Steiner Trees in Uniformly Quasi-Bipartite Graphs * †

Clemens Gröpl Stefan Hougardy Till Nierhoff Hans Jürgen Prömel

Abstract

The area of approximation algorithms for the Steiner tree problem in graphs has seen continuous progress over the last years. Currently the best approximation algorithm has a performance ratio of 1.550. This is still far away from 1.0074, the largest known lower bound on the achievable performance ratio. As all instances resulting from known lower bound reductions are uniformly quasi-bipartite, it is interesting whether this special case can be approximated better than the general case. We present an approximation algorithm with performance ratio $73/60 < 1.217$ for the uniformly quasi-bipartite case. This improves on the previously known ratio of 1.279 of Robins and Zelikovsky. We use a new method of analysis that combines ideas from the greedy algorithm for set cover with a matroid-style exchange argument to model the connectivity constraint. As a consequence, we are able to provide a tight instance.

Keywords: Analysis of algorithms, approximation algorithms, network design, Steiner tree problem in graphs.

1 Introduction

Given a graph $G = (V, E)$, a length function on its edges, and a set $R \subseteq V$ of *terminals*, a *Steiner tree* is a connected subgraph of G spanning all vertices in R. The Steiner tree problem in graphs is to find a shortest Steiner tree. It is a classical NP-hard problem [10] and relevant to many real-world applications. This motivates the search for good approximation algorithms.

It has been known at least since 1968 [6, p. 24] that the performance ratio of the minimum spanning tree heuristic is 2. During the last ten years, several authors published algorithms with decreasing performance ratios [3, 9, 11, 12, 18, 19]. The best value known today is 1.550 and due to Robins and Zelikovsky [15]. For more details on these approximation algorithms see [8].

The PCP-Theorem [1] and an approximation preserving reduction from vertex cover [4] imply that the performance ratio of a polynomial time approximation algorithm for the Steiner tree problem in graphs cannot get arbitrarily close to 1. There exists a lower bound on the achievable performance ratio which has been enlarged in some effort over the last years. Currently the best value known is 1.0074 and due to Thimm [16].

The gap between lower and upper bounds is still quite large. As a step towards narrowing this gap, we show in this article that there is an algorithm with a performance ratio of $73/60 < 1.217$ for the special case of uniformly quasi-bipartite instances. An instance of the Steiner tree problem is called *uniformly quasi-bipartite*, if the set $V \setminus R$ is stable and if, for each vertex in that set, all incident edges have the same length.

As all existing lower bound reductions [4,10,16] produce uniformly quasi-bipartite instances, our result has two consequences: On the one hand, it implies that reductions of this type cannot yield inapproximability results better than 1.217. On the other hand, our method of analysis differs from those previously applied, so that it might indicate an alternative approach to the general case. As a by-product, this method allows for a simple instance that shows that the performance ratio of $73/60$ is tight. Such instances are not yet known for the other approximation algorithms for the Steiner tree problem in graphs [3, 9, 11, 18, 19]. Lower bounds for the performance ratio of some of these algorithms are given in [7].

A slightly more general case are *quasi-bipartite* graphs. In these instances, the set $V \setminus R$ is stable, but the edges incident with a vertex in that set may have different lengths. Rajagopalan and Vazirani [13] gave a $3/2+\varepsilon$ approximation algorithm based on the primal-dual method for quasi-bipartite instances. Robins and Ze-

*The project "Approximative, randomisierte und probabilistische Algorithmen für Kombinatorische Optimierungsprobleme" is supported by the Deutsche Forschungsgemeinschaft, no. Pr 296/6-1.

†Address of all authors: Institut für Informatik, Humboldt-Universität zu Berlin, 10099 Berlin, Germany,
{groepl, hougardy, nierhoff, proemel}@informatik.hu-berlin.de.

likovsky [15] showed that the popular 1-Steiner heuristic has a performance ratio of $3/2$ in this case. Moreover, they showed that the performance ratio of their loss contracting algorithm is 1.279 in quasi-bipartite instances.

2 Algorithm `Greedy-MSS`

Every Steiner tree can be split into so called *full components*. A full component is a Steiner tree for a subset of R in which every terminal is a leaf. The length of a full component is the sum of its edge lengths. A Steiner tree is a collection of full components which is connected and covers R. The Steiner tree problem can be formulated as a hypergraph problem as follows: Every full component is represented by a hyperedge consisting of its terminals. Let FC be the set of these hyperedges and let $|t|$ denote the length of the full component represented by $t \in FC$. Then the Steiner tree problem is the *minimum spanning subhypergraph* problem in the weighted hypergraph $(R, FC, |\cdot|)$. Since FC is closed under taking subsets, a minimum spanning subhypergraph contains no cycle, i.e., it is a hypertree. In the following, we will use this formulation of the Steiner tree problem.

The algorithm that we consider in this paper is shown in Figure 1. We call it `Greedy-MSS`. In every iteration of the While-loop a hyperedge is selected which minimizes the length per connection ratio

$$f_i(t) := \frac{|t|}{c_i(t)}\,,$$

where $c_i(t)$ is defined as the difference between the numbers of connected components of $(R, \{t_1, \ldots, t_i\})$ and $(R, \{t, t_1, \ldots, t_i\})$.

Note that in quasi-bipartite graphs, algorithm `Greedy-MSS` is identical to Rayward-Smith's average distance heuristic [14].

```
Input  a weighted hypergraph (R, FC, |·|)
i ← 0
While (R, {t_1, ..., t_i}) is not connected:
      Select t_{i+1} ∈ FC that minimizes f_i
      i ← i + 1
p ← i
Output  t_1, ..., t_p
```

Figure 1: Algorithm `Greedy-MSS`.

3 Performance Analysis

In general the minimum spanning subhypergraph problem is even harder than the Steiner tree problem. Wolsey [17] showed that the greedy algorithm achieves a performance ratio of $H(k-1)$ if the size of every hyperedge is bounded by k. Here $H(k) := \sum_{i=1}^{k} 1/i = \ln k + O(1)$ denotes the k-th harmonic number. This result is best possible [5] and does not immediately lead to a good approximation algorithm for the Steiner tree problem.

Note that in hypergraphs which result from Steiner tree problems (in contrast to general hypergraphs), all subhyperedges of a hyperedge exist and have smaller weights. Therefore we may focus on solutions which contain no cycles. Our performance analysis is based on an even stronger property which holds for uniformly quasi-bipartite instances.

In these instances, the full components are stars in which all edges have the same length. The weight of a subhyperedge $t' \subseteq t$ is proportional to its size. Therefore the algorithm is not quite as bad off if it fails to select a "good" hyperedge in one step. It may later decide to include a subset of it without having to pay the full price.

Theorem 3.1
`Greedy-MSS` *computes a* $73/60 < 1.217$ *approximation for the Steiner tree problem in uniformly quasi-bipartite graphs.*

Proof. Let s be a hyperedge of the optimal solution. We know that when the algorithm decided to select t_i, there was a competing minimal subhyperedge $s' \subset s$ with $c_{i-1}(s') = c_{i-1}(s)$. As s' represents a star with $c_{i-1}(s) + 1$ edges, we know that

$$|s'| = \frac{c_{i-1}(s) + 1}{c_0(s) + 1} \cdot |s|$$

and we get the inequality

$$\frac{|t_i|}{c_{i-1}(t_i)} \leqslant \frac{|s'|}{c_{i-1}(s')} = \frac{\frac{c_{i-1}(s)+1}{c_0(s)+1}|s|}{c_{i-1}(s)} = \left(1+\frac{1}{c_{i-1}(s)}\right)\frac{|s|}{c_0(s)+1}\,. \quad (1)$$

Let $ALG := \{t_1,\dots,t_p\}$ and let OPT be the set of hyperedges of an optimal solution. In order to estimate the length of ALG in terms of the length of OPT, we introduce artificial edge sets A resp. B for the algorithmic and for the optimal solution which stand for the connections accomplished by the hyperedges of both solutions. Then we distribute the length of the edges in A among those in B using a matroid-style exchange argument.

Let $ALG_i := \{t_1,\dots,t_i\}$. In the following we define $\ell_i \in \mathbb{N}$ and edges $a_1,\dots,a_{\ell_i}$ between terminals such that $A_i := \{a_1,\dots,a_{\ell_i}\}$ has the same connected components as ALG_i. We start with $\ell_0 := 0$ and $A_0 := \emptyset$ and set $\ell_i := \ell_{i-1} + c_{i-1}(t_i) = \ell_{i-1} + c_0(t_i)$. The new edges $a_{\ell_{i-1}+1},\dots,a_{\ell_i} \subseteq t_i$ are chosen to form an arbitrary spanning tree for the vertices of t_i. The length of t_i is distributed uniformly among the new edges, i. e., their lengths are $|a_j| := |t_i|/(\ell_i - \ell_{i-1})$ for $j = \ell_{i-1}+1,\dots,\ell_i$. Let $A := A_{\ell_p}$. Clearly, ℓ_p is the number of terminals minus one.

Let $B_0 := \emptyset$ and $T_0 := OPT$. We successively define sets $B_j = \{b_1,\dots,b_j\}$ of edges between terminals and spanning hypertrees T_j for $j = 1,\dots,\ell_p$. If we insert a_j into T_{j-1}, there are two possibilities. If $a_j \in T_{j-1}$ then we let $b_j := a_j$ and $T_j := T_{j-1}$. Otherwise, $T_{j-1} + a_j$ contains a cycle. As A is a tree, this cycle must contain a hyperedge $s \in T_{j-1} \setminus A$. We let $T'_{j-1} := T_{j-1} - s$ and choose an edge $b_j \subseteq s$ which connects the two connected components of T'_{j-1} that contain the endpoints of a_j. Then we remove one of the endpoints of b_j from s to obtain s'. Finally, we let $T_j := T'_{j-1} + s' + a_j$, which is again a spanning hypertree. In this way, we have replaced one of the connections in OPT with an edge from A. Let $B := B_{\ell_p}$. The replacement process defines a bijection $\varphi\colon A \leftrightarrow B$ by $\varphi(a_j) := b_j$. Observe that $\{a_1,\dots,a_j,b_{j+1},\dots,b_{\ell_p}\} = A_j \cup (B \setminus B_j)$ is a spanning tree for all j.

Note that for every edge $b \in B$ there is a unique hyperedge $s \in OPT$ with $b \subseteq s$, because OPT contains no cycle. For a hyperedge $s \in OPT$ we denote the set of edges from B contained in s and defined during the replacement steps $j = \ell_i+1,\dots,\ell_p$ by

$$U_i(s) := \{\, b \in B \setminus B_{\ell_i} \mid b \subseteq s \,\}.$$

Let $u_i(s) := \#U_i(s)$. Then the number of edges from B contained in s which are defined during the replacement steps $j = \ell_{i-1}+1,\dots,\ell_i$ is $u_{i-1}(s) - u_i(s)$.

The above definitions imply that

$$\begin{aligned} |ALG| &= \sum_{i=1}^{p} |t_i| = \sum_{a\in A} |a| = \sum_{b\in B} |\varphi^{-1}(b)| \\ &= \sum_{s\in OPT} \sum_{\substack{b\in B\\ b\subseteq s}} |\varphi^{-1}(b)| \\ &= \sum_{s\in OPT} \sum_{i=1}^{p} \sum_{\substack{b\in B_{\ell_i}\setminus B_{\ell_{i-1}}\\ b\subseteq s}} |\varphi^{-1}(b)| \\ &= \sum_{s\in OPT} \sum_{i=1}^{p} \big(u_{i-1}(s) - u_i(s)\big)\,\frac{|t_i|}{c_{i-1}(t_i)}\,. \end{aligned} \quad (2)$$

We can estimate the inner sum using (1) in the following way.

$$\begin{aligned} &\sum_{i=1}^{p} \big(u_{i-1}(s) - u_i(s)\big)\,\frac{|t_i|}{c_{i-1}(t_i)} \\ \leqslant\ &\sum_{i=1}^{p} \big(u_{i-1}(s) - u_i(s)\big)\left(1+\frac{1}{c_{i-1}(s)}\right)\frac{|s|}{c_0(s)+1} \\ =\ &|s| \cdot \frac{u_0(s) - u_p(s) + \sum_{i=1}^{p} \frac{u_{i-1}(s)-u_i(s)}{c_{i-1}(s)}}{c_0(s)+1}\,. \end{aligned} \quad (3)$$

In the numerator we have $u_0(s) = c_0(s)$ and $u_p(s) = 0$. Note that $A_{\ell_i} \cup U_i(s) \subseteq A_{\ell_i} \cup (B \setminus B_{\ell_i})$ contains no cycles and the edges of $U_i(s)$ are contained in s. Therefore, if we add s to ALG_i, the number of connected components decreases by at least $u_i(s)$. This implies $u_i(s) \leqslant c_i(s)$. Observing that $u_0(s),\dots,u_p(s)$ is a non-increasing sequence of natural numbers,

$$\begin{aligned} \sum_{i=1}^{p} \frac{u_{i-1}(s)-u_i(s)}{c_{i-1}(s)} &\leqslant \sum_{i=1}^{p} \frac{u_{i-1}(s)-u_i(s)}{u_{i-1}(s)} \\ &\leqslant H(u_0(s)) - H(u_p(s)) = H(c_0(s))\,. \end{aligned}$$

We have

$$\frac{c_0(s)+H(c_0(s))}{c_0(s)+1} \leqslant \max_{x\in\mathbb{N}} \frac{x+H(x)}{x+1} = \frac{4+H(4)}{4+1} = \frac{73}{60},$$

so (3) is bounded by $\frac{73}{60}|s|$. Putting things together, we obtain

$$\begin{aligned} |ALG| &= \sum_{s\in OPT} \sum_{i=1}^{p} \big(u_{i-1}(s)-u_i(s)\big) \frac{|t_i|}{c_{i-1}(t_i)} \\ &\leqslant \sum_{s\in OPT} |s| \cdot \frac{73}{60} \\ &= \frac{73}{60} \cdot |OPT|, \end{aligned}$$

as desired. □

4 Tightness

Theorem 4.1
The performance ratio $73/60$ of `Greedy-MSS` *is tight even for unweighted quasi-bipartite Graphs.*

Proof. The worst case instance is shown in Figure 2. All edges have length 1. A Steiner tree using the Steiner vertices for the ellipses has the total length $5 \cdot 12 = 60$, which is in fact optimal. `Greedy-MSS` will always choose a full component that connects as many connected components as possible. Therefore, it may end up with a Steiner tree using the Steiner vertices for the dashed rectangles from top to bottom. Since the total length of this Steiner tree is $3\cdot5+4\cdot4+6\cdot3+12\cdot2 = 73$, the theorem follows. □

A worst case instance for `Greedy-MSS` which is smaller, but requires edge lengths, is given in [8].

5 Running Time

Theorem 5.1
The running time of `Greedy-MSS` *is $O(\#V\#R)$.*

Proof. We need to assume some preprocessing to achieve the running time as claimed. We first compute minimum spanning trees for the connected components of the graph induced by R and remove all other edges between vertices of R. Clearly, this can be done in $O(\#R\#R)$ and does not change the behaviour of our algorithm. Further, by placing extra vertices on edges between terminals, we obtain a bipartite graph $(R \dot\cup S, E)$ where $\#S \leqslant \#V$.

Consider the maximal sets $C \subseteq R$ of terminals which are connected by the current solution in iteration i. For $v \in S$ let $d_i(v)$ be the number of C's that contain a neighbor of v. Then f_i can be minimized in $O(\#S)$ if the values $d_i(v)$ are known for all $v \in S$. As there are at most $\#R$ iterations, the total running time of the algorithm is $O(\#S\#R)$ as claimed.

To efficiently compute the values $d_i(v)$ we proceed as follows. For each C we maintain an ordered list $\lambda(C)$ of its neighbors in S. The values $d_0(v)$ and the initial lists $\lambda(C)$ can be computed in $O(\#S\#R)$.

When several C's are merged in iteration i, we successively merge two of the corresponding lists and decrease the values $d_i(v)$ for each v that appears in both lists. It takes $O(\#S)$ time to merge two lists. In total $\#R-1$ pairs of lists have to be merged. Therefore, updating the values $d_i(v)$ can be done in $O(\#S\#R)$. □

6 Concluding Remarks

All known lower bound reducions for the Steiner tree problem in graphs produce uniformly quasi-bipartite instances. Theorem 3.1 shows that reductions of this type cannot prove lower bounds greater than $73/60$.

Our analysis of `Greedy-MSS` generalizes the analysis of the greedy algorithm for the set cover problem. This can be pushed a bit further to give a combinatorial proof that the greedy algorithm for the minimum spanning subset problem in k-polymatroids has a performance ratio of $H(k-1) = \frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{k-1}$ [2]. We also used this analogy between the greedy algorithms for the Steiner tree problem and for the set cover problem to construct the worst-case instance shown in Figure 2.

It is conceivable that further exploiting this analogy also yields better approximation algorithms for the general case of the Steiner tree problem. Similarly, tight lower bounds for the set cover problem [5] might be used to find improved lower bounds for the Steiner tree problem. Progress in both directions would be very interesting.

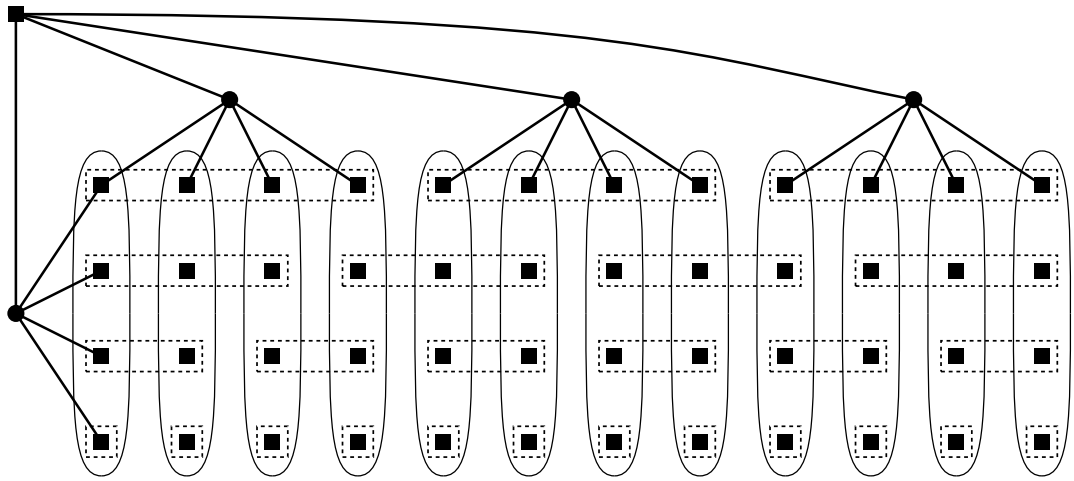

Figure 2: A worst-case instance for `Greedy-MSS`. Square dots are terminals. Round dots are non-terminals. For every ellipse and every dashed rectangle there is a non-terminal which is connected to the terminals inside plus the terminal to the top left.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof verification and hardness of approximation problems*, J. ACM 45 (1998), 501–555.

[2] G. Baudis, C. Gröpl, S. Hougardy, T. Nierhoff, H.J. Prömel, *Approximating minimum spanning sets in hypergraphs and polymatroids*, technical report, Humboldt-Universität zu Berlin, 2000.

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

[4] M. Bern, P. Plassmann, *The Steiner problem with edge lengths 1 and 2*, Inform. Process. Lett. 32 (1989), 171–176.

[5] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, J. ACM 45 (1998), 634–652.

[6] E.N. Gilbert, H.O. Pollak, *Steiner minimal trees*, SIAM J. Appl. Math. 16 (1968), 1–29.

[7] C. Gröpl, S. Hougardy, T. Nierhoff, H.J. Prömel, *Lower bounds for approximation algorithms for the Steiner tree problem*, In: Graph-Theoretic Concepts in Computer Science, WG 2001, Springer LNCS 2204, 217–228.

[8] C. Gröpl, S. Hougardy, T. Nierhoff, H.J. Prömel, *Approximation algorithms for the Steiner tree problem in graphs*, In: X. Cheng und D.-Z. Du (ed.): Steiner Trees in Industry, Kluwer Academic Publishers (2001), 235–279.

[9] S. Hougardy, H.J. Prömel, *A 1.598 approximation algorithm for the Steiner problem in graphs*, In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1999, 448–453.

[10] R.M. Karp, *Reducibility among combinatorial problems*, In: Complexity of Computer Computations, Plenum, New York (1972), 85-103.

[11] M. Karpinski, A. Zelikovsky, *New approximation algorithms for the Steiner tree problems*, J. Comb. Optim. 1 (1997), 47–65.

[12] H.J. Prömel, A. Steger, *A new approximation algorithm for the Steiner tree problem with performance ratio 5/3*, J. Algorithms 36 (2000), 89–101.

[13] S. Rajagopalan, V.V. Vazirani, *On the bidirected cut relaxation for the metric Steiner problem*, In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1999, 742–751.

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

[15] G. Robins, A. Zelikovsky, *Improved Steiner tree approximation in graphs*, In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, 770–779.

[16] M. Thimm, *On the approximability of the Steiner tree problem*, In: Mathematical Foundations of Computer

Science 2001, MFCS 2001, Springer LNCS 2136, 678–689.

[17] L.A. Wolsey, *An analysis of the greedy algorithm for the submodular set covering problem*, Combinatorica 2 (1982), 385–393.

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

[19] A. Zelikovsky, *Better approximation bounds for the network and Euclidean Steiner tree problems*, technical report CS-96-06, University of Virginia, 1996.