Computing Optimal Steiner Trees in Polynomial Space*

Fedor V. Fomin† Fabrizio Grandoni‡ Dieter Kratsch§ Daniel Lokshtanov¶
Saket Saurabh‖

Abstract

Given an n-node edge-weighted graph and a subset of k terminal nodes, the NP-hard (weighted) Steiner tree problem is to compute a minimum-weight tree which spans the terminals. All the known algorithms for this problem which improve on trivial $O(1.62^n)$-time enumeration are based on dynamic programming, and require exponential space.

Motivated by the fact that exponential-space algorithms are typically impractical, in this paper we address the problem of designing faster polynomial-space algorithms. Our first contribution is a simple $O((27/4)^k n^{O(\log k)})$-time polynomial-space algorithm for the problem. This algorithm is based on a variant of the classical tree-separator theorem: every Steiner tree has a node whose removal partitions the tree in two forests, containing at most $2k/3$ terminals each. Exploiting separators of logarithmic size which evenly partition the terminals, we are able to reduce the running time to $O(4^k n^{O(\log^2 k)})$. This improves on trivial enumeration for roughly $k < n/3$, which covers most of the cases of practical interest. Combining the latter algorithm (for small k) with trivial enumeration (for large k) we obtain a $O(1.59^n)$-time polynomial-space algorithm for the weighted Steiner tree problem.

As a second contribution of this paper, we present a $O(1.55^n)$-time polynomial-space algorithm for the cardinality version of the problem, where all edge weights are one. This result is based on a improved branching strategy. The refined branching is based on a charging mechanism which shows that, for large values of k, convenient local configurations of terminals and non-terminals exist. The analysis of the algorithm relies on the Measure & Conquer approach: the non-standard measure used here is a linear combination of the number of nodes and number of non-terminals. Using a recent result in [Nederlof'09], the running time can be reduced to $O(1.36^n)$. The previous best algorithm for the cardinality case runs in $O(1.42^n)$ time and exponential space.

Keywords: Steiner tree, exact algorithms, space complexity.

1 Introduction

The *Steiner tree problem* is one of the best-studied problems in Computer Science. Given a connected graph $G = (V, E)$ on $n = |V|$ nodes, edge weights $w : E \to \mathbb{R}^+$, and a set $T \subseteq V$ of k *terminals*, the objective is finding a subtree ST of G spanning T such that the weight

*A preliminary version of this paper appeared in ESA'08 [19].

†Department of Informatics, University of Bergen, N-5020 Bergen, Norway, fomin@ii.uib.no
‡IDSIA, University of Italian Switzerland, Galleria 1, 6928, Manno, Switzerland, fabrizio@idsia.ch
§LITA, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France, kratsch@univ-metz.fr
¶Department of Informatics, University of Bergen, N-5020 Bergen, Norway, daniello@ii.uib.no
‖The Institute of Mathematical Sciences, Taramani, Chennai 600 113, saket@imsc.res.in

$w(ST) := \sum_{e \in ST} w(e)$ of ST is minimized. In the *cardinality* version of the problem, all edge weights are one. Steiner trees are important in various applications such as VLSI routings [32], phylogenetic tree reconstruction [31] and network routing [34]. We refer to the book of Prömel and Steger [38] for an overview of the results and applications of the Steiner tree problem.

The Steiner tree problem is known to be NP-hard [26]. Furthermore, it is APX-complete, even when the graph is complete and all edge costs are either 1 or 2 [3]. Finding the best (polynomial-time) approximation algorithm for the Steiner tree problem has been a challenge and many papers have been written on this subject. The currently best approximation algorithm, due to Byrka, Grandoni, Rothvoß, and Sanità, has approximation ratio $\ln 4 + \varepsilon < 1.39$ [8]. Robins and Zelikovsky [39] establish an approximation ratio of 1.28 for complete graphs with edge costs 1 or 2. The Steiner tree problem remains NP-hard for Euclidean and rectilinear metrics [25]. On the positive side, Arora established polynomial-time approximation schemes for those two important variants of the problem [1].

The Steiner tree problem plays a crucial role also in parameterized algorithms [10, 15, 37]. The aim here is designing the fastest possible algorithm under the natural assumption that $k \ll n$. In particular, an algorithm for the Steiner tree problem is *fixed-paramer-tractable* (FPT) if its running time is $O(\tau(k)n^{O(1)}) = O^*(\tau(k))$, where $\tau(k)$ is a function of k only[1]. For more than 30 years the fastest FPT algorithm for the Steiner Tree problem was the classical $O^*(3^k)$ dynamic programming algorithm by Dreyfus and Wagner [11]. Dreyfus-Wagner's algorithm is still a popular algorithm used for solving different variants of the problem in practice [9, 24]. This algorithm and its variations are also used as a subroutine in many other algorithms. For example, recent applications of it can be found in FPT algorithms for certain vertex cover problems [29] and for near-perfect phylogenetic tree reconstruction [6]. Recent progress in parameterized complexity and exact algorithms led to new insights on the Steiner tree problem. Mölle, Richter, and Rossmanith [35] (see also [23]) improved the running time to $O^*((2+\epsilon)^k)$, for any constant $\epsilon > 0$. More recently, Björklund, Husfeldt, Kaski, and Koivisto [5] obtained an $O^*(2^k)$ time algorithm for the cardianility version of the problem. This result can be easily generalized to the case of polynomially-bounded integral weights by splitting edges. All the mentioned algorithms are based on a dynamic programming approach: they store useful auxiliary information for every subset of the terminal set, and thus use exponential space $\Omega(2^k)$.

For arbitrary values of k, the fastest known $O^*(1.4143^n)$-time (exponential-space) algorithm for the weighted Steiner tree problem is obtained by combining the algorithm by Mölle et al. [35] (for small k) with trivial enumeration (for large k). This is (essentially) also the fastest algorithm for the cardinality version of the problem.

Exponential-space versus polynomial-space. The situation with exact algorithms for the Steiner tree problem is quite typical for a number of other NP-hard problems: the best exponential time complexity is achieved by algorithms with exponential space complexity [40]. However, algorithms with very high space complexity are unlikely to be fast in practice, especially when external memory accesses are frequent (which is likely the case, due to the huge space usage). This kind of phenomena is not captured by the standard RAM model. Hence it makes sense to search for algorithms with low memory requirements, even

[1]Throughout this paper we use the O^* notation which suppresses polynomial factors: for any polynomial $p(n)$, $O(p(n)T)$ is $O^*(T)$.

if they are asymptotically slower than their exponential-space counterpart. Polynomial-space exact algorithms have been studied for various NP-hard problems, among them Hamiltonian Path [2, 30, 33] and Coloring [4].

For $k = \omega(\log n)$, the existing parameterized algorithms for the Steiner tree problem are not polynomial-space. Under that assumption, the fastest known polynomial-space algorithm is the (almost) trivial enumerative algorithm, based on the following observation. Since all the leaves of any optimal Steiner tree are terminals, the number of Steiner nodes T' of degree 3 or larger is at most k. Given T', the Steiner tree problem is equivalent to the minimum spanning tree problem on $G_M[T \cup T']$, where G_M is the metric closure of G. Such problem can be solved in polynomial time. Hence it is sufficient to list all the subsets $T' \subseteq N := V \setminus T$ of size at most k, and then apply the observation above. This takes time $O(\sum_{i=0}^{k} \binom{n-k}{i} n^{O(1)})$. This running time is $O^*(n^k)$ and $O^*(1.6181^n)$.

Our Results and Techniques. Motivated by the practical limitations of exponential-space algorithms and by the theoretical interest of the topic itself, in this paper we address the problem of designing faster polynomial-space exact algorithms for the Steiner tree problem. We obtained the following results.

• We describe a new, easy-to-implement, (weighted) Steiner tree algorithm, taking time $O^*((27/4)^k n^{O(\log k)})$ and polynomial space. Our algorithm is based on a simple variant of the classical tree-separator theorem: there is a node s in every Steiner tree whose removal partitions the tree in two forests, containing at most $2k/3$ terminals each. This is exploited in a top-down recursive implementation of the classical algorithm by Dreyfus and Wagner (where no partial solution is stored to keep the space complexity polynomial). In more detail, guessing s and the partition (T_1, T_2) of $T \setminus \{s\}$, one obtains two Steiner tree problems on terminal sets $T_1 \cup \{s\}$ and $T_2 \cup \{s\}$, which can be solved recursively.

We remark that our algorithm is not FPT, but *quasi-FPT*, that is, its running time is of the form $O^*(\tau(k)n^{polylog(k)})$. As we will see, quasi-FPT algorithms turn out to be useful for the design of exact algorithms (where the value of k is arbitrary). Combining our algorithm for $k > \log n$ with the algorithm of Dreyfus and Wagner for $k \leq \log n$, one obtains the first polynomial-space FPT algorithm for the problem, of running time $O^*(2^{O(k \log k)})$.[2]

• We obtain a quasi-FPT algorithm running in $O^*(4^k n^{O(\log^2 k)})$ time and polynomial space. This means an improvement on known polynomial-space results for roughly $\omega(\log n) = k < n/3$, which covers many real-world instances. The improved algorithm is based on a novel, simple lemma which shows that, by removing a logarithmic-size (in k) subset of nodes S from any Steiner tree, one can partition the terminal set in two *perfectly balanced* forests, containing at most $k/2$ terminals each. The algorithm however becomes slightly more complicated. In particular, defining two proper Steiner tree subproblems is less obvious in this case for $|S| > 1$.

Combining our algorithm (for small k) with trivial enumeration (for large k), we improve the polynomial-space time complexity of the weighted Steiner tree problem from $O^*(1.62^n)$ to $O^*(1.59^n)$.

• We present a $O^*(1.55^n)$-time polynomial space algorithm for the cardinality Steiner tree problem. Our algorithm combines the quasi-FPT algorithm above (for small k) with a improved branching strategy (for large k). The refined branching exploits the fact that, for k large enough, there must be clusters of terminals "close" to each other. This property can be

[2] Throughout this paper, $\log x$ stands for $\log_2 x$.

used to guide the branching process. In particular, one can branch so that "large" connected components of terminals can be *contracted* afterwards. From a technical point of view, we use a simple charging mechanism to show that, for large k, the graph must contain one of a small list of local configurations of terminals and non-terminals. On such configurations we are able to branch better than trivially.

A standard analysis of our algorithm does not provide any improved time bound. For this reason we use the Measure & Conquer analytical technique described in [21] (see also [17, 18]), which is based on the quasiconvex analysis of multivariate recurrences by Eppstein [13]. The basic idea is designing a convenient (non-trivial) measure of the size of the problem. This measure is used to bound in a tighter way the progress made by the considered recursive algorithm at each branching step. The running time obtained with respect to the refined measure is eventually turned into the equivalent running time in terms of some standard measure (typically the number of nodes or edges for graph problems). Measure & Conquer has been successfully applied to the design of exact algorithms for coloring [12], independent set [18], dominating set [17, 22, 27, 28], cubic-TSP [14], feedback vertex set [16], and maximum leaf spanning tree [20], among others. As it will be clearer from the analysis, a convenient measure in our case is a linear combination of the number n of nodes and number $n - k$ of non-terminals in the graph. (In particular, non-terminals are counted more than once).

• Using a similar algorithm and analysis, we are also able to solve the cardinality Steiner tree problem in $O^*(1.36^n)$ time and exponential space, hence improving on the previous best $O^*(1.42^n)$ running time.

• After the publication of the conference version of this paper, Nederlof [36] presented a $O^*(2^k)$-time polynomial-space algorithm for the cardinality Steiner tree problem. Combining his algorithm with our techniques, one obtains an algorithm running in $O^*(1.36^n)$-time and polynomial space.

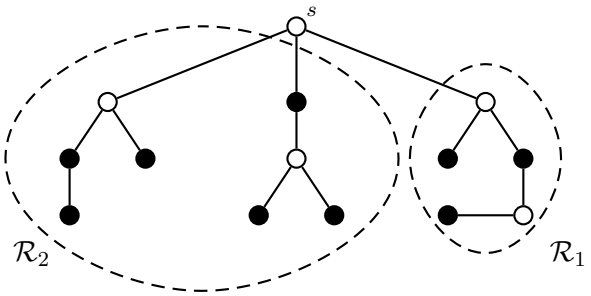
Organization. The rest of this paper is organized as follows. In Section 2 we introduce notation and preliminary notions, including the Steiner separators that we will use in next sections. In Section 3 we present our quasi-FPT algorithms. The improved algorithm for the cardinality Steiner tree problem is described and analyzed in Section 4. The corresponding exponential-space variant is given in Section 5. Conclusions and open problems are discussed in Section 6.

2 Preliminaries

Given a graph $G = (V, E)$, we sometimes use $V(G)$ and $E(G)$ to denote the set of nodes and edges of G, respectively. The minimum weight of a Steiner tree of G on terminals T is denoted by $st_G(T)$. In the cardinality version of the problem, $st_G(T)$ is simply the number of edges in a minimum-size Steiner tree. When the graph G is clear from the context, we will simply write $st(T)$.

One of the crucial operations in our algorithms is contraction of adjacent nodes. By *contracting* a pair of adjacent nodes v and u, we mean (i) removing v and u from the graph, (ii) adding a new node w (*contracted node*), and (iii) adding one edge between w and the nodes $N(u) \cup N(v) \setminus \{u, v\}$, i.e. the neighbors of u and v distinct from those two nodes. Observe that, if a node z is adjacent to both u and v, then edges zu and zv are replaced by

Figure 1 Tight example for Lemma 2 (black nodes are terminals): a separator $S = \{s\}$, and the corresponding forests $\mathcal{R}_1$ and $\mathcal{R}_2$ with $|T_1| = k/3$ and $|T_2| = 2k/3$ terminals, respectively.



a unique edge zw. In the weighted case, we let the weight of zw be the minimum weight of zu and zv.

In the cardinality Steiner tree problem, adjacent terminals can be contracted, hence reducing the size of the problem without branching.

Lemma 1 (Contraction Lemma) *Let (G, T) be an instance of the cardinality Steiner tree problem, and t' and t'' be two adjacent terminals. Let moreover G' be the graph resulting from the contraction of t' and t'' into t. Then*

$$st_G(T) = 1 + st_{G'}(T \setminus \{t', t''\} \cup \{t\}).$$

Proof. Consider an optimum Steiner tree ST. If the edge $t't''$ belongs to ST, the claim is trivially true. Otherwise, adding $t't''$ to ST and removing any other edge of ST in the resulting cycle gives an alternative optimal Steiner tree ST' containing $t't''$. Tree ST' falls in the first case. □

2.1 Steiner Separators

A set of nodes S is called an *α-separator* of a graph G, $0 < \alpha \leq 1$, if the vertex set $V(G) \setminus S$ can be partitioned into sets V_L and V_R of size at most $\alpha\, n$ each, such that no vertex of V_L is adjacent to any vertex of V_R. We next define a similar notion, which turns out to be useful for Steiner trees. Given a Steiner tree ST on terminals T, an α-Steiner separator S of ST is a subset of nodes which partitions ST in two forests $\mathcal{R}_1$ and $\mathcal{R}_2$, each one containing at most αk terminals.

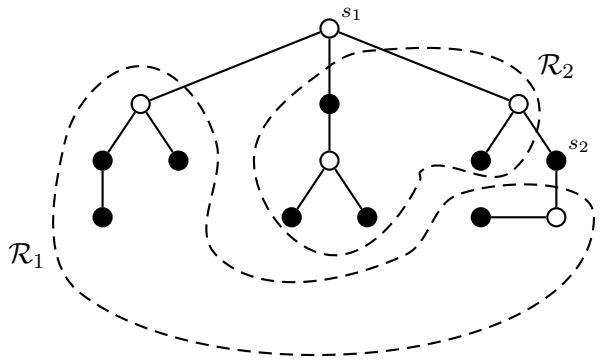
The following result, whose proof is given for the sake of completeness, is implied by [7].

Lemma 2 [7] *Every Steiner tree ST on terminal set T, $|T| = k \geq 3$, has a 2/3-Steiner separator $S = \{s\}$ of size one.*

Proof. We describe a procedure to find one such node s. Take any internal node $v \in ST$ and consider the subtrees $ST_1(v), ST_2(v), \ldots, ST_{p(v)}(v)$ obtained by removing v. Name the subtrees such that $ST_1(v)$ contains the largest number of terminals. With a slight notational abuse, we identify each tree/forest with the set of its nodes. If $|ST_1(v) \cap T| > 2\,k/3$, replace v with the root of $ST_1(v)$, and iterate the process. Otherwise, set $s = v$.

Note that, as far as the condition $|ST_1(v) \cap T| > 2\,k/3$ is satisfied, the union of all the subtrees $ST_i(v)$, $i \geq 2$, contains less than $k/3$ terminals. Moreover, in each iteration the

Figure 2 A perfectly-balanced separator $S = \{s_1, s_2\}$, of size $2 \leq \log 9 + 1$, and the corresponding forests $\mathcal{R}_1$ and $\mathcal{R}_2$, each one containing $4 \leq 9/2$ terminals.



number of terminals in $ST_1(v)$ cannot increase, while the number of its nodes decreases by at least one. It follows that the procedure halts.

It remains to show that the final node s has the desired property. Note that, for each i, $|ST_i(s) \cap T| \leq 2k/3$ by the halting condition. If $ST_1(s)$ contains at least $k/3$ terminals, we are done: $\mathcal{R}_1 = \{ST_1(s)\}$ and $\mathcal{R}_2 = \{ST_2(s), ST_3(s), \ldots, ST_{p(s)}\}$. Otherwise the definition is satisfied by the forests $\mathcal{R}_1 = \{ST_1(s), \ldots, ST_j(s)\}$ and $\mathcal{R}_2 = \{ST_{j+1}(s), \ldots, ST_{p(s)}(s)\}$, where j is the smallest index such that $|\mathcal{R}_1 \cap T| \geq k/3$ (which implies $|\mathcal{R}_2 \cap T| \leq 2k/3$). In fact, since by assumption all the $ST_i(s)$'s contain less than $k/3$ terminals, it must be $|\mathcal{R}_1 \cap T| < 2\,k/3$ by the minimality of j. □

An example of (*roughly-balanced*) 2/3-Steiner separator is given in Figure 1. This example is tight. In particular, in general it is not possible to find a (*perfectly-balanced*) 1/2-Steiner separator of size 1. We next show that a 1/2-Steiner separator of size at most $\log k + 1$ always exists.

We first need the following lemma, whose proof is analogous to the proof of Lemma 2.

Lemma 3 ([7]) *Given a Steiner tree ST on terminal set T, $|T| = k \geq 3$, there is a node s whose removal partitions ST into a forest where each tree contains at most $k/2$ terminals.*

Lemma 4 *Given a Steiner tree ST on terminal set T, $|T| = k \geq 3$, there is a 1/2-Steiner separator S of size at most $\log k + 1$.*

Proof. We construct $V_1 := V(\mathcal{R}_1)$, $V_2 := V(\mathcal{R}_2)$ and S iteratively, starting from empty sets, as follows. Let $G = ST$. By Lemma 3 there is a node s such that, for any connected component $G[C]$ of $G \setminus \{s\}$, $|C \cap T| \leq k/2$. We add s to S and for each component $G[C]$ of $G \setminus \{s\}$, add C to V_1 or V_2 if this does not violate $|V_1 \cap T| \leq k/2$ or $|V_2 \cap T| \leq k/2$, respectively.

Let us show that at most one component $G[C]$ is left outside $V_1 \cup V_2$. Suppose by contradiction that there are at least 2 such components, say $G[C_1]$ and $G[C_2]$. Observe that V_1, V_2, C_1 and C_2 are pairwise dijoint. W.l.o.g. assume $|C_1 \cap T| \leq |C_2 \cap T|$. Since C_1 was added to no V_1 nor V_2, it must be the case that $|V_1 \cap T| + |C_1 \cap T| > k/2$ and $|V_2 \cap T| + |C_1 \cap T| > k/2$. Consequently,

$$|V_1 \cap T| + |V_2 \cap T| + 2|C_1 \cap T| > k.$$

However, this contradicts the fact that

$$|V_1 \cap T| + |V_2 \cap T| + 2|C_1 \cap T| \leq |V_1 \cap T| + |V_2 \cap T| + |C_1 \cap T| + |C_2 \cap T| \leq k.$$

Now we iteratively reapply the construction above up to $\log k$ times, each time considering as graph G the component $G[C]$ left from previous step, if any. Eventually we add C to either V_L or V_R.

Since the number of terminals in $G[C]$ halves at each step, at the end of the process C contains no terminal. Hence the number of terminals in V_1 and V_2 is at most $k/2$. The final size of S is at most $\log k + 1$ by construction. □

An example of perfectly-balanced separator is given in Figure 2. We observe that the Steiner separator of Lemma 4 can be computed in polynomial time, given the Steiner tree. However, we will never exploit this feature, since we will consider the optimum Steiner tree which is unknown. For this reason, we will simply guess the separator by trying all the possible $\sum_{i=1}^{\log k+1} \binom{n}{i} \leq 2n^{\log k+1}$ combinations of at most $\log k + 1$ nodes.

3 Steiner Tree via Steiner Separators

In Section 3.1 we describe a simple polynomial-space algorithm for the (weighted) Steiner tree problem of running time $O^*((27/4)^k n^{O(\log k)})$, based on the roughly-balanced separators given by Lemma 2. We later show in Section 3.2 how to reduce the time complexity to $O^*(4^k n^{O(\log^2 k)})$, exploiting the perfectly-balanced separators implied by Lemma 4. Combining this second algorithm with trivial enumeration one obtains a $O^*(1.59^n)$-time polynomial-space algorithm for the weighted Steiner tree problem.

3.1 Roughly-Balanced Separators

Our algorithm is inspired by the classical dynamic programming algorithm `D&W` by Dreyfus and Wagner [11], which takes $O^*(3^k)$ time and exponential space. Algorithm `D&W` is based on the following observation. Consider any Steiner tree ST on the set of terminals T, $k := |T| \geq 3$. There must be an internal node $s \in ST$, not necessarily a terminal, such that the subtrees of ST rooted at s can be partitioned in two forests $\mathcal{R}_1$ and $\mathcal{R}_2$, each one containing at least one terminal. Let T_i be the terminals in $\mathcal{R}_i$, $i \in \{1, 2\}$. If we compute optimal Steiner trees ST_1 and ST_2 on terminals $T_1 \cup \{s\}$ and $T_2 \cup \{s\}$, respectively, and we merge them together, we obtain an optimal Steiner tree for the original problem. Of course we do not know s nor (T_1, T_2) a priori, but we can guess them by enumerating all the possible cases. Recall that $st_G(T) = st(T)$ is the minimum cost of a Steiner tree of G on terminals T. The following equation holds:

$$st(T) = \min_{s \in V} \min_{(T_1,T_2) \in \mathcal{P}(s,T)} \{st(T_1 \cup \{s\}) + st(T_2 \cup \{s\})\}, \tag{1}$$

where $\mathcal{P}(s, T)$ is the set of possible partitions (T_1, T_2) of $T \setminus \{s\}$ in two non-empty subsets. Algorithm `D&W` essentially applies Equation (1) to any subset of T, in a bottom-up fashion, storing each partial solution computed for later computations. Storing the partial solutions takes $\Omega(2^k)$ space.

A simple-minded approach to obtain a polynomial-space variant of `D&W` is to apply Equation (1) recursively, in a top-down fashion, without storing any partial solution. When $|T| \leq 2$, the problem is solved trivially in polynomial time and space (*base case*). Unfortunately, this approach leads to a very high running time. The main reason is that, by applying Equation (1) as it is, one generates some subproblems with almost the same number of terminals as in the original problem.

This problem can be circumvented by exploiting Lemma 2. In particular, when applying Equation (1), we do not really need to consider all the partitions in $\mathcal{P}(s,T)$, but it is sufficient to consider only the subset $\mathcal{B}(s,T) \subseteq \mathcal{P}(s,T)$ of (roughly balanced) partitions (T_1,T_2) where $|T_1| \leq |T_2| \leq 2k/3$:

$$st(T) = \min_{s \in V} \min_{(T_1,T_2) \in \mathcal{B}(s,T)} \{st(T_1 \cup \{s\}) + st(T_2 \cup \{s\})\}. \quad (2)$$

Using Equation (2) instead of (1) makes no substantial difference with the dynamic programming approach by Dreyfus and Wagner: in fact, in their case the most frequent partitions (which determine the running time) contain a balanced number of terminals, and such partitions are contained both in $\mathcal{B}(s,T)$ and in $\mathcal{P}(s,T)$. The situation changes drastically in the top-down recursive implementation of the algorithm: here the running time is essentially determined by the most *unbalanced* partitions. Hence, replacing $\mathcal{P}(s,T)$ with $\mathcal{B}(s,T)$ has a tremendous impact on the performance of the algorithm.

The following Steiner tree algorithm `paramST` summarizes the discussion above:

- **(base case)** If $T = \{v\}$, return 0. If $T = \{v,w\}$, return the shortest path distance between v and w.
- **(recursive case)** For every $s \in V$ and for every partition (T_1,T_2) of $T \setminus \{s\}$, $|T_1| \leq |T_2| \leq 2k/3$, compute recursively minimum-weight Steiner trees ST_1 and ST_2 over $T_1 \cup \{s\}$ and $T_2 \cup \{s\}$, respectively. Return the cheapest Steiner tree $ST_1 \cup ST_2$ obtained.

Theorem 1 *Algorithm* `paramST` *solves the weighted Steiner tree problem in* $O((27/4)^k n^{O(\log k)})$ *time and polynomial space.*

Proof. The correctness of the algorithm follows from the discussion above, and its space complexity is trivially polynomial. Let $P(k)$ be the number of base instances generated by the algorithm to solve the problem. The time complexity of the algorithm is $O(P(k)n^{O(1)} \log k) = O^*(P(k))$, where we use the fact that each branching step takes polynomial time and the depth of the recursion is $O(\log k)$.

It remains to bound $P(k)$. We will show by induction on $k \geq 2$ that $P(k) \leq Cn^{c \ln k}\alpha^k$, for proper constants $C > 0$, $c > 0$, and $\alpha \geq 4$. Clearly the condition is true for $k < k'$, for an arbitrarily large constant k': it is sufficient to choose large enough C and c. Now assume the condition is satisfied for every $2 \leq h \leq k-1$, and consider an instance with $k \geq k'$ terminals. For a given partition (T_1,T_2), the number of base instances generated is $P(|T_1|+1) + P(|T_2|+1)$. For the sake of simplicity, assume that $k/2$ and $2k/3$ are integers, and $|T_1|+|T_2| = k$: other cases can be handled similarly. By construction, $k/2 \leq |T_2| \leq 2k/3$. Hence, for sufficiently large constants C and c and for $\alpha = 8$, the following inequalities hold:

$$\begin{aligned} P(k) &\leq n \sum_{i=k/2}^{2k/3} \binom{k}{i} (P(i+1) + P(k-i+1)) \leq 2n \sum_{i=k/2}^{2k/3} \binom{k}{i} P(i+1) \\ &\leq 2n\, P(2k/3+1) \sum_{i=k/2}^{2k/3} \binom{k}{i} \leq 2n\, Cn^{c \ln(2k/3+1)} \alpha^{2k/3+1}\, 2^k \\ &\leq C\, n^{c \ln k} (2\alpha^{2/3})^k \leq C\, n^{c \ln k} \alpha^k. \end{aligned}$$

Above we used the fact that $2\alpha^{2/3} = \alpha$ for $\alpha = 8$. In order to obtain a better value of α, we use the following observation.

Fact 1 *For every fixed $x \geq 4$, function $f(y) = \frac{x^y}{y^y(1-y)^{1-y}}$ is increasing on interval $(0, 2/3]$.*

Recall that $i \in [k/3, 2k/3]$. From Stirling's formula, for any $\epsilon > 0$ and k large enough,

$$\begin{aligned}\binom{k}{i} = \frac{k!}{i! \cdot (k-i)!} &\leq (1+\epsilon)\frac{(k/e)^k\sqrt{2\pi\, k}}{(i/e)^i\sqrt{2\pi\, i}\cdot((k-i)/e)^{k-i}\sqrt{2\pi\,(k-i)}} \\ &= (1+\epsilon)\sqrt{\frac{k}{2\pi\, i(k-i)}}\cdot\frac{k^k}{i^i(k-i)^{k-i}} \\ &\leq (1+\epsilon)\sqrt{\frac{9}{4\pi\, k}}\cdot\frac{1}{((i/k)^{i/k}(1-i/k)^{1-i/k})^k} < \frac{1}{((i/k)^{i/k}(1-i/k)^{1-i/k})^k}. \qquad (3)\end{aligned}$$

Combining (3) with Fact 1, one obtains

$$\begin{aligned}\binom{k}{i}P(i+1) &\leq \frac{C n^{c\ln(i+1)}\alpha^{i+1}}{((i/k)^{i/k}(1-i/k)^{1-i/k})^k} \leq \frac{(\alpha^{i/k})^k\alpha\, C\, n^{c\ln(2k/3+1)}}{((i/k)^{i/k}(1-i/k)^{1-i/k})^k} \\ &\leq \alpha\, C\, n^{c\ln(2k/3+1)}\left(\frac{\alpha^{2/3}}{(2/3)^{2/3}(1/3)^{1/3}}\right)^k = \alpha\, C\, n^{c\ln(2k/3+1)}\left(\frac{3\alpha^{2/3}}{2^{2/3}}\right)^k.\end{aligned}$$

It follows that

$$P(k) \leq 2n\sum_{i=k/2}^{2k/3}\binom{k}{i}P(i+1) \leq 2n\, k\,\alpha\, C\, n^{c\ln(2k/3+1)}\left(\frac{3\alpha^{2/3}}{2^{2/3}}\right)^k \leq C\, n^{c\ln k}\alpha^k,$$

for sufficiently large constants C and c and for $\alpha = 27/4$. The claim follows. □

Algorithm `paramST` is not FPT, due to the $n^{O(\log k)}$ factor. However, it can be used to get a factorial-time polynomial-space FPT algorithm.

Corollary 1 *There is a $O^*(2^{O(k\log k)})$-time polynomial-space algorithm for the Steiner tree problem.*

Proof. It is sufficient to run the algorithm of Dreyfus and Wagner for $k \leq \log n$, and algorithm `paramST` otherwise. In both cases the space complexity is polynomial. In the first case the running time is polynomial. In the second case the running time is $O^*((27/4)^k n^{O(\log k)}) = O^*(2^{O(k\log k)})$, where we exploit the fact that $n \leq 2^k$. □

3.2 Perfectly-Balanced Separators

In this section we describe a variant of `paramST`, running in time $O^*(4^k n^{O(\log^2 k)})$ and polynomial space. The basic idea is combining the approach of Section 3.1 with the perfectly-balanced separators S guaranteed by Lemma 4.

The main difficulty here is that the optimal solution to each subproblem needs not to be a tree. More precisely, let ST be the optimal Steiner tree, $S \subseteq V(ST)$ be the separator, and (V_1, V_2) be the partition of $V(ST) \setminus S$ induced by S. We use $T_i = V_i \cap T$ to denote the terminals in V_i. The subgraphs $ST_1 := ST[V_1 \cup S]$ and $ST_2 := ST[V_2 \cup S]$ of ST are in general forests. They are guaranteed to be trees only for $|S| = 1$, which is the case in Section 3.1. See the left part of Figure 3 for an example.

Figure 3 On the left, a Steiner tree ST of a graph G. (The other edges of G are not depicted for visualization reasons). Black nodes denote terminals. Squared nodes define a perfectly-balanced separator $S = \{a, b, c, d, e, f\}$. The nodes above and below S define the node sets V_1 and V_2, respectively. In particular, T_1 and T_2 are given by the top and bottom black nodes, respectively. Observe that $ST_1 = ST[V_1 \cup S]$ is a forest. On the right, the corresponding tree ST'. In particular, $F = \{bc, de\}$, $E_1 = \{ab, ef\}$, and $E_2 = \{cd\}$.

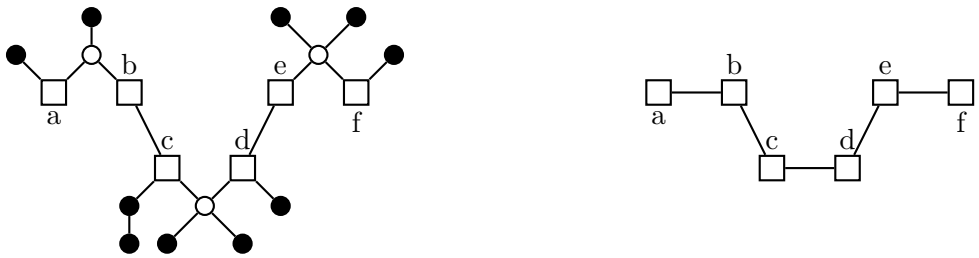
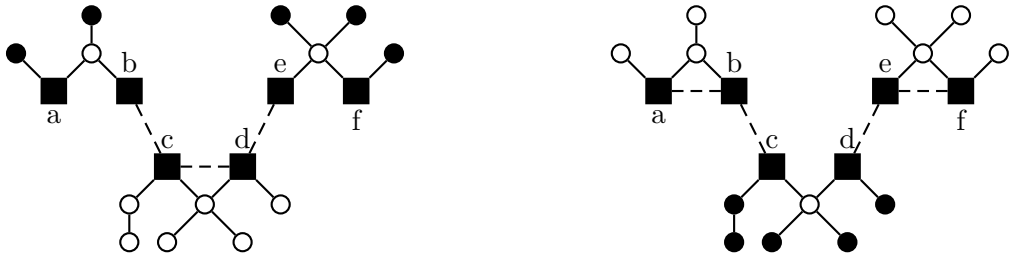


Figure 4 The two subproblems corresponding to the instance of Figure 3. For visualization reasons, we drew only the edges of ST, plus the edges in $F \cup E_2$ and $F \cup E_1$ (dashed edges on the left and right, respectively).



In order to circumvent this problem, we use the following observation. Let us iteratively contract the edges of ST with at least one endpoint not in S. More precisely, the resulting contracted node is the node in S, if any, and otherwise one of the two endpoints chosen arbitrarily. Note that the resulting graph ST' is indeed a tree, and the node-set of ST' is S.

The tree ST' will (possibly) contain some edges F between nodes in S. These edges belong to ST (and hence to G) as well, since they are not contracted. The remaining edges $\{s', s''\}$ correspond to trees of either ST_1 or ST_2, which contain both s' and s''. In other terms, the internal nodes of the path between s' and s'' in ST are contained either in V_1 or in V_2. Let us call these edges E_1 and E_2, respectively. We remark that (F, E_1, E_2) is a 3-partition of $E(ST')$. See the right part of Figure 3 for an example.

The idea is then as follows. In order to compute ST_1, it is sufficient to compute the cheapest forest which spans $T_1 \cup V(E_1)$, and such that each pair $\{s', s''\} \in E_1$ belongs to the same tree. The latter problem can be reduced to a standard Steiner tree problem: it is sufficient to set to zero the cost of edges in $E_2 \cup F$, and compute the cheapest Steiner tree on terminals $T_1 \cup S$. Then one removes the edges $E_2 \cup F$. Symmetrically, one can construct ST_2. Hence, given the tuple $(S, T_1, T_2, F, E_1, E_2)$, one can split the original problem in two subproblems, which can be solved independently. See Figure 4 for an example.

This intuition is formalized in the next lemma. For notational convenience, let us replace the input graph with its metric closure[3]. This does not change the value of the optimal solution and any solution to the new instance can be turned into a solution of the original instance of no-larger cost (by replacing edges with the corresponding shortest paths). For a given $E' \subseteq E$, let $G_{E'}$ be the weighted graph obtained from G by setting to zero the cost of edges E'.

Lemma 5 *Given a graph $G = (V, E)$ and a set of terminals T, $|T| \geq 3$,*

$$st_G(T) = \min\{w(F) + st_{G_{E_2 \cup F}}(T_1 \cup S) + st_{G_{E_1 \cup F}}(T_2 \cup S)\}, \tag{4}$$

where the minimum is taken over

- *All the subsets $S \subseteq V$ of at most $\log k + 1$ nodes;*
- *All the partitions (T_1, T_2) of $T \setminus S$ with $|T_1|, |T_2| \leq |T|/2$;*
- *All the disjoint subsets $F, E_1, E_2 \in \binom{S}{2}$, such that $ST' := (S, F \cup E_1 \cup E_2)$ is a tree.*

Proof. We first observe that, for a given tuple $(S, T_1, T_2, F, E_1, E_2)$, one can construct a subgraph spanning the terminals T of cost exactly $w(F) + st_{G_{E_2 \cup F}}(T_1 \cup S) + st_{G_{E_1 \cup F}}(T_2 \cup S)$. In fact, consider the tree ST_1 of cost $st_{G_{E_2 \cup F}}(T_1 \cup S)$, and remove from ST_1 the edges $E_2 \cup F$. Construct ST_2 symmetrically. Then the graph $ST_1 \cup ST_2 \cup F$ is a connected subgraph of G which spans T, and its cost is as claimed.

It remains to show that there is a choice of the tuple $(S, T_1, T_2, F, E_1, E_2)$ leading to a solution of cost at most $st_G(T)$. Let ST be the optimal Steiner tree. We let S be the perfectly-balanced Steiner separator of ST which is guaranteed by Lemma 4, and (T_1, T_2) and (V_1, V_2) be the corresponding partitions of $T \setminus S$ and $V \setminus S$. Observe that S, T_1 and T_2 satisfy the conditions of the claim.

Let us construct the auxiliary tree ST' and the 3-partition (F, E_1, E_2) of its edge set as described before, with respect to ST and S. Also the triple (F, E_1, E_2) satisfies the claim. Next consider the forests $ST_1 := ST[V_1 \cup S]$ and $ST_2 := ST[V_2 \cup S]$. Note that $w(ST) = w(F) + w(ST_1) + w(ST_2)$ by construction. Furthermore, for $\{i, j\} = \{1, 2\}$, $ST_i \cup E_j \cup F$ is a feasible solution to the Steiner tree problem on graph $G_{E_j \cup F}$ and terminals $T_i \cup S$, of cost $w(ST_i)$. We can conclude that

$$st_G(T) = w(ST) = w(F) + w(ST_1) + w(ST_2) \geq w(F) + st_{G_{E_2 \cup F}}(T_1 \cup S) + st_{G_{E_1 \cup F}}(T_2 \cup S).$$

The claim follows □

We are now ready to describe the improved version of `paramST`. For $|T| \leq k'$, for a proper constant $2 \leq k' = O(1)$, the problem is solved by brute force in polynomial time and space. Otherwise, the algorithm branches according to Equation (4).

Theorem 2 *Algorithm* `paramST`*, modified as above, solves the weighted Steiner tree problem in time $O^*(4^k n^{O(\log^2 k)})$ and polynomial space.*

Proof. The correctness of the algorithm follows from Lemma 5 and its space complexity is trivially polynomial. Let $P(k)$ be the number of base instances generated by the algorithm to

[3]We recall that the *metric closure* of a weighted graph $G = (V, E)$ is a complete graph on node-set V, with weights given by the shortest path distances with respect to original weights.

solve an instance on k terminals. By the same argument as in Theorem 1, the running time of the algorithm is $O^*(P(k))$.

We next show by induction that $P(k) \leq Cn^{c\log^2 k}4^k$, for some constants $C > 0$, $c > 0$. Clearly the condition is true for $k < k' = O(1)$. Now assume it is satisfied for every $2 \leq h \leq k-1$, and consider an instance with $k \geq k'$ terminals. There are at most $\sum_{i=1}^{\log k+1}\binom{n}{i} \leq 2n^{\log k+1}$ possible choices for the separator S. For a fixed choice of S, there are $2^{|T\setminus S|} \leq 2^k$ possible partitions (T_1, T_2) of $T \setminus S$.

By Cayley's formula, the number of spanning trees ST' of S is $|S|^{|S|-2} \leq (\log k+1)^{\log k-1} \leq (\log k)^{2\log k}$. There are at most $3^{|S|-1} \leq 3^{\log k}$ many ways to 3-partition the edges of a given ST'. Consequently, there are at most $(\log k)^{2\log k}3^{\log k} \leq (\log k)^{3\log k}$ possible choices for the triple (F, E_1, E_2).

Altogether the algorithm generates at most $2n^{\log k+1} \cdot 2^k \cdot (\log k)^{3\log k} \leq n^{5\log k}2^k$ pairs of subproblems, on at most $k/2 + \log k + 1$ terminals each. Here we are using the fact that $k \geq k'$ is large enough. By the inductive hypothesis, for proper constants C and c,

$$\begin{aligned} P(k) &\leq n^{5\log k}2^k \cdot 2\,P(k/2+\log k+1) \leq n^{5\log k}2^k \cdot 2\,C\,n^{c\log^2(k/2+\log k+1)}4^{k/2+\log k+1} \\ &\leq C\,n^{3+5\log k+c\log^2(k/2+\log k+1)}4^k \leq Cn^{c\log^2 k}4^k. \end{aligned}$$

The claim follows. □

We can exploit algorithm `paramST` in combination with trivial enumeration to obtain an improved exact algorithm for the weighted Steiner tree problem for arbitrary values of k.

Corollary 2 *The weighted Steiner tree problem can be solved in $O^*(1.5875^n)$ time and polynomial space.*

Proof. Consider the algorithm which runs `paramST` if $k < n/3$, and trivial enumeration otherwise. This algorithm is obviously correct and its space complexity is trivially polynomial. For $k < n/3$, the running time is $O^*(4^{n/3+o(n)}) = O^*(1.5875^n)$. For $k \geq n/3$, the running time is $O^*(\sum_{i=0}^{k}\binom{n-k}{i}) = O^*(\binom{2n/3}{n/3}) = O^*(2^{2n/3}) = O^*(1.5875^n)$. □

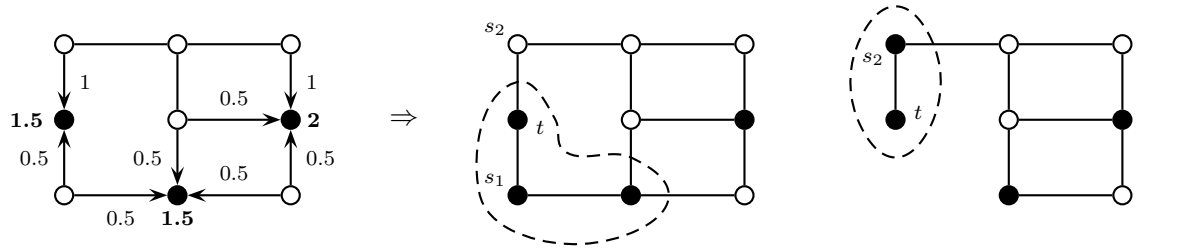
4 Branching on Small-Load Terminals

In Section 4.1 we describe a simple, recursive algorithm `exactST` for the cardinality Steiner tree problem. Our algorithm computes the size $st_G(T)$ of an optimal Steiner tree, but it can be easily modified in order to produce one optimal Steiner tree. In Section 4.2 we show that `exactST` runs in $O^*(1.5468^n)$ time and polynomial space. The running time analysis is based on the Measure & Conquer technique [21]. In particular, we will measure the size of the subproblems in terms of a linear combination of the number n of nodes and number $n-k$ of non-terminals.

4.1 Algorithm

The main idea behind our algorithm is as follows. If $k \leq cn$ for a suitable constant $c < 1$, it is convenient to use the $O^*(4^k n^{O(\log^2 k)})$-time algorithm `paramST` from Section 3.2. Otherwise, there must be a terminal t which is at distance at most one from "many" other terminals. Thus, if by branching we add to T one or more non-terminals adjacent to t, we can contract

Figure 5 Assignment of load to terminals (black nodes) on the left, and multiple branching on the minimum load terminal t: in the first subproblem s_1 is added to the terminals (middle); in the second s_1 is deleted and s_2 is added to the terminals (right). Dashed curves indicate components of terminals which will be contracted in the following step.



a "large" connected component of terminals afterwards (applying the Contraction Lemma 1). This phenomenon is not exploited in trivial enumeration, and it is at the base of our refined branching algorithm.

In order to formalize in a convenient way the mentioned scenario, we introduce the following definition of *load* of a terminal. Let each non-terminal node $s \in N := V \setminus T$ be initially assigned a load one. Node s evenly distributes its load among the terminals adjacent to it (if any). The final load $\ell(t)$ of each terminal t is the sum of the loads received by its non-terminal neighbors. The process is illustrated in Figure 5. As it will be clearer from the analysis, we can branch efficiently on terminals of small load.

We are now ready to describe algorithm `exactST`:

1. (**base**) If $k = |T| \in \{0, 1\}$, $st_G(T) = 0$:

$$\texttt{exactST}(G, T) = 0.$$

2. (**contraction**) If there is a connected component V' of at least 2 terminals, we apply iteratively the Contraction Lemma 1. Let G' be the graph obtained from G by iteratively contracting the nodes of V' until a unique node v' is left. Let moreover $T' = T \setminus V' \cup \{v'\}$. Then

$$\texttt{exactST}(G, T) = |V'| - 1 + \texttt{exactST}(G', T').$$

3. (**reduction**) If there is a terminal t adjacent to a unique (non-terminal) node s, we add s to the terminals since s must belong to any Steiner tree (being $k \geq 2$):

$$\texttt{exactST}(G, T) = \texttt{exactST}(G, T \cup \{s\}).$$

4. (**small k**) If $k \leq 2n/7$, we apply algorithm `paramST` from Section 3.2:

$$\texttt{exactST}(G, T) = \texttt{paramST}(G, T).$$

5. (**single branch**) If there is a non-terminal s adjacent to at least 3 terminals, we simply branch by either removing s from the graph, or by adding it to the terminals:

$$\texttt{exactST}(G, T) = min\{\texttt{exactST}(G \setminus \{s\},\, T),\ \texttt{exactST}(G,\, T \cup \{s\})\}.$$

6. (**multiple branch**) Let t be a terminal of minimum load according to the definition above, and let $s_1, s_2, \ldots, s_p$ be the (non-terminal) neighbors of t, sorted in decreasing number $m_1, m_2, \ldots, m_p$ of adjacent terminals. We branch on the p subproblems obtained by removing $s_1, s_2, \ldots, s_{i-1}$, and adding s_i to the terminals, for $i \in \{1, 2, \ldots, p\}$ (see also Figure 5):

$$\texttt{exactST}(G, T) = \min_{i \in \{1,2,\ldots,p\}} \{\texttt{exactST}(G \setminus \{s_1, s_2, \ldots, s_{i-1}\}, T \cup \{s_i\})\}.$$

Observe that Algorithm `exactST` does not work in the weighted case. This is essentially due to the fact that the Contraction Lemma 1 does not extend to such case. Finding an improved branching strategy (when k is large) for the weighted Steiner tree problem is an interesting open problem.

4.2 Analysis

We next analyze algorithm `exactST` with the Measure & Conquer approach described in [21]. Let $n_N := n - k$ be the number of non-terminals. We will measure the size of the subproblems in terms of $h := n + \alpha\, n_N$, where $\alpha > 0$ is a proper constant that we obtained numerically.

Theorem 3 *Algorithm* `exactST` *solves the cardinality Steiner tree problem in time* $O^*(1.5468^n)$ *and polynomial space.*

Proof. The correctness of the algorithm is discussed above and its space complexity is trivially polynomial. For $k \leq 2n/7$ the running time of the algorithm is $O^*(4^k n^{O(\log^2 k)}) = O^*(4^{2n/7+o(n)}) = O^*(1.4860^n)$. So assume that initially $k > 2n/7$. We let $h := n + \alpha\, n_N$ be the *size* of the problem, where $\alpha = 0.4875$. Denote by $T(h)$ the time required to solve a problem of size h. We will show by induction that $T(h) = O^*(1.38197^h)$. The claim follows since, being initially $n_N \leq 5n/7$ by assumption, $O^*(1.38197^h) = O^*(1.38197^{1.34822 \cdot n}) = O^*(1.5468^n)$.

Let $poly(h) = O(n^{O(1)})$ be the maximum (polynomial) time spent at each step of the algorithm (excluding the recursive calls). For $h = 0$, we have $k = 0$ and hence $T(h) \leq poly(h) = O^*(1)$. Assume now that $T(h') = O^*(1.38197^{h'})$ for any $h' < h$, and consider the different steps of the algorithm.

Case 1 (base). The problem is solved directly:

$$T(h) \leq poly(h).$$

Case 2 (contraction). The algorithm generates a unique subproblem containing at most $n - 1$ nodes and n_N non-terminals:

$$T(h) \leq poly(h) + T(h-1) = poly(h) + O^*(1.38197^{h-1}) = O^*(1.38197^h).$$

Case 3 (reduction). The algorithm adds s to the set of terminals (and hence removes one node from the non-terminals), and then removes at least one node by Case 2:

$$T(h) \leq 2\, poly(h) + T(h - \alpha - 1) = 2\, poly(h) + O^*(1.38197^{h-\alpha-1}) = O^*(1.38197^h).$$

Observe that, from a technical point of view, we are considering two distinct recursive calls: in the first call we add s to the set of terminals, and in the second we perform a contraction. This explains the factor 2 in front of $poly(h)$. The coefficients in front of $poly(h)$ in the following recurrences have a similar motivation.

$(m_1, m_2, \ldots, m_p)$	$\ell(t)$	size decrease
$(1,1)$	$4/2$	$1+\alpha, 2+2\,\alpha$
$(2,1)$	$3/2$	$2+\alpha, 2+2\,\alpha$
$(2,2)$	$2/2$	$2+\alpha, 3+2\,\alpha$
$(2,2,1)$	$4/2$	$2+\alpha, 3+2\,\alpha, 3+3\,\alpha$
$(2,2,2)$	$3/2$	$2+\alpha, 3+2\,\alpha, 4+3\,\alpha$
$(2,2,2,2)$	$4/2$	$2+\alpha, 3+2\,\alpha, 4+3\,\alpha, 5+4\,\alpha$

Table 1: Feasible values of $(m_1, m_2, \ldots, m_p)$ for multiple branch on terminal t, with the corresponding load (strictly smaller than $5/2$), and decrease of the problem size. In particular, a decrease of $x + \alpha\, y$ reflects the removal of x nodes and y non-terminals.

Case 4 (small k). The problem is solved by applying algorithm `paramST`, in time $O^*(4^k n^{O(\log^2 k)})$. Recall that $k \leq 2n/7$ in this case. Then

$$k = (n + \alpha\, n_N)\frac{k}{n + \alpha\, n_N} = h\,\frac{k}{(1+\alpha)n - \alpha\, k} \leq h\frac{2n/7}{(1+5\alpha/7)n} = h\,\frac{2}{7+5\,\alpha}.$$

Moreover, the value of $|n-k|$ changes by a constant amount in each step of the algorithm. (In particular, when several nodes are removed, most of them are terminals). This implies $k = \Theta(n) = \Theta(h)$, since this condition holds initially by assumption. Hence the running time is

$$T(h) = O^*(4^k n^{O(\log^2 k)}) = O^*(4^{k+o(k)}) = O^*(4^{2h/(7+5\,\alpha)+o(h)}) = O^*(1.3415^h).$$

Case 5 (single branch). Let $p \geq 3$ be the number of terminals adjacent to the selected non-terminal s. The algorithm generates two subproblems. In the first subproblem it removes s from the graph. In the second subproblem it adds s to the set of terminals, and then it removes p nodes by Case 2. Hence

$$\begin{aligned} T(h) &\leq 2\,poly(h) + T(h-\alpha-1) + T(n-\alpha-3) \\ &= 2\,poly(h) + O^*(1.38197^{\,h-\alpha-1}) + O^*(1.38197^{\,h-\alpha-3}) = O^*(1.38197^{\,h}). \end{aligned}$$

Case 6 (multiple branch). Observe that, being $k > 2n/7$ by Case 4, the minimum load of a node is at most $\frac{n-k}{k} < \frac{5n/7}{2n/7} = 5/2$. In particular, for the selected terminal t, $\ell(t) < 5/2$. Recall that $s_1, s_2, \ldots, s_p$ are the (non-terminal) neighbors of t, in decreasing order $m_1, m_2, \ldots, m_p$ of the number of adjacent terminals. By Case 3, $p \geq 2$ (each terminal is adjacent to at least 2 non-terminals). Note that the load assigned by s_i to t is exactly $1/m_i$. By Case 5 it must be $m_i \in \{1,2\}$ for each i (each non-terminal has between 0 and 2 terminal neighbors). Hence the contribution of each s_i to the load $\ell(t)$ of t is either 1 or $1/2$. It follows by a simple case enumeration that the sequence $(m_1, m_2, \ldots, m_p)$ must be one of the sequences in Table 1.

In the ith subproblem, $i \in \{1, 2, \ldots, p\}$, the algorithm removes nodes $s_1, s_2, \ldots, s_{i-1}$ from the graph, and adds node s_i to the terminals, which later determines the removal of m_i nodes by Case 2. Note that in the ith step i non-terminals are removed. Hence, by an easy

case-by-case check,

$$T(h) \leq (1+p)\,poly(h) + \sum_{i=1}^{p} T(h-(i-1)-m_i-\alpha\cdot i)$$
$$= O^*(\sum_{i=1}^{p} 1.38197^{\,h-(1+\alpha)\,i-m_i+1}) = O^*(1.38197^{\,h}).$$

This concludes the proof. □

5 An Exponential-Space Algorithm

As a by-product of our approach, we are able to improve on the current best $O^*(1.4143^n)$-time exponential-space algorithm for the cardinality Steiner tree problem. This is achieved by modifying algorithm `exactST` in the following way.

- In Step 4 replace `paramST` with the $O^*(2^k)$-time exponential-space algorithm of [5], and increase the corresponding threshold from $k \leq 2n/7$ to $k \leq 3n/7$.
- In Step 5 increase the threshold number of adjacent terminals from 3 to 5.

In its analysis, the best choice for α turns out to be zero (i.e., the subproblems size h is simply measured in terms of the number n of nodes).

Theorem 4 *Algorithm* `exactST`*, modified as above, solves the cardinality Steiner tree problem in time* $O(1.3533^n)$ *and exponential space.*

Proof. We can use the same type of analysis and notation as in Theorem 3, but using the standard measure $h = n$. We show by induction that the running time is $T(n) = O^*(1.3533^n)$.

The base of the induction and the inductive hypothesis for Cases 1, 2, and 3 are trivially satisfied. In Case 4 (in its modified version), the problem is solved in time $O^*(2^k) = O^*(2^{3n/7}) = O^*(1.3460^n)$. In Case 5 (in its modified version), the node considered is adjacent to $p \geq 5$ terminals. Hence

$$T(n) \leq 2\,poly(n) + T(n-2) + T(n-1-p) \leq 2\,poly(n) + T(n-2) + T(n-6)$$
$$= 2\,poly(n) + O^*(1.3533^{n-2}) + O^*(1.3533^{n-6}) = O^*(1.3533^n).$$

In Case 6, being $k > 3n/7$, the minimum load of a node is at most $\frac{n-k}{k} < \frac{4n/7}{3n/7} = 4/3$. Recall that $s_1, s_2, \ldots, s_p$, $p \geq 2$, are the (non-terminal) neighbors of t, in decreasing order $m_1, m_2, \ldots, m_p$ of the number of adjacent terminals. By Case 5 (in its modified form) it must be $m_i \in \{1,2,3,4\}$. The corresponding set of feasible local configurations is described in Table 2.

In the ith subproblem, $i \in \{1,2,\ldots,p\}$, the algorithm removes nodes $s_1, s_2, \ldots, s_{i-1}$ from the graph, and adds node s_i to the terminals, which later determines the removal of m_i nodes by Case 2. Altogether, $i-1+m_i$ nodes are removed. A case-by-case check shows that the following set of inequalities is satisfied for any feasible choice of the vector $(m_1, m_2, \ldots, m_p)$:

$$T(n) \leq (1+p)\,poly(n) + \sum_{i=1}^{p} T(n-(i-1)-m_i) = O^*(\sum_{i=1}^{p} 1.3533^{n-i-m_i+1}) = O^*(1.3533^n).$$

This concludes the proof. □

$(m_1, m_2, \ldots, m_p)$	$\ell(t)$	size decrease
$(4,4)$	$6/12$	$4,5$
$(4,3)$	$7/12$	$4,4$
$(4,2)$	$9/12$	$4,3$
$(4,1)$	$15/12$	$4,2$
$(3,3)$	$8/12$	$3,4$
$(3,2)$	$10/12$	$3,3$
$(2,2)$	$12/12$	$2,3$
$(4,4,4)$	$9/12$	$4,5,6$
$(4,4,3)$	$10/12$	$4,5,5$
$(4,4,2)$	$12/12$	$4,5,4$
$(4,3,3)$	$11/12$	$4,4,5$

$(m_1, m_2, \ldots, m_p)$	$\ell(t)$	size decrease
$(4,3,2)$	$13/12$	$4,4,4$
$(4,2,2)$	$15/12$	$4,3,4$
$(3,3,3)$	$12/12$	$3,4,5$
$(3,3,2)$	$14/12$	$3,4,4$
$(4,4,4,4)$	$12/12$	$4,5,6,7$
$(4,4,4,3)$	$13/12$	$4,5,6,6$
$(4,4,4,2)$	$15/12$	$4,5,6,5$
$(4,4,3,3)$	$14/12$	$4,5,5,6$
$(4,3,3,3)$	$15/12$	$4,4,5,6$
$(4,4,4,4,4)$	$15/12$	$4,5,6,7,8$

Table 2: Feasible values of $(m_1, m_2, \ldots, m_p)$ for multiple branch on terminal t in the exponential-space algorithm, with the corresponding load (strictly smaller than 4/3) and number of nodes removed.

6 Conclusions and Open Problems

In this paper we investigated the problem of computing optimal Steiner trees in polynomial space. We developed the first non-trivial algorithms for this problem, both for $k \ll n$ and for arbitrary k. Nederlof very recently developed an algorithm for the cardinality Steiner tree problem running in $O^*(2^k)$ time and polynomial space [36]. Combining our approach with his result, one obtains the following improvement of Theorems 3 and 4.

Theorem 5 *There is an algorithm which solves the cardinality Steiner tree problem in time* $O(1.3533^n)$ *and polynomial space.*

Proof. It is sufficient to use the algorithm in [36] in Step 4 of the algorithm of Section 5. The running time analysis is not affected and the space usage becomes polynomial. □

We remark that Nederlof's approach does not generalize to the weighted version of the problem. In that case, our $O^*(4^k n^{O(\log^2 k)})$ algorithm remains the best-known result in polynomial space. There is an interesting parallel between the state of the art for the Steiner tree problem and the traveling salesman problem. For the latter problem the best-known polynomial-space algorithm for the unweighted case (i.e. for the Hamiltonian cycle problem) runs in $O^*(2^n)$ time [2, 33], while the best-known polynomial-space running time for the weighted case is $O^*(4^n n^{O(\log n)})$ [30]. Developing improved algorithms to compute Steiner trees of minimum weight, both for small and for large k, is an interesting open problem.

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *Journal of the ACM*, 45:753–782, 1998.

[2] E. T. Bax. Inclusion and exclusion algorithm for the hamiltonian path problem. *Information Processing Letters*, 47:203–207, 1993.

[3] M. Bern and P. Plassmann. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[4] A. Björklund and T. Husfeldt. Inclusion-exclusion algorithms for counting set partitions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–582, 2006.

[5] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbious: Fast subset convolution. In *ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2007.

[6] G. E. Blelloch, K. Dhamdhere, E. Halperin, R. Ravi, R. Schwartz, and S. Sridhar. Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 667–678, 2006.

[7] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

[8] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In *ACM Symposium on Theory of Computing (STOC)*, 2010. Best Paper Award. To appear.

[9] L. L. Deneen, G. M. Shute, and C. D. Thomborson. A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Structures and Algorithms*, 5(4):535–557, 1994.

[10] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.

[11] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1971/72.

[12] D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 329–337, 2001.

[13] D. Eppstein. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms. *ACM Transactions on Algorithms*, 2(4):492–509, 2006.

[14] D. Eppstein. The Traveling Salesman Problem for Cubic Graphs. *Journal of Graph Algorithms and Applications*, 11(1):61–81, 2007.

[15] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[16] F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon. On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica*, 52(2): 293–307, 2008.

[17] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination - a case study. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 191–203, 2005.

[18] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: a simple $O(2^{0.288\,n})$ independent set algorithm. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 18–25, 2006.

[19] F. V. Fomin, F. Grandoni and D. Kratsch. Faster Steiner Tree Computation in Polynomial-Space. In *European Symposium on Algorithms (ESA)*, pages 430–441, 2008.

[20] F. V. Fomin, F. Grandoni, D. Kratsch, Solving Connected Dominating Set Faster than 2^n. *Algorithmica*, 52: 153–166, 2008.

[21] F. V. Fomin, F. Grandoni, D. Kratsch. A Measure & Conquer Approach for the Analysis of Exact Algorithms. *Journal of the ACM,* 56(5): 2009.

[22] F. V. Fomin, F. Grandoni, A. Pyatkin, and A. Stepanov. Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. *ACM Transactions on Algorithms,* 5(1): 2008.

[23] B. Fuchs, W. Kern, D. Mölle, S. Richter, P. Rossmanith, and X. Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3): 493–500, 2007.

[24] J. L. Ganley. Computing optimal rectilinear Steiner trees: a survey and experimental evaluation. *Discrete Applied Mathematics*, 90(1-3):161–171, 1999.

[25] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, 1979.

[27] F. Grandoni. *Exact Algorithms for Hard Graph Problems*. PhD thesis, Università di Roma Tor Vergata, Roma, Italy, Mar. 2004.

[28] F. Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4(2):209–214, 2006.

[29] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In *International Workshop on Algorithms and Data Structures (WADS)*, pages 36–48. Springer, 2005.

[30] Y. Gurevich and S. Shelah. Expected computation time for Hamiltonian path problem. *SIAM Journal on Computing*, 16(3):486–502, 1987.

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

[32] A. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer, Dordrecht, 1995.

[33] R. M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operation Research Letters*, 1:49–51, 1982.

[34] B. Korte, H. J. Prömel, and A. Steger. Steiner trees in VLSI-layout. In *Paths, Flows, and VLSI-Layout*, pages 185–214, 1990.

[35] D. Mölle, S. Richter, and P. Rossmanith. A faster algorithm for the Steiner tree problem. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 561–570, 2006.

[36] J. Nederlof. Fast polynomial-space algorithms using Mobius inversion: Improving on Steiner Tree and related problems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 713–725, 2009.

[37] R. Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

[38] H. J. Prömel and A. Steger. *The Steiner tree problem*. Advanced Lectures in Mathematics. Friedr. Vieweg & Sohn, Braunschweig, 2002.

[39] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 770–779, 2000.

[40] G. Woeginger. Space and time complexity of exact algorithms: Some open problems. In *International Workshop on Parameterized and Exact Computation (IWPEC)*, pages 281–290, 2004.