}+C_{k-1}\cdot\left(1+\ln\frac{d_{k-2}}{C_{k-1}}\right)\right)$$

$$\le x_k\cdot C_{k-1}\cdot\left(\frac{2\alpha_{k-1}}{2+\alpha_{k-1}}+1+\right.$$
$$\left.+\frac{2}{2+\alpha_{k-1}}\cdot\left(\alpha_{k-2}\cdot\frac{C_{k-1}}{d_{k-2}}-\alpha_{k-1}\right)\right)$$

$$\le x_k\cdot\frac{C_{k-1}}{d_{k-2}}\cdot\frac{1}{2+\alpha_{k-1}}\cdot$$
$$\cdot((2+\alpha_{k-1})d_{k-2}+2\alpha_{k-2}C_{k-1})$$

$$\le x_k\cdot x_{k-1}\left(d_{k-2}+\right.$$
$$\left.+\alpha_{k-2}\left(1+\sum_{j=1}^{k-2}\frac{C_j\left(1+\ln\frac{d_{j-1}}{C_j}\right)-d_j}{\alpha_j}\right)\right)$$

$$\le x_k x_{k-1}\left(\alpha_{k-2}\left(1+\sum_{j=1}^{k-3}\frac{C_j\left(1+\ln\frac{d_{j-1}}{C_j}\right)-d_j}{\alpha_j}\right)\right.$$
$$\left.+ C_{k-2}\left(1+\ln\frac{d_{k-3}}{C_{k-2}}\right)\right)$$

The result stated in the theorem follows by induction. $\square$

6   Numerical results

For a fixed value of k using (5.9) and (5.10) one can easily optimize the values of $\alpha_1, \ldots, \alpha_k$ to obtain the best possible bound for d_k. The following table lists the performance ratio of our algorithm for some small values of k for optimally chosen values of α_i.

k	ratio
1	1.694
2	1.644
3	1.626
4	1.616
5	1.611
6	1.607
7	1.604
$\vdots$	$\vdots$
11	1.598
12	1.597
$\vdots$	$\vdots$
∞	≈ 1.588

Table 2: Performance ratio of k-IRGH

Note that in the special case $k = 1$ we obtain the algorithm of Zelikovsky [14] while for $k = 2$ we obtain the algorithm of Karpinski and Zelikovsky [10]. In the latter case α_1 has to be chosen as 0.436. For $k = 3$ one has to choose $\alpha_1 = 0.698$ and $\alpha_2 = 0.248$. In the case $k = 11$ the sequence $(\alpha_1, \ldots, \alpha_k)$ looks as follows:
$$(1.365, 1.026, 0.792, 0.615, 0.474,$$
$$0.360, 0.264, 0.183, 0.114, 0.053, 0)$$
The last row in the table contains the limit performance ratio that was obtained by using a computer program that numerically estimates the performance ratio of our algorithm for large values of k. We obtain that the performance ratio behaves roughly as $1.588 + 0.114/k$ which indicates that the limit performance ratio of our algorithm lies at 1.588.

7   Concluding Remarks

We have presented a general iterative framework for improving the approximation ratio of Steiner tree approximation algorithms. By applying this framework to one specific algorithm we obtain a new Steiner tree approx-

imation algorithm with a performance ratio of 1.598. This beats the so far best known ratio of 1.644.

The number 1.598 is only an upper bound on the performance ratio of the algorithm IRGH($\vec{\alpha}$). We do not know of any reasonable lower bound. That is we do not even know any graphs where IRGH($\vec{\alpha}$) can achieve a performance ratio of 1.5. It is very likely that an improved analysis of our algorithm yields an even better bound on the performance ratio of IRGH($\vec{\alpha}$).

The iteration framework presented here can also be applied to other Steiner tree type problems. The directed Steiner tree problem [7] and the group Steiner tree problem [9] are examples of such problems which have been studied recently.

The ratio 1.598 presented in this paper is still far away from the best possible (unless P=NP) lower bound which is currently 5601/5600 [8]. The interesting question of the best possible performance ratio of a polynomial time approximation algorithm for the Steiner tree problem in graphs remains open.

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] S. Arora, C. Lund, R. Motwani, M. Sudan and M.Szegedy, *Proof verification and hardness of approximation problems*, Proceedings 33rd Annual Symposium on Foundations of Computer Science (1992), 14–23.

[3] S. Arora, M. Grigni, D. Karger, P. Klein and A. Woloszyn, *A polynomial-time approximation scheme for weighted planar graph TSP*, Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (1998), 33–41.

[4] P. Berman and V. Ramaiyer, *Improved approximations for the Steiner tree problem* Journal of Algorithms 17 (1994), 381–408.

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

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

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

[8] 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).

6

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

[10] M. Karpinski and A. Zelikovsky, *New approximation algorithms for the Steiner tree problems*, Journal of Combinatorial Optimization 1 (1997), 47–65.

[11] H.J.Prömel and A.Steger, *RNC-approximation algorithms for the Steiner problems*, Proceedings 14th Annual Symposium on Theoretical Aspects of Computer Science (1997), 559–570.

[12] H. Takahashi and A. Matsuyama, *An approximate solution for the Steiner problem in graphs*, Math. Jap. 24 (1980), 573–577.

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

[14] A. Zelikovsky, *Better approximation bounds for the network and Euclidean Steiner tree problems*, Technical report CS-96-06, University of Virginia.