Lower Bounds for Approximation Algorithms for the Steiner Tree Problem

Clemens Gröpl, Stefan Hougardy, Till Nierhoff, and Hans Jürgen Prömel

Humboldt-Universität zu Berlin
Institut für Informatik
10099 Berlin
`{groepl,hougardy,nierhoff,proemel}@informatik.hu-berlin.de`

Abstract. The Steiner tree problem asks for a shortest subgraph connecting a given set of terminals in a graph. It is known to be APX-complete, which means that no polynomial time approximation scheme can exist for this problem, unless P=NP. Currently, the best approximation algorithm for the Steiner tree problem has a performance ratio of 1.55, whereas the corresponding lower bound is smaller than 1.01. In this paper, we provide for several Steiner tree approximation algorithms lower bounds on their performance ratio that are much larger. For two algorithms that solve the Steiner tree problem on quasi-bipartite instances, we even prove lower bounds that match the upper bounds. Quasi-bipartite instances are of special interest, as currently all known lower bound reductions for the Steiner tree problem in graphs produce such instances.

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. This problem is a classical NP-hard problem [10] and even worse it is also known to be APX-complete, i.e., there exists some constant $c > 1$ such that no polynomial time approximation algorithm for this problem can have a performance ratio smaller than c, unless P=NP. (The performance ratio of an approximation algorithm is the largest ratio between the length of a solution found by the algorithm and the optimal length.)

The Steiner tree problem appears in many different applications, as for example in VLSI-design, in the design of telecommunication networks, and in the reconstruction of phylogenetic trees in biology. Therefore it is an important task to find good approximation algorithms for this problem and, eventually, to prove that no better approximation algorithms can be possible, unless P=NP.

Currently, the best approximation algorithm for the Steiner tree problem is due to Robins and Zelikovsky [7] and has a performance ratio of $1 + \frac{1}{2} \ln 3 < 1.550$. On the other hand, the largest known lower bound for the approximability of the Steiner tree problem has a value not exceeding 1.01 [4, 12].

This large gap between lower and upper bounds suggests that there might be still some room for improvements on both sides. There exist several approximation algorithms for the Steiner tree problem in graphs that achieve a performance ratio less

	approximation algorithm	known upper bound	our lower bound
general graphs	Relative Greedy Algorithm	1.694	1.333
	Loss Contracting Algorithm	1.550	1.200
quasi-bipartite graphs	Iterated 1-Steiner Heuristic	1.500	1.500
	Loss Contracting Algorithm	1.279	1.195
	Greedy-MSS	$1.21\overline{6}$	$1.21\overline{6}$

Table 1. Summary of results

than 2 [6]. But for none of these algorithms it is known whether their analysis is tight. Therefore it is not clear which of these algorithms actually achieves the best performance ratio.

An instance of the Steiner tree problem is *quasi-bipartite* if the set $V \setminus R$ is independent. Quasi-bipartite graphs are an important special case as the instances resulting from lower bound reductions [4, 12] have this form. For these graphs there exist several approximation algorithms for which better performance ratios have been proved than in the general case.

In this paper, we provide for several Steiner tree approximation algorithms lower bounds on their performance ratio. For two algorithms that solve the Steiner tree problem on quasi-bipartite instances, we even prove lower bounds that match the upper bounds. Such lower bound results, even if they are not tight, allow to estimate the quality of a given performance analysis and can provide ideas on how to improve it. Moreover, these lower bounds give good examples of instances on which every Steiner tree approximation algorithm must perform well, and thus might yield to better approximation algorithms for the Steiner tree problem.

All algorithms considered in this paper are based on a general greedy framework which we describe in Section 2. In Sections 3 and 4 we present lower bounds for two approximation algorithms for the Steiner tree problem in general graphs. Lower bounds for approximation algorithms for the quasi-bipartite case are given in Sections 4, 5 and 6. For two of these algorithms we even provide tight lower bounds. Our results are summarized in Table 1.

2   Notations and a General Framework for Greedy Algorithms

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. We denote a Steiner minimum tree by SMT and its length by smt. A Steiner tree usually contains not only vertices from R but also from $V \setminus R$. These vertices are called *Steiner vertices*. 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. Therefore, a Steiner tree is a collection of full components which is connected and covers R.

$\{ FC$ is a set of full components $\}$
$i \leftarrow 0$
While an improving full component exists:
    Find $T_{i+1} \in FC$ that minimizes f_i
    $i \leftarrow i + 1$
$i_{\max} \leftarrow i$
Output Steiner tree using $T_1, \ldots, T_{i_{\max}}$

Fig. 1. A general framework for greedy algorithms.

Most of the known approximation algorithms for the Steiner tree problem in graphs fit into the general framework shown in Figure 1. Let FC be a set of full components. The algorithm chooses the $(i + 1)$-st full component $T_{i+1} \in FC$ such that a certain function $f_i \colon FC \to \mathbb{R}_+$ is minimized. The algorithm stops when no further improvement is possible. The precise meaning of this condition depends on the function f_i. Finally a Steiner tree is output that uses the full components that have been selected in the while-loop.

3 Relative Greedy Algorithm

In this section we consider Zelikovsky's relative greedy algorithm [13], which achieves a performance ratio of less than 1.694. This algorithms fits into the general greedy framework for Steiner tree approximation algorithms as follows. Let $G = (V, E)$ be a graph with a weight function on the edges and $R \subseteq V$ be the set of terminals. Fix some constant $k \in \mathbb{N}$ and define FC to be the set of all full components with at most k terminals. The weight of a full component $X \in FC$ is simply its length and will be denoted by $|X|$. Note that FC can be computed in polynomial time.

From G one can easily derive the so called *terminal distance graph* G' which is a complete graph on the set R where the weight of an edge is the length of a shortest path in G connecting the two endpoints of this edge. If $T_1, \ldots T_i$ are some full components in FC then $MST(R/T_1 \ldots T_i)$ denotes a minimum spanning tree in the graph that is obtained from G' by contracting each of the full components $T_1, \ldots, T_i$. Its length is denoted by $mst(R/T_1 \ldots T_i)$.

Now the function f_i on a full component $X \in FC$ is defined as follows:

$$f_i(X) := \frac{|X|}{mst(R/T_1 \ldots T_i) - mst(R/T_1 \ldots T_i X)} .$$

The while-loop of the algorithm will be executed, as long as there exists some full component $X \in FC$ with $f_i(X) \leq 1$. Zelikovsky proved that on termination of the relative greedy algorithm the following inequality holds:

$$|T_1| + \ldots + |T_{i_{\max}}| \leq (1 + \ln 2 + \varepsilon) \cdot smt .$$

If ε is chosen small enough this value is smaller than 1.694 .
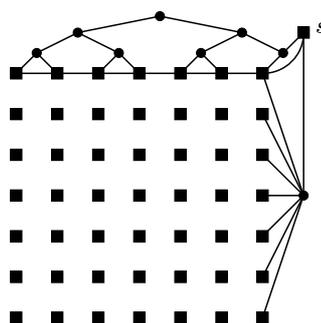
Fig. 2. Part of the instance F_3.

Theorem 1. *The performance ratio of the relative greedy algorithm is at least* $4/3$.

Proof. We will construct a family of instances F_i for the Steiner tree problem and prove that the ratio between the length of a Steiner minimum tree and the length of a Steiner tree that is found by the relative greedy algorithm tends to $4/3$ as $i \to \infty$.

The instance F_i has $(2^i - 1)(2^i - 1) + 1$ terminals which are arranged in a $(2^i - 1) \times (2^i - 1)$-grid with one extra terminal s. We assume that s belongs to every row and every column of the $(2^i - 1) \times (2^i - 1)$-grid, i.e., we can think of s as the last row and last column of a $2^i \times 2^i$-grid. For each row, we have a binary tree, called the *row tree*, which has depth i and the 2^i terminals of a row as its leaves. For each column we have a star, called the *column star*, which has the 2^i terminals of a column as its endpoints. No two row trees or column stars have a vertex in common, except for terminals. Thus, in total the instance F_i has $(2^i - 1)(2^i - 1) + (2^i - 1) = 2^i(2^i - 1)$ non-terminal vertices. Finally, we connect any two consecutive terminals in a row by an edge. In Figure 2 one row tree and one column star and the edges between any two consecutive terminals in one row of the instance F_3 are shown.

We now have to specify the weights for the edges of the instance F_i. For all row trees all edges within one level have the same weight. The edges of the first and second level of the binary tree, i.e., the six edges incident to the root of the binary tree and its two children, have weight $1/4$. The 2^i edges of the i-th level of the binary tree, i.e., the edges incident to the terminals have weight 1. All other edges in the binary tree have weight $1/2$. The total weight of all row trees is therefore

$$(2^i - 1)(2^i \cdot 1 + (2^i - 8) \cdot \frac{1}{2} + 6 \cdot \frac{1}{4}) = (2^i - 1)(3 \cdot 2^{i-1} - \frac{5}{2}).$$

Next we specify the weights for the edges of the column stars. All edges within one column star have the same weight, which depends on the column. The columns are numbered from left to right by $1, 2, \dots, 2^i - 1$. We number a column star with a value that is obtained from the binary representation of the column number read from right to left. E.g., for $i = 3$ the binary representation of the coulmn numbers is $001, 010, 011, 100, 101, 110, 111$ and these numbers read from right to left give $100, 010, 110, 001, 101, 011, 111$. Thus the column stars are numbered $4, 2, 6, 1, 5, 3, 7$ from left to right.

i	lower bound	edge weights (levelwise)	lower bound	edge weights (levelwise)
3	$\frac{47}{38} \doteq 1.2368$	$1, \frac{1}{4}, \frac{1}{4}$	1.2895	$1, \frac{1}{4}, \frac{1}{4}$
4	$\frac{225}{172} \doteq 1.3081$	$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}$	1.3151	$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}$
5	$\frac{965}{728} \doteq 1.3255$	$1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}$	1.3290	$1, \frac{1}{2}, \frac{3}{8}, \frac{1}{4}, \frac{1}{4}$
6	$\frac{3981}{2992} \doteq 1.3305$	$1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}$		

Table 2. Small lower bound instances for the relative greedy algorithm.

All edges in the column stars with numbers 1 to $2^{i-1} + 2$ have the same weight. We define it as $(2 - 2^{1-i})(1 - 3 \cdot 2^{-i-1})$. The edges in all other column stars get the weight $2 - 2^{1-i}$. As each column star has 2^i edges, the total weight of all column stars is

$$2^i \cdot (2 - 2^{1-i}) \left((2^{i-1} + 2)(1 - 3 \cdot 2^{-i-1}) + (2^{i-1} - 3) \right) \; = \; 2 \cdot (2^i - 1) \cdot (2^i - \frac{7}{4} - \frac{3}{2^i}).$$

Finally we have to specify the edges connecting two consecutive terminals in a row. They are all defined as $(2 - 3 \cdot 2^{-i})(2 - 2^{1-i})$.

Note that the set of all row trees is a solution to the Steiner tree problem (in fact, it is easily seen that it is the optimum solution for the instances F_i). We claim that the relative greedy algorithm started with $k = 2^i$ will return the set of all column stars as solution. Therefore the performance ratio of the relative greedy algorithm is at least

$$\frac{2 \cdot (2^i - 1) \cdot (2^i - \frac{7}{4} - \frac{3}{2^i})}{(2^i - 1)(3 \cdot 2^{i-1} - \frac{5}{2})}.$$

This expression tends to $4/3$ as i goes to infinity, which proves the statement of the theorem.

To prove that the relative greedy algorithm indeed returns the set of column stars as solution we assume that it selects as the j-th full components the column star numbered with j, i.e., we assume that the ties between the weights of the column stars are broken exactly in this way. By adding a weight of $j \cdot \varepsilon$ to the j-th column star one could break all the ties, but this would make the calculations more complicated. It is now a straightforward but quite tedious calculation to show that the cost function of the relative greedy algorithm is minimized in the j-th step by the column star numberd with j. Details of this calculation are omitted in this version of the paper due to space restrictions. $\square$

In column two of Table 2 the lower bounds that are obtained by the construction described in the proof of Theorem 1 are given. The third column contains the weights of the edges of the row trees. The construction given in the proof is not optimal as is indicated by columns four and five of this table. The lower bounds can be improved slightly by choosing better weights for the edges of the column stars and row trees. For
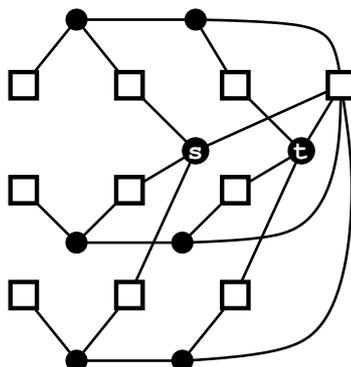
Fig. 3. A 1.2 lower bound instance for the loss contracting algorithm. All edges of the row trees have weight 5. The edges incident to s and t have weights 9 and 6, respectively.

$i = 3, 4, 5$ we computed such optimal weights and obtained the lower bounds given in column four of Table 2. We were not able to calculate a general formula for the lower bound of these instances, but it seems to be the case that the lower bounds also converge to $4/3$.

4   Loss Contracting Algorithm

Currently the best approximation algorithm for the Steiner tree problem in graphs the loss contracting algorithm of Robins and Zelikovsky [7] which has a performance ratio of at most $1 + \frac{\ln 3}{2} < 1.550$. The loss contracting algorithm is very similar to the relative greedy algorithm, but instead of contracting a selected full component T entirely, it contracts only a subset $Loss(T)$ of its edges in order to achieve the same effect at a lower price. The *loss* of a full component T is a minimum length forest contained in T such that every Steiner vertex of T is connected to a terminal. Its length is denoted by $loss(T)$. The contracted parts are connected by a minimum spanning tree using edges from the terminal distance graph. We denote its length by $m(\cdot) := mst(R/Loss(\cdot))$. Clearly, the length of the Steiner tree corresponding to the full components $T_1, \ldots, T_i$ is $loss(T_1, \ldots, T_i) + m(T_1, \ldots, T_i)$.

The loss contracting algorithm fits into the framework of Figure 1. In each step, the loss contracting algorithm chooses a full component that minimizes the function

$$f_i(T) := \frac{loss(T)}{m(T_1 \ldots T_i) - m(T_1 \ldots T_i T)} \, .$$

It stops when there are no more improving full components, i. e., $\min_{T \in FC} f_i(T) \geq 1$.
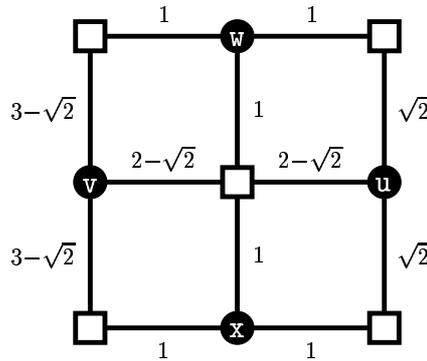
Fig. 4. A quasi-bipartite 1.195 lower bound instance for the loss contracting algorithm.

Theorem 2. *The performance ratio of the loss contracting algorithm is at least* 1.2.

Proof. We use a construction similar to the one used in the proof of Theorem 1. Unfortunately, this construction breaks down for grids of size larger than 4×4. We explain the best weight assignment for a 4×4-grid as shown in Figure 3. The edges of the row trees have weight 5. The edges incident to s and t have weights 9 and 6, respectively. We denote the full components containing s and t by T_s and T_t. The optimal Steiner tree uses the row trees and has length 75.

Notice that the edges in $MST(R)$ connecting the vertices of the upper row of terminals have lengths 10, 15, and 10 from left to right. For each row tree, the *loss* is 10 and f_0 is $10/\big(3 \cdot 35 - (2 \cdot 35 + 15)\big) = 0.5$. One can easily check that subtrees of the row trees are worse. Since $f_0(T_s) = 9/\big(3 \cdot (35 - (20 + 9))\big) = 0.5$ and $f_0(T_t) = 6/\big(3 \cdot (35 - (25 + 6))\big) = 0.5$, the algorithm is allowed to choose T_s. (Choosing T_s could be forced by perturbing the edge weights.) In the second step, the value of f_1 for each (complete) row tree is $10/\big(3 \cdot 29 - (2 \cdot 29 + 15)\big) = 10/14 > 0.7$, and $f_1(T_t) = 6/\big(3 \cdot (29 - (10 + 9 + 6))\big) = 0.5$. Therefore T_t is chosen. Now the value of f_2 for each row tree is $10/\big(3 \cdot 25 - (2 \cdot 25 + 15)\big) = 1$, and the algorithm stops with a solution of length $loss(T_s, T_t) + m(T_s, T_t) = (9 + 6) + 3 \cdot (10 + 9 + 6) = 90$. We get a lower bound of $90/75 = 1.2$. □

Robins and Zelikovsky [7] also showed that the performance ratio of the loss contracting algorithm is at most 1.279 if the graph is quasi-bipartite. A quasi-bipartite lower bound instance for the loss contracting algorithm is shown in Figure 4. It is only slightly weaker than the one shown in Figure 3 and comes fairly close to the performance ratio proved in [7].

Theorem 3. *The performance ratio of the loss contracting algorithm in quasi-bipartite graphs is at least* $(5 - \sqrt{2})/3 > 1.195$.

Proof. The quasi-bipartite lower bound instance is shown in Figure 4. It is based on the observation that *one* full component chosen by the algorithm can 'destroy the benefit'

of *two* full components of the optimal solution. The Steiner minimum tree consists of the full components T_w and T_x and has length 6. $MST(R)$ uses four diagonal edges and has length 8.

At the beginning, we have $f_0(T_u) = \left(2 - \sqrt{2}\right)/\left(8 - (2 \cdot 2 + 2 \cdot \sqrt{2})\right) = 0.5$ and $f_0(T_v) = \left(2 - \sqrt{2}\right)/\left(8 - (2 \cdot 2 + 2 \cdot (3 - \sqrt{2}))\right) = 1/\sqrt{2} > 0.5$, whereas $f_0(T_w) = f_0(T_x) = 1/\left(8 - (2 \cdot 2 + 2 \cdot 1)\right) = 0.5$. Therefore we may assume that the loss contracting algorithm decides for T_u. $MST(R/Loss(T_u))$ has the same form as $MST(R)$, but the two edges to the right have shrunk from 2 to $\sqrt{2}$. Therefore $m(T_u) = 4 + 2\sqrt{2}$.

In the next step, we have $f_1(T_v) = \left(2 - \sqrt{2}\right)/\left((4 + 2\sqrt{2}) - (2 \cdot (3 - \sqrt{2}) + 2 \cdot \sqrt{2})\right) = 1/\sqrt{2}$ and $f_1(T_w) = f_1(T_x) = 1/\left((4 + 2\sqrt{2}) - (1 + 1 + 2 + \sqrt{2})\right) = 1/\sqrt{2}$. We may assume that the loss contracting algorithm chooses T_v. The edges on the left hand side of $MST(R/Loss(T_u, T_v))$ have shrunk from 2 to $3 - \sqrt{2}$. Now we have $m(T_u, T_v) = 2 \cdot (3 - \sqrt{2}) + 2 \cdot \sqrt{2} = 6$.

As $f_2(T_w) = f_2(T_x) = 1/\left(6 - (1 + 1 + (3 - \sqrt{2}) + \sqrt{2})\right) = 1$, no further improvement is possible and the algorithm will stop at this point without using T_w and T_x. The length of the Steiner tree found by the loss contracting algorithm is $|T_u| + |T_v| = 10 - 2\sqrt{2}$. The lower bound follows. $\qquad\square$

5   Iterated 1-Steiner Heuristic

The iterated 1-Steiner heuristic is a simple greedy local search heuristic. Recall that the Steiner minimum tree for a set of required vertices R can be reconstructed from its set of Steiner vertices I as $SMT(R) = MST(R \cup I)$. Starting from a spanning tree for the terminal set, i.e. $I = \emptyset$, the iterated 1-Steiner heuristic adds in each step a single Steiner vertex to I if this will reduce the size of the MST. The algorithm stops if no such improvement is possible. A vertex s is accepted if

$$f_i(s) := mst(R \cup \{s_1, \ldots, s_i, s\}) - mst(R \cup \{s_1, \ldots, s_i\}) < 0 \,.$$

Since iterated 1-Steiner starts from a spanning tree in the terminal distance graph and $mst(R) \leq 2 \cdot smt(R)$, it is clear that its solution is never more than a factor of 2 away from the optimum. No better performance ratio has been proven. Minoux [11] showed how to accelerate iterated 1-Steiner in quasi-bipartite graphs, but did not improve the performance ratio.

Rajagopalan and Vazirani gave a $3/2 + \varepsilon$ approximation algorithm for the Steiner tree problem in quasi-bipartite graphs which is based on the primal-dual method. They pointed out that approximation algorithms for the Steiner tree problem in quasi-bipartite graphs are a significant step towards better approximation algorithms for general instances. By now, the best approximation ratio known for the quasi-bipartite case is 1.279 and belongs to the loss contracting algorithm of Robins and Zelikovsky [7]. In the same paper, they also showed that already the simple iterated 1-Steiner heuristic achieves an approximation ratio of $3/2$ in quasi-bipartite graphs. Here we give a family of instances for which this ratio is asymptotically attained. This shows that the analysis of iterated 1-Steiner in quasi-bipartite graphs in [7] is tight. Interestingly, their result does not rely on a greedy (locally optimal) choice of the next Steiner vertex which is
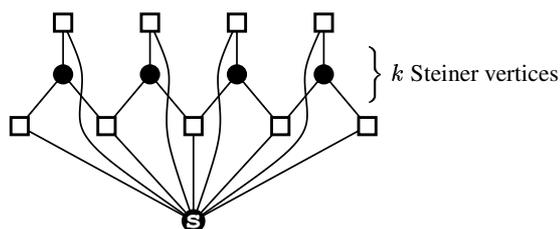
Fig. 5. A simple lower bound instance for iterated 1-Steiner.
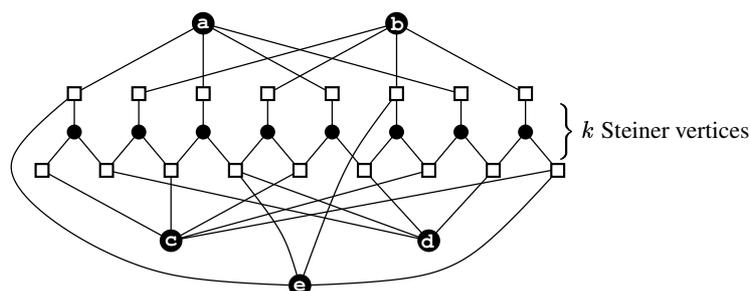
Fig. 6. A lower bound instance for iterated 1-Steiner with vertex removal.

included into the set of Steiner vertices I. It is not necessary to find a vertex s that minimizes $f_i(s)$; non-positivity suffices.

A slight improvement of the iterated 1-Steiner heuristic is to remove Steiner vertices which have degree one or two in $MST(R \cup I)$. (These may result from changes in the topology during the course of the algorithm.) Steiner vertices of degree one can only add to the length of the minimum spanning tree without connecting a terminal, and Steiner vertices of degree two can be removed safely because a shortcut is present in the terminal distance graph due to the triangle inequality.

Theorem 4. *The performance ratio of the iterated 1-Steiner heuristic in quasi-bipartite graphs is $3/2$ (even if Steiner vertices of degree 1 or 2 are removed).*

Proof. A bad instance for the iterated 1-Steiner heuristic without vertex removal is shown in Figure 5. All edges have unit weight. Assume that the algorithm has already chosen the k Steiner vertices of degree 3 in the row (each of them reduced the value of $mst(R \cup I)$ by one when it was added to I). Then the iterated 1-Steiner heuristic will stop at this point without including s, because this would increase the length of the minimum spanning tree from $3k$ to $2k + 1 + k$. The Steiner minimum tree, however, consists of the star around s. We get a lower bound of $\frac{3k}{2k+1}$, which converges to $\frac{3}{2}$ as $k \to \infty$.

The instance from Figure 5 does not work for the iterated 1-Steiner heuristic with vertex removal, since it will remove all the Steiner vertices from I and keep only s

for the final solution. Consider Figure 6 instead. Again we assume that the set I consists of the Steiner vertices of degree 3. We have split the vertex s from Figure 5 into four vertices a, b, c, and d, each of degree $\sim \frac{k}{2}$. Another vertex e of degree four is connected to one terminal from each of these stars around a, b, c, and d. The connection points are spread out in such a way that for each $u \in \{a, b, c, d, e\}$, no vertex in I is adjacent to more than one terminal from the star around u. This implies that no vertex of I can have degree one in $MST(R \cup I \cup \{u\})$ and consequently, $mst(R \cup I \cup \{u\}) = mst(R \cup I) + 1$. Therefore iterated 1-Steiner with vertex removal will stop at this point. The resulting lower bound is $\frac{3k}{2k+5} \to \frac{3}{2}$. □

6 Greedy-MSS

Recently, a new approximation algorithm for the Steiner tree problem in quasi-bipartite graphs was presented in [5]. The analysis of this algorithm uses a matroid-style exchange argument, and uses ideas from the analysis of the greedy algorithm for set cover. (In fact, it can even be extended to the minimum spanning set problem in polymatroids [3].)

The resulting algorithm Greedy-MSS [5] achieves a performance ratio of 1.217 for the Steiner tree problem in quasi-bipartite graphs where all edges incident to a Steiner vertex have the same weight. Such instances are called *uniformly quasi-bipartite*. Previously the best perfomance ratio for this class of instances was 1.279 and due to the loss contracting algorithm of [7]. (See also Section 4.)

In quasi-bipartite instances of the Steiner tree problem in graphs, every full component contains a unique Steiner vertex. Algorithm Greedy-MSS tries to minimize the length-per-connection ratio greedily, i. e., we have

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

in the general greedy framework (Fig. 1). Here $|T|$ denotes the length of the full component T and $c_i(T)$ is defined as the difference between the number of components before and after T is added to the subhypergraph $(R, \{T_1, \ldots, T_i\})$. The algorithm stops when $(R, \{T_1, \ldots, T_p\})$ is a spanning subhypergraph. While the number of full components around a particular Steiner vertex which might be added in step i can in general be exponential, it is easy to see that Greedy-MSS will always choose one that yields as many new connections as possible among them, and due to the uniformity condition all of these full components have the same weight. Therefore, the algorithm only has to evaluate one full component for each Steiner vertex in each phase. Greedy-MSS can be implemented to run in $O(\#V \#R)$ time.

Theorem 5. *The performance ratio of algorithm Greedy-MSS is* $73/60 = 1.21\overline{6}$.

Proof. The upper bound on the performance ratio was proved in [5]. Here we give an instance for which the performance ratio $73/60$ of Greedy-MSS is attained.

The worst case instance is shown in Figure 7. All edges have length 1. The terminals $\{r_{i,j} \mid 1 \le i \le 4, 1 \le j \le 12\}$ are arranged in form of a grid. (For notational
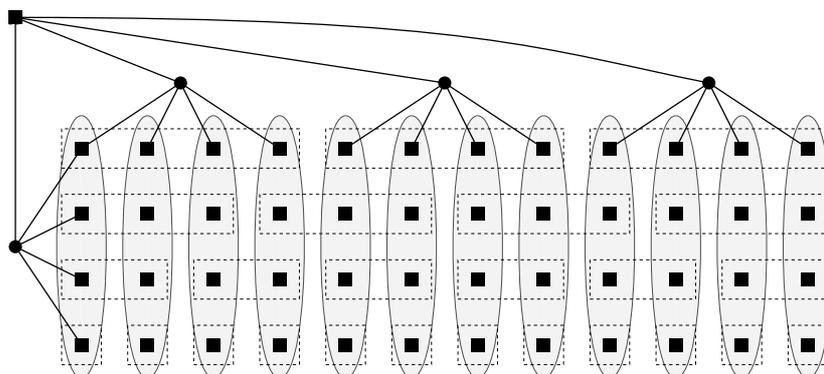
Fig. 7. A worst-case instance for Greedy-MSS.

convenience we will number the rows from bottom to top.) We also have a terminal r_0 which is shown in the top left corner. The set of potential Steiner vertices is $X \cup Y$. The set X consists of the vertices $\{x_1, \ldots, x_{12}\}$, where x_j is connected to $r_{1,j}, \ldots, r_{4,j}$ and r_0. A Steiner tree using the Steiner vertex set X has total length $5 \cdot 12 = 60$, which is in fact optimal. The set Y consists of vertices $y_{i,j}$ which are connected to r_0 and a subset of terminals in a row, as indicated by the dashed rectangles. Formally, we have $Y = \{ y_{i,j} \mid 1 \leq i \leq 4,\ 1 \leq j \leq 12/i \}$, and $y_{i,j}$ is connected to $r_{i,i(j-1)+1}, \ldots, r_{i,ij}$ and r_0. Next we show that Greedy-MSS may end up with a Steiner tree using the Steiner vertex set Y. Since the total length of this Steiner tree is $3 \cdot 5 + 4 \cdot 4 + 6 \cdot 3 + 12 \cdot 2 = 73$, the theorem follows.

Note that in the unweighted quasi-bipartite case, $f_i(T) = |T|/(|T| - 1)$ for every full component T that does not create a cycle. Therefore Greedy-MSS will always choose a full component that connects as many connected components as possible. Assume that the Steiner vertices $y_{4,1}, \ldots, y_{4,3}$ have already been chosen. Then the full components around the Steiner vertices in X have been 'nibbled off' such that the size of any largest full component that does not create a cycle has dropped down to 4. Therefore Greedy-MSS might continue choosing the Steiner vertices $y_{3,1}, \ldots, y_{3,4}$. Going forth in this way, we arrive at Y and stop since the graph is connected. $\square$

A smaller worst case instance for Greedy-MSS was given in[5]. It was inspired by our proof of the performance ratio of Greedy-MSS, but requires edge weights.

References

1. 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.
2. A. Borchers and D.-Z. Du, *The k-Steiner ratio in graphs*, SIAM J. Computing 26 (1997), 857–869.

3. 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, Institut für Informatik, 2000.

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

5. C. Gröpl, S. Hougardy, T. Nierhoff, H. J. Prömel, *Steiner trees in quasi-bipartite graphs*, Technical report, Humboldt-Universität zu Berlin, Institut für Informatik, 2000.

6. C. Gröpl, S. Hougardy, T. Nierhoff, H.J. Prömel, *Approximation algorithms for the Steiner tree problem in graphs*, technical report, Humboldt-Universität zu Berlin, 2000. To appear in: D.-Z. Du, X. Cheng, Steiner Trees in Industries, Kluwer Academic Publishers, Dordrecht.

7. G. Robins, A. Zelikovsky, *Improved Steiner tree approximation in graphs*, In Proc. Symposium on Discrete Algorithms, 770–779, 2000.

8. J. Håstad, *Some optimal inapproximability results*, manuscript, 1996.

9. J. Håstad, *On bounded occurence constraint satisfaction*, Information Processing Letters 74 (2000) 1–6.

10. R.M. Karp, *Reducibility among combinatorial problems*, In: Complexity of Computer Computations, (Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972). New York: Plenum 1972, pp. 85-103.

11. M. Minoux, *Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity*, INFOR 28 (1990), 221–233.

12. M. Thimm, *On the approximability of the Steiner tree problem*, manuscript, Humboldt-Universität zu Berlin, August 2000.

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