Parameterized Approximation Schemes
for Steiner Trees with Small Number of Steiner Vertices[*]

Pavel Dvořák[†][1], Andreas Emil Feldmann[2],
Dušan Knop[2,3], Tomáš Masařík[2], Tomáš Toufar[1], and Pavel Veselý[1]

[1]Computer Science Institute, Charles University, Prague, Czech Republic
{koblich,toufar,vesely}@iuuk.mff.cuni.cz
[2]Department of Applied Mathematics, Charles University, Prague, Czech Republic
feldmann.a.e@gmail.com, {knop,masarik}@kam.mff.cuni.cz
[3]Department of Informatics, University of Bergen, Bergen, Norway

Abstract

We study the STEINER TREE problem, in which a set of *terminal* vertices needs to be connected in the cheapest possible way in an edge-weighted graph. This problem has been extensively studied from the viewpoint of approximation and also parametrization. In particular, on one hand STEINER TREE is known to be APX-hard, and W[2]-hard on the other, if parameterized by the number of non-terminals (*Steiner vertices*) in the optimum solution. In contrast to this we give an *efficient parameterized approximation scheme* (EPAS), which circumvents both hardness results. Moreover, our methods imply the existence of a *polynomial size approximate kernelization scheme* (PSAKS) for the assumed parameter.

We further study the parameterized approximability of other variants of STEINER TREE, such as DIRECTED STEINER TREE and STEINER FOREST. For neither of these an EPAS is likely to exist for the studied parameter: for STEINER FOREST an easy observation shows that the problem is APX-hard, even if the input graph contains no Steiner vertices. For DIRECTED STEINER TREE we prove that computing a constant approximation for this parameter is W[1]-hard. Nevertheless, we show that an EPAS exists for UNWEIGHTED DIRECTED STEINER TREE. Also we prove that there is an EPAS and a PSAKS for STEINER FOREST if in addition to the number of Steiner vertices, the number of connected components of an optimal solution is considered to be a parameter.

1 Introduction

In this paper we study several variants of the STEINER TREE problem. In its most basic form this optimization problem takes an undirected graph $G = (V, E)$ with edge weights $w(e) \in \mathbb{R}_0^+$ for every $e \in E$, and a set $R \subseteq V$ of *terminals* as input. The non-terminals in $V \setminus R$ are called *Steiner vertices*. A *Steiner tree* is a tree in the graph G, which spans all terminals in R and may contain some of the Steiner vertices. The objective is to minimize the total weight $\sum_{e \in E(T)} w(e)$ of the computed Steiner tree $T \subseteq G$. This fundamental optimization problem is one of the 21 original NP-hard problems listed by Karp [25] in his seminal paper from 1972, and has been intensively studied since then. The STEINER TREE problem and its variants have applications in network design, circuit layouts, and phylogenetic tree reconstruction, among others (see survey [23]).

Two popular ways to handle the seeming intractability of NP-hard problems are to design *approximation* [35] and *parameterized* [13] algorithms. For the former, an *α-approximation* is computed in polynomial time for some factor α specific to the algorithm, i.e., the solution is always at most a multiplicative factor of α worse than the optimum of the input instance. The STEINER TREE problem, even in its basic form as defined above, is APX-hard [12], i.e., it is NP-hard to obtain an approximation factor of $\alpha = \frac{96}{95} \approx 1.01$. However a factor of $\alpha = \ln(4) + \varepsilon \approx 1.39$ can be achieved in polynomial time [6], which is the currently best factor known for this runtime.

For parameterized algorithms, an instance is given together with a *parameter p* describing some property of the input. The optimum solution is computed in time $f(p) \cdot n^{O(1)}$, where f is a computable function independent of the input size n. If such an algorithm exists, we call the problem *fixed-parameter tractable* (FPT) for parameter p. A well-studied parameter for the STEINER TREE problem is the number of terminals $|R|$. It is known since the classical result of Dreyfus and Wagner [16] that the STEINER TREE problem is FPT for this parameter, as the problem can be solved in time $3^{|R|} \cdot n^{O(1)}$ if $n = |V|$, which for unweighted graphs can be improved to $2^{|R|} \cdot n^{O(1)}$ using the results of Björklund et al. [2]. A somewhat complementary and less-studied parameter to the number of terminals is the number of Steiner vertices in the optimum solution, i.e., $p = |V(T) \setminus R|$ if T is an optimum Steiner tree. It is known [15] that STEINER TREE is W[2]-hard for parameter p and therefore is unlikely to be FPT, in contrast to the parameter $|R|$. This parameter p has been mainly studied in the context of unweighted graphs before. The problem remains W[2]-hard in this special case and therefore the focus has been on designing parameterized algorithms for restricted graph classes, such as planar or d-degenerate graphs [24, 33].

In contrast to this, our question is: what can be done in the most general case, in which the class of input graphs is unrestricted and edges may have weights? Our main result is that we can overcome the APX-hardness of STEINER TREE on one hand, and on the other hand also the W[2]-hardness for our parameter of choice p, by combining the two paradigms of approximation and parametrization. This relatively new and growing area has gained quite a bit of attention recently (see e.g., [3, 7, 9, 10, 11, 17, 20, 26, 27, 28, 29, 32, 34]). We show that there is an *efficient parameterized approximation scheme* (EPAS), which for any $\varepsilon > 0$ computes a $(1 + \varepsilon)$-approximation in time $f(p, \varepsilon) \cdot n^{O(1)}$ for a computable function f independent of n. Note that here we consider the approximation factor of the algorithm as a parameter as well, which accounts for the "efficiency" of the approximation scheme (analogous to an *efficient polynomial time approximation scheme* or EPTAS).

Theorem 1. *There is an algorithm for* STEINER TREE, *which given an edge-weighted undirected graph $G = (V, E)$, terminal set $R \subseteq V$, and integer p, computes a $(1 + \varepsilon)$-approximation to the optimum Steiner tree $T \subseteq G$ in time $2^{O(p^2/\varepsilon^4)} \cdot n^{O(1)}$, if $p \geq |V(T) \setminus R|$ and $\varepsilon > 0$.* [1]

Many variants of the STEINER TREE problem exist, and we explore the applicability of our techniques to some common ones. For the DIRECTED STEINER TREE problem the aim is to compute an *arborescence*, i.e., a directed graph obtained by orienting the edges of a tree so that exactly one vertex called the *root* has in-degree zero (which means that all vertices are reachable from the root). More concretely, the input consists of a directed graph $G = (V, A)$ with arc weights $w(a) \in \mathbb{R}_0^+$ for every $a \in A$, a terminal set $R \subseteq V$, and a specified terminal $r \in R$. A *Steiner arborescence* is an arborescence in G with root r containing all terminals R. The objective is to find a Steiner arborescence $T \subseteq G$ minimizing the weight $\sum_{a \in A(T)} w(a)$. This problem is notoriously hard to approximate: no $O\left(\log^{2-\varepsilon}(n)\right)$-approximation exists unless NP $\subseteq$ ZTIME$(n^{\text{polylog}(n)})$ [22]. But even for the UNWEIGHTED DIRECTED STEINER TREE problem in which each arc has unit weight, a fairly simple reduction from the SET COVER problem implies that no $O(\log n)$-approximation algorithm is possible unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$ [19, 22]. At the same time, even UNWEIGHTED DIRECTED STEINER TREE is W[2]-hard for our considered parameter p [24, 30], just as the undirected case. For this reason, all previous results have focused on restricted inputs: Jones et al. [24] prove that when combining the parameter p with the size of the largest excluded topological minor of the input graph, UNWEIGHTED DIRECTED STEINER TREE is FPT. They also show that if the input graph is acyclic and d-degenerate, the problem is FPT for the combined parameter p and d.

Our focus again is on general unrestricted inputs. We are able to leverage our techniques to

the unweighted directed setting, and obtain an EPAS, as summarized in the following theorem. Here an optimum Steiner arborescence is one containing the minimum number of arcs.

Theorem 2. *There is an algorithm for* UNWEIGHTED DIRECTED STEINER TREE, *which given an unweighted directed graph* $G = (V, A)$, *terminal set* $R \subseteq V$, *root* $r \in R$, *and integer* p, *computes a* $(1 + \varepsilon)$-*approximation to the optimum Steiner arborescence* $T \subseteq G$ *in time* $2^{p^2/\varepsilon} \cdot n^{O(1)}$, *if* $p \geq |V(T) \setminus R|$ *and* $\varepsilon > 0$. [1]

Can our techniques be utilized for the even more general case when arcs have weights? Interestingly, in contrast to the above theorem we can show that in general the DIRECTED STEINER TREE problem most likely does not admit such approximation schemes, even when allowing "non-efficient" runtimes of the form $f(p, \varepsilon) \cdot n^{g(\varepsilon)}$ for any computable functions f and g. This follows from the next theorem, since setting ε to any constant, the existence of such a $(1 + \varepsilon)$-approximation algorithm would imply $\mathsf{W}[1] = \mathsf{FPT}$.

Theorem 3. *For any constant* α, *it is* $\mathsf{W}[1]$-*hard to compute an* α-*approximation of the optimum Steiner arborescence* T *for* DIRECTED STEINER TREE *parameterized by* $p = |V(T) \setminus R|$, *if the input graph is arc-weighted.*

Other common variants of STEINER TREE include the PRIZE COLLECTING STEINER TREE and STEINER FOREST problems. The latter takes as input an edge-weighted undirected graph $G = (V, E)$ and a list $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$ of terminal pairs, i.e., $R = \{s_i, t_i \mid 1 \leq i \leq k\}$. A *Steiner forest* is a forest F in G for which each $\{s_i, t_i\}$ pair is in the same connected component, and the objective is to minimize the total weight of the forest F. For this variant it is not hard to see that parametrizing by $p = |V(F) \setminus R|$ cannot yield any approximation scheme, as a simple reduction from STEINER TREE shows that the problem is APX-hard even if the input has no Steiner vertices (see Section 2.1). For the PRIZE COLLECTING STEINER TREE problem, the input is again a terminal set in an edge-weighted graph, but the terminals have additional costs. A solution tree is allowed to leave out a terminal but has to pay its cost in return (cf. [35]). It is also not hard to see that this problem is APX-hard, even if there are no Steiner vertices at all. These simple results show that our techniques to obtain approximation schemes reach their limit quite soon: with the exception of UNWEIGHTED DIRECTED STEINER TREE, most common variants of STEINER TREE seem not to admit approximation schemes for our parameter p. We are however able to generalize our EPAS to STEINER FOREST if we combine p with the number c of connected components in the optimum solution. In fact, our main result of Theorem 1 is a corollary of the next theorem, using only the first part of the above reduction from STEINER TREE (cf. Section 2.1). Due to this, it is not possible to have a parametrized approximation scheme for the parameter c alone, as such an algorithm would imply a polynomial time approximation scheme for the APX-hard STEINER TREE problem. Hence the following result necessarily needs to combine the parameters p and c.

Theorem 4. *There is an algorithm for* STEINER FOREST, *which given an edge-weighted undirected graph* $G = (V, E)$, *a list* $\{s_1, t_1\}, \ldots, \{s_k, t_k\} \subseteq V$ *of terminal pairs, and integers* p, c, *computes a* $(1 + \varepsilon)$-*approximation to the optimum Steiner forest* $F \subseteq G$ *in time* $(2c)^{O((p+c)^2/\varepsilon^4)} \cdot n^{O(1)}$, *if* $p \geq |V(F) \setminus R|$ *where* $R = \{s_i, t_i \mid 1 \leq i \leq k\}$, *the number of connected components of* F *is at most* c, *and* $\varepsilon > 0$. [1]

A topic tightly connected to parameterized algorithms is kernelization. We here use the framework of Lokshtanov et al. [28], who also give a thorough introduction to the topic (see Section 2.2 for some formal definitions). Loosely speaking, a *kernelization algorithm* runs in polynomial time, and, given an instance of a parameterized problem, computes another instance of the same problem, such that the size of the latter instance is at most $f(p)$ for some computable function f in the parameter p of the input instance. The computed instance is called the *kernel*, and for an optimization problem it must be possible to convert an optimum solution to the kernel into an optimum solution to the input instance.

---

[1] If the input to this optimization problem is malformed (e.g., if p is smaller than the number of Steiner vertices of the optimum) then the output of the algorithm can be arbitrary (cf. [28])

A fundamental result of parameterized complexity says that a problem is FPT if and only if it has a kernelization algorithm [13]. This means that for our parameter p, most likely STEINER TREE does not have a kernelization algorithm, as it is W[2]-hard. For this reason, the focus of kernelization results have previously again shifted to special cases. By a folklore result, STEINER TREE is FPT for our parameter p if the input graph is planar (cf. [24]). Of particular interest are *polynomial kernels*, which have size polynomial in the input parameter. The idea is that computing the kernel in this case is an efficient preprocessing procedure for the problem, such that exhaustive search algorithms can be used on the kernel. Suchý [33] proved that UNWEIGHTED STEINER TREE parameterized by p admits a polynomial kernel if the input graph is planar.

Our aspirations again are to obtain results for inputs that are as general as possible, i.e., on unrestricted edge-weighted input graphs. We prove that STEINER TREE has a polynomial *lossy* kernel, despite the fact that the problem is W[2]-hard: an α-*approximate kernelization algorithm* is a kernelization algorithm that computes a new instance for which a given β-approximation can be converted into an $\alpha\beta$-approximation for the input instance in polynomial time. The new instance is now called a *(polynomial) approximate kernel*, and its size is again bounded as a function (a polynomial) of the parameter of the input instance.

Just as for our parameterized approximation schemes in Theorems 1 and 4, we prove the existence of a lossy kernel for STEINER TREE by a generalization to STEINER FOREST where we combine the parameter p with the number c of connected components in the optimum solution. Also, our lossy kernel can approximate the optimum arbitrarily well: we prove that for our parameter the STEINER FOREST problem admits a *polynomial size approximate kernelization scheme* (PSAKS), i.e., for every $\varepsilon > 0$ there is a $(1 + \varepsilon)$-approximate kernelization algorithm that computes a polynomial approximate kernel. An easy corollary then is that STEINER TREE parametrized only by p also has a PSAKS, by setting $c = 1$ in Theorem 5 and using the above mentioned reduction from STEINER TREE to STEINER FOREST (cf. Section 2.1).

Theorem 5. *There is a $(1 + \varepsilon)$-approximate kernelization algorithm for STEINER FOREST, which given an edge-weighted undirected graph $G = (V, E)$, a list $\{s_1, t_1\}, \ldots, \{s_k, t_k\} \subseteq V$ of terminal pairs, and integers p, c, computes an approximate kernel of size $((p + c)/\varepsilon)^{2^{O(1/\varepsilon)}}$, if for the optimum Steiner forest $F \subseteq G$, $p \geq |V(F) \setminus R|$ where $R = \{s_i, t_i \mid 1 \leq i \leq k\}$, the number of connected components of F is at most c, and $\varepsilon > 0$.* [1]

Analogous to approximation schemes, it is possible to distinguish between efficient and non-efficient kernelization schemes: a PSAKS is *efficient* if the size of the approximate kernel is bounded by $f(\varepsilon) \cdot p^{O(1)}$, where p is the parameter and f is a computable function independent of p. Our bound on the approximate kernel size in Theorem 5 implies that we do *not* obtain an efficient PSAKS for either STEINER FOREST or STEINER TREE. This is in contrast to the existence of efficient approximation schemes for the same parameters in Theorems 1 and 4. We leave open whether an efficient PSAKS can be found in either case. Interestingly, we also do not obtain any PSAKS for the UNWEIGHTED DIRECTED STEINER TREE problem, even though by Theorem 2 an EPAS exists. We leave open whether a PSAKS can be found for this variant as well.

1.1 Used techniques

Our algorithms are based on the intuition that a Steiner tree containing only few Steiner vertices but many terminals must either contain a large component induced by terminals, or a Steiner vertex with many terminal neighbours forming a large star. A high-level description of our algorithms for UNWEIGHTED DIRECTED STEINER TREE and STEINER FOREST therefore is as follows. In each step a tree is found in the graph in polynomial time, which connects some terminals using few Steiner vertices. We save this tree as part of the approximate solution and then contract it in the graph. The vertex resulting from the contraction is declared a terminal and the process repeats for the new graph. Previous results [24, 33] have also built on this straightforward procedure in order to obtain FPT algorithms and polynomial kernels for special cases of UNWEIGHTED DIRECTED STEINER TREE and UNWEIGHTED STEINER TREE. In particular, in the unweighted undirected

setting it is a well-known fact (cf. [33]) that contracting an adjacent pair of terminals is always a safe option, as there always exists an optimum Steiner tree containing this edge. However this immediately breaks down if the input graph is edge-weighted, as an edge between terminals might be very costly and should therefore not be contained in any (approximate) solution.

Instead we employ more subtle contraction rules, which use the following intuition. Every time we contract a tree with ℓ terminals we decrease the number of terminals by $\ell - 1$ (as the vertex arising from a contraction is a terminal). Our ultimate goal would be to reduce the number of terminals to one—at this point, the edges that we contracted during the whole run connect all the terminals. Decreasing the number of terminals by one can therefore be seen as a "unit of work". We will pick a tree with the lowest cost per unit of work done, and prove that as long as there are sufficiently many terminals left in the graph, these contractions only lose an ε-factor compared to the optimum. As soon as the number of terminals falls below a certain threshold depending on the given parameter, we can use an FPT algorithm computing the optimum solution. This algorithm is parametrized by the number of terminals, which now is bounded by our parameter. For the variants of STEINER TREE considered in our positive results, such an FPT algorithm can easily be obtained from the ones for STEINER TREE [2, 16]. Adding this exact solution to the previously contracted trees gives a feasible solution that is a $(1 + \varepsilon)$-approximation.

Each step in which a tree is contracted in the graph, can be seen as a *reduction rule* as typically used for kernelization algorithms. Typically, a proof for a kernelization algorithm will define a set of reduction rules and then show that the instance resulting from applying the rules exhaustively has size bounded as a function in the parameter. To obtain an α-approximate kernelization algorithm, additionally it is shown that each reduction rule is α-*safe*. Roughly speaking, this means that at most a factor of α is lost when applying any number of α-safe reduction rules (see Section 2.2 for formal definitions).

Contracting edges in a directed graph may introduce new paths, which did not exist before. Therefore, for the UNWEIGHTED DIRECTED STEINER TREE problem, we need to carefully choose the arborescence to contract. In order to prove Theorem 2 we show that each contraction is a $(1 + \varepsilon)$-safe reduction rule. However, the total size of the graph resulting from exhaustively applying the contractions is not necessarily bounded as a function of our parameter. Thus we do not obtain an approximate kernel.

For STEINER FOREST the situation is in a sense the opposite. Choosing a tree to contract follows a fairly simple rule. On the downside however, the contractions we perform are not necessarily $(1 + \varepsilon)$-safe reduction rules. In fact there are examples in which a single contraction will lose a large factor compared to the optimum cost. We are still able to show however, that after performing all contractions exhaustively, any β-approximation to the resulting instance can be converted into a $(1 + \varepsilon)\beta$-approximation to the original input instance. Even though the total size of the resulting instance again cannot be bounded in terms of our parameter, for STEINER FOREST we can go on to obtain a PSAKS. For this we utilize a result of Lokshtanov et al. [28], which shows how to obtain a PSAKS for STEINER TREE if the parameter is the number of terminals. This result can be extended to STEINER FOREST, and since our instance has a number of terminals bounded in our parameter after applying all contractions, we obtain Theorem 5.

Finally, to obtain our inapproximability result of Theorem 3, we use a reduction from the DOMINATING SET problem. It was recently shown by Chen and Lin [9] that this problem does not admit parameterized α-approximation algorithms for any constant α, if the parameter is the solution size, unless $\mathsf{W}[1] = \mathsf{FPT}$. We are able to exploit this to also show that no such algorithm exists for DIRECTED STEINER TREE with edge weights, under the same assumption.

1.2 Related work

As the STEINER TREE problem and its variants have been studied since decades, the literature on this topic is huge. We only present a selection of related work here, that was not yet mentioned above.

For planar graphs [5] it was shown that an EPTAS exists for STEINER TREE. For STEINER FOREST a 2-approximation can be computed in polynomial time on general inputs [1], but an EPTAS also exists if the input is planar [18]. If the UNWEIGHTED STEINER TREE problem is parametrized by

the solution size, it is known [14] that no polynomial (exact) kernel exists, unless $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$. If the input is restricted to planar or bounded-genus graphs it was shown that polynomial kernels do exist for this parametrization [31]. It was later shown [33] that for planar graphs this is even true for our parameter p. For the DIRECTED STEINER TREE problem it is a long standing open problem whether a polylogarithmic approximation can be computed in polynomial time. It is known that an $O\left(|R|^\varepsilon\right)$-approximation can be computed in polynomial time [8], and an $O\left(\log^2 n\right)$-approximation in quasi-polynomial time [8]. A recent result [21] considers generalizations of DIRECTED STEINER TREE and characterizes which of these problems are FPT and which are $\mathsf{W}[1]$-hard for parameter $|R|$.

2 Preliminaries

2.1 Reducing Steiner tree to Steiner forest

We may reduce the STEINER TREE problem to STEINER FOREST by picking an arbitrary terminal r of the STEINER TREE instance, and for every other terminal v of this instance, introduce a terminal pair $\{v, r\}$ for STEINER FOREST.

If we want to construct an instance without Steiner vertices, we can add a new vertex w' for every Steiner vertex w of STEINER TREE and add an edge ww' of cost 0. Additionally we introduce a terminal pair $\{w, w'\}$ to our STEINER FOREST instance. Hence $\{s_i, t_i \mid 1 \le i \le k\} = V$ in the constructed STEINER FOREST instance (i.e., there are no Steiner vertices), but an optimum Steiner forest in the constructed graph costs exactly as much as an optimum Steiner tree in the original graph. As STEINER TREE is APX-hard, the same is true for STEINER FOREST, even if all vertices are terminals.

2.2 Lossy kernels

We give a brief introduction to the lossy kernel framework as introduced by Lokshtanov et al. [28]. See the latter for a thorough introduction to the topic.

For an optimization problem, a *polynomial time pre-processing algorithm* is a pair of polynomial time algorithms: the *reduction algorithm* $\mathcal{R}$ and the *solution lifting algorithm* $\mathcal{L}$. The former takes an instance I with parameter p of a given problem as input, and outputs another instance I' with parameter p'. The solution lifting algorithm $\mathcal{L}$ converts a solution for the instance I' to a solution of the input instance I: given a solution s' to I', $\mathcal{L}$ computes a solution s for I, such that s is optimal for I if s' is optimal for I'. If additionally the output of $\mathcal{R}$ is bounded as a function of p, i.e., when $|I'| + p' \le f(p)$ for some computable function f independent of $|I|$, then the pair given by $\mathcal{R}$ and $\mathcal{L}$ is called a *kernelization algorithm*, and I' together with parameter p' is the *kernel*. If the reduction and solution lifting algorithms get an input that is not an instance of the problem (for example if the parameter does not correctly describe some property of the optimum solution), then the outputs of the algorithms are undefined and can be arbitrary.

An α-*approximate polynomial time pre-processing algorithm* is again a pair of a reduction algorithm $\mathcal{R}$ and a solution lifting algorithm $\mathcal{L}$, both running in time polynomial in the input size. The reduction and solution lifting algorithms are as before, but there is a different property on the output of the latter: if the given solution s' to the instance I' computed by $\mathcal{R}$ is a β-approximation, then the output of $\mathcal{L}$ is a solution s that is an $\alpha\beta$-approximation for the original instance I. Analogous to before, an α-*approximate kernelization algorithm* is an α-approximate polynomial time pre-processing algorithm for which the size of the output of the reduction algorithm is bounded in terms of p only. The output of $\mathcal{R}$ is in this case called an *approximate kernel*, and it is *polynomial* if its size is bounded by a polynomial in p.

In the context of lossy kernels a *reduction rule* is a reduction algorithm $\mathcal{R}$. It is called α-*safe* if a solution lifting algorithm $\mathcal{L}$ exists, which together with $\mathcal{R}$ form a *strict* α-approximate polynomial time pre-precessing algorithm. This means that if s' is a β-approximation for the instance computed by $\mathcal{R}$, then $\mathcal{L}$ computes a $(\max\{\alpha; \beta\})$-approximation s for the input instance. As shown in [28], the advantage of considering this stricter definition is that, as usual, reduction rules can be applied exhaustively, until a stable point is reached in which none of the rules would change the instance

any longer: the algorithm resulting from applying these rules together with their corresponding solution lifting algorithms, forms a strict α-approximate polynomial time pre-precessing algorithm (which is not necessarily the case when using the non-strict definition; see [28]).

3 The weighted undirected Steiner forest and Steiner tree problems

In this section we describe an approximate polynomial time preprocessing algorithm that returns an instance of STEINER FOREST containing at most $\mathrm{O}\left((p+c)^2/\varepsilon^4\right)$ terminals if the optimum Steiner forest has at most p Steiner vertices and at most c connected components. We can use this algorithm in two ways. Either we can proceed with a kernelization derived from Lokshtanov et al. [28] and obtain a polynomial size lossy kernel (Theorem 5), or we can run an exact FPT algorithm derived from Dreyfus and Wagner [16] on the reduced instance, obtaining an EPAS running in single exponential time with respect to the parameters (Theorems 1 and 4). In both cases we use the combined parameter (p, c).

	STEINER FOREST
Input:	A graph $G = (V, E)$, with edge weights $w(e) \in \mathbb{R}^+$ for each $e \in E$, and a list $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$ of pairs of terminals.
Solution:	A Steiner forest $F \subseteq G$ containing an s_i-t_i path for every $i \in [k]$

We first rescale all weights so that every edge has weight strictly greater than 1. Then, in each step of our algorithm we pick a star, add it to the solution, and contract the star in the current graph. We repeat this procedure until the number of terminals falls below a specified bound depending on ε, p, and c. To describe how we pick the star to be contracted in each step, we need to introduce the *ratio* of a star. Let C be a set of edges of a star, i.e., all edges of C are incident to a common vertex which is the *center* of the star, and denote by Q the set of terminals incident to C. Provided $|Q| \geq 2$, we define the *ratio* of C as $w(C)/(|Q| - 1)$, where $w(C) = \sum_{e \in C} w(e)$. Note that we allow C to contain only a single edge if it joins two terminals. Observe also that due to rescaling of edge weights each star has ratio strictly greater than 1.

In every step, our algorithm contracts a star with the best available ratio (i.e., the lowest ratio among all stars connecting at least two terminals). Due to the following lemma, the star with best ratio has a simple form: it consists of the cheapest i edges incident to its center vertex and some terminal. As there are n possible center vertices and at most n incident edges to each center which can be sorted in time $\mathrm{O}\left(n \log n\right)$, the best ratio star can be found in time $\mathrm{O}\left(n^2 \log n\right)$. Later we show that there is a star with at least two terminals in every step, provided that the number of terminals is more that p.

Lemma 6. *Let v be a vertex and denote by $q_1, q_2, \ldots$ the terminals adjacent to v, where $w(vq_1) \leq w(vq_2) \leq \cdots$, i.e., the terminals are ordered non-decreasingly by the weight of the corresponding edge vq_i. The star with the best ratio having v as its center has edge set $\{vq_1, vq_2, \ldots, vq_\ell\}$ for some ℓ.*

Proof. Let C be an edge set of a star with center vertex v. First note that if this star contains a Steiner vertex w as a leaf, vw can be removed from C in order to decrease the ratio $w(C)/(|Q| - 1)$, since only the terminals Q of the star are counted in the denominator. Also if C does not contain some edge vq_i but an edge vq_j with $j > i$, then we may switch the edge vq_j for vq_i in C in order to optimize the ratio: the denominator stays the same, but the numerator cannot increase, as the terminals $q_1, q_2, \ldots$ are ordered non-decreasingly according to the weights vq_i. $\qquad \square$

To analyse our algorithm we need to keep track of the different graphs resulting from each contraction step t. Initially we set G_0 to the input graph, and in each step $t \geq 0$ we obtain a new graph G_{t+1} from G_t by contracting a set of edges C_t in G_t, such that C_t forms a star of minimum ratio in G_t. That is, we obtain G_{t+1} from G_t by identifying all vertices incident to edges in C_t, removing all resulting loops, and among the resulting parallel edges we delete all but the lightest one with respect
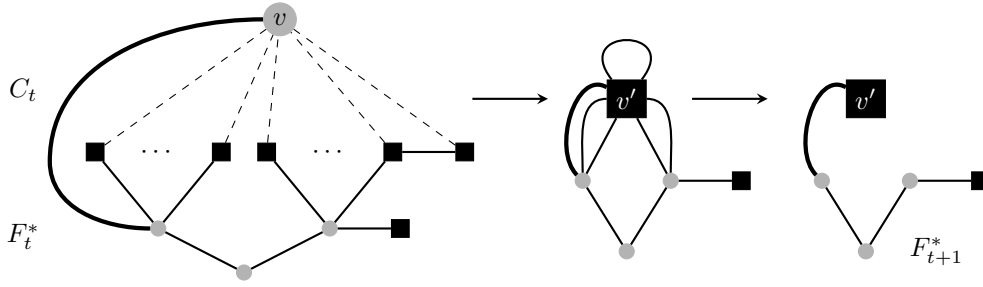
Figure 1: An example of creating F_{t+1}^* from F_t^* after a contraction C_t. Each edge in C_t (dashed) may or may not be in F_t^*. The thick edge cannot be in D_t because it is not incident to any terminal.

to their weights. We also adjust the terminal pairs in a natural way: let v be the vertex of G_{t+1} resulting from contracting C_t. If G_t had a terminal pair $\{s, t\}$ such that s is incident to some edge of C_t while t is not, then we introduce the terminal pair $\{v, t\}$ for G_{t+1}. Also every terminal pair $\{s, t\}$ of G_t for which neither s nor t is incident to any edge of C_t is introduced as a terminal pair of G_{t+1}. Any terminal pair for which both s and t are incident to edges of C_t is going to be connected by a path in the computed solution, as it will contain C_t. Hence, such a terminal pair can be safely removed.

The algorithm stops contracting best-ratio stars when there are less than τ terminals left; the exact value of τ depends on p, c, and the desired approximation factor, it will satisfy $\tau > p$ and we specify it later. If the algorithm stops in step $\tilde{t}$, the solution lifting algorithm takes a feasible solution F of $G_{\tilde{t}}$ and returns the union of F and $\bigcup_{t=0}^{\tilde{t}} C_t$. Such a solution is clearly feasible, since we adapted the terminal pairs accordingly after each contraction.

For the purpose of analysis, we consider a solution in the current graph G_t that originates from an optimal solution of the original instance G_0, but may contain edges that are heavier than those in G_t. More concretely, denote by F_0^* an optimal solution in G_0, i.e., F_0^* is a Steiner forest containing every s_i-t_i path with at most p Steiner vertices and at most c connected components. Given F_t^* for $t \geq 0$, we modify this solution to obtain a new feasible solution F_{t+1}^* on the terminal pairs of G_{t+1}. Note that the edges of the contracted star C_t might not be part of F_t^*. We still mimic the contraction of the star in F_t^*: to obtain F_{t+1}^* from F_t^*, we identify all leaves of C_t (which are terminals by Lemma 6 and thus part of the solution F_t^*) and possibly also the center v of C_t if it is in F_t^*. This results in a vertex v'. We now want to delete edges incident to v' in such a way that we are left with an acyclic feasible solution. If we delete an inclusion-wise minimal feedback edge set, we clearly get a feasible solution. Let Q_t denote the set of terminals incident to C_t. We choose a feedback edge set D_t for which every edge was incident to a vertex of Q_t before the contraction in F_t^*, i.e., an edge of G_t corresponding to an edge of D_t never connects two Steiner vertices. Note that such an inclusion-wise minimal feedback edge set always exists: if we delete all edges of F_t^* incident to Q_t except C_t and then contract C_t, we get an acyclic graph. See Fig. 1 for an illustration.

The resulting graph is F_{t+1}^*, which now forms a forest connecting all terminal pairs of G_{t+1}. Note that for each edge in F_{t+1}^* there is a corresponding edge in G_{t+1}, which however may be lighter in G_{t+1}, as from each bundle of parallel edges in G_t we keep the lightest one, but this edge may not exist in F_t^*.

We now observe that there is always a star with at least two terminals and thus the algorithm always selects some star.

Lemma 7. *Provided that there are at least $\tau > p$ terminals in G_t, there is a star with a least two terminals in G_t.*

Proof. Note that it is sufficient to find such a star in F_t^* as edges in F_t^* are also present in G_t (even if their weight may be smaller). If there is an edge e between two terminals in F_t^*, then we are done as e itself is a star. Otherwise, all terminals are incident to Steiner vertices only. Thus there must be a Steiner vertex incident to at least two terminals in F_t^*, since F_t^* contains at most p Steiner vertices but more than p terminals. □

To show that our algorithm only loses an ε-factor compared to the cost of the optimum solution F_0^*, we will compare the cost of the edges C_t contracted by our algorithm to the set $D_t = E(F_{t+1}^*) \setminus E(F_t^*)$ of deleted edges of F_t^*. Note that there are at least $|Q_t| - c$ edges in D_t, since we contracted Q_t terminals in the forest F_t^* with at most c connected components to obtain F_{t+1}^*, and a forest on n vertices and k components has $n - k$ edges. We decrease the number of vertices of F_t^* by at least $|Q_t| - 1$ (one more if the center of the star with edge set C_t was a Steiner vertex present in F_t^*), and we decrease the number of components by at most $c - 1$. Note also that for any two time steps $t \neq t'$, the sets D_t and $D_{t'}$, but also the sets C_t and $C_{t'}$, are disjoint. Thus if $w(C_t) \leq (1 + \varepsilon)w(D_t)$ for every t, then our algorithm computes a $(1 + \varepsilon)$-approximation. Unfortunately, this is not always the case: there are contractions for which this condition does not hold (see Fig. 2) and we have to account for them differently.

Definition 8. If $w(C_t) \leq (1 + \varepsilon)w(D_t)$ we say that the contracted edge set C_t in step t is *good*; otherwise C_t is *bad*. Moreover, if F_t^* has strictly more components than F_{t+1}^*, we say that C_t is *multiple-component*, otherwise it is *single-component*.

Our goal is to show that the total weight of bad contraction is bounded by an ε-fraction of the weight of an optimum solution. We start by proving that if the set Q_t of terminals in C_t is sufficiently large, then the contraction is good. We define

$$\lambda := \frac{(1 + \varepsilon)(p + c)}{\varepsilon}.$$

Lemma 9. If $|Q_t| \geq \lambda$, then the contracted edge set C_t is good.

Proof. For brevity, we drop the index t. Let $r = w(C)/(|Q| - 1)$ be the ratio of the contracted star, and let ℓ' be the number of deleted edges in D that connect two terminals. Note that any such edge has weight at least r, since it spans a star with two terminals, which has ratio equal to its weight, and since each edge in F^* (of which D is a subset) can only be heavier than the corresponding edge in the current graph G.

Let $u_1, \ldots, u_q$ be the Steiner vertices adjacent to edges in D, and let ℓ_i be the number of edges in D incident to one such Steiner vertex u_i (see Fig. 3). Since D is a feedback edge set in which any edge was incident to a terminal in Q before the contraction, there is no edge in D which connects two Steiner vertices. Consider the star spanned by the ℓ_i edges of D incident to u_i. If $\ell_i \geq 2$, the ratio of this star is at least r, since its edges are at least as heavy as the corresponding edges in G and the algorithm chose the star with minimum ratio in G. Thus, the weight of edges in D incident to u_i is at least $r(\ell_i - 1)$. In the case where $\ell_i = 1$, the lower bound $r(\ell_i - 1) = 0$ on the weight holds trivially.

Any edge in D not incident to any Steiner vertex u_i connects two terminals. Therefore, we have $\ell' + \sum_{i=1}^q \ell_i = |D|$ as any edge in D is incident to a vertex in Q and we thus do not count any edge twice. Also recall that $|D| \geq |Q| - c$. Since F contains at most p Steiner vertices we have $q \leq p$, and we obtain

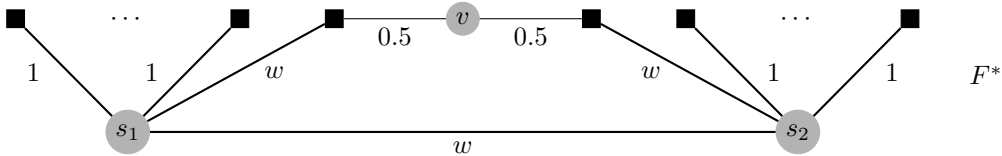$$w(D) \geq r\ell' + \sum_{i=1}^q r(\ell_i - 1) = r\left(\ell' + \sum_{i=1}^q \ell_i - q\right) \geq r(|Q| - p - c).$$

Figure 2: An example of a bad contraction. The numbers of terminals can be arbitrarily large and the weight w can be arbitrarily small. The star centered at v has ratio 1 while every star centered either at s_1 or s_2 has ratio slightly more than 1. By contracting the star centered at v we create a cycle containing only edges of weight w. Thus, for a sufficiently small value of w the contraction cannot be charged.
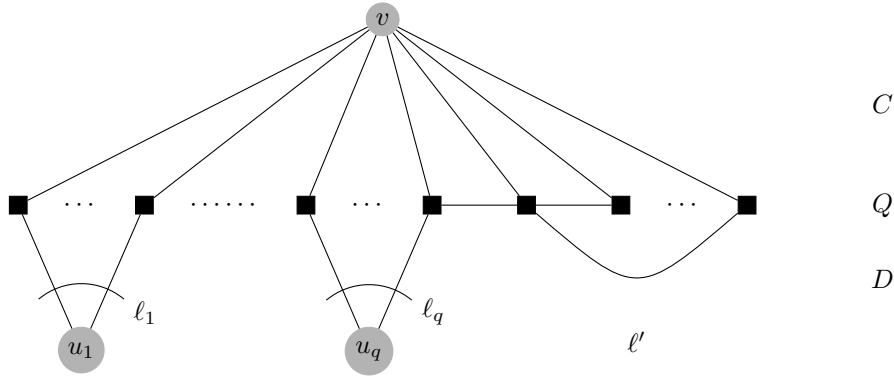
Figure 3: The contracted star C and a part of the optimal solution spanned by the terminals Q of the star C.

Finally, using $|Q| \geq \lambda$ we bound $w(C)$ by $(1+\varepsilon)w(D)$ as follows:

$$(1+\varepsilon)w(D) \geq (1+\varepsilon)r(|Q| - p - c) = r(|Q| - 1) + r\big(\varepsilon|Q| - (1+\varepsilon)(p+c) + 1\big)$$
$$\geq w(C) + r\left(\varepsilon\frac{(1+\varepsilon)(p+c)}{\varepsilon} - (1+\varepsilon)(p+c)\right) = w(C). \quad \square$$

Note that there may be a lot of contractions with $|Q| < \lambda$. However, we show that only a bounded number of them is actually bad. The key idea is to consider contractions with ratio in some interval $((1+\delta)^i; (1+\delta)^{i+1}]$ for some $\delta > 0$ and integer i. Due to the rescaling of weights every star belongs to an interval with $i \geq 0$. The following crucial lemma of our analysis shows that the number of bad single-component contractions in each such interval is bounded in terms of p and ε, if δ is a function of ε. In particular, let $\delta := \sqrt{1+\varepsilon} - 1$, so that $(1+\delta)^2 = 1+\varepsilon$. We call an edge set C with ratio r in the i-th interval, i.e., with $r \in ((1+\delta)^i; (1+\delta)^{i+1}]$, an i-contraction, and define

$$\kappa := \frac{(1+\delta)p}{\delta} + p.$$

Lemma 10. *For any integer i the number of bad single-component i-contractions is at most κ.*

Proof. Suppose for a contradiction that the number of bad single-component i-contractions is larger than κ. Let $\tilde{t}$ be the first step with a bad single-component i-contraction, i.e., $\tilde{t}$ is the minimum among all t for which $w(C_t) > (1+\varepsilon)w(D_t)$ and $w(C_t)/(|Q_t| - 1) \in ((1+\delta)^i; (1+\delta)^{i+1}]$ and the contraction is single-component. The plan is to show that at step $\tilde{t}$ there is a "light" star in $G_{\tilde{t}}$ with ratio at most $(1+\delta)^i$ and consequently the algorithm would do a j-contraction for some $j < i$. This leads to a contradiction, since we assumed that in step $\tilde{t}$ the contraction has ratio in interval i. Note that it is sufficient to find such a light star in $F_{\tilde{t}}^*$ as for each edge in $F_{\tilde{t}}^*$ there is an edge in the graph $G_{\tilde{t}}$ between the same vertices of the same weight or even lighter.

We claim that for each step t in which the algorithm does a bad single-component i-contraction there is an edge $e_t \in D_t$ with weight at most $(1+\delta)^{i-1}$. We have $w(C_t) > (1+\varepsilon)w(D_t)$ as C_t is bad and $w(C_t) \leq (1+\delta)^{i+1}(|Q_t| - 1)$ as the ratio of C_t is in interval i. Putting it together and using the definition of δ we obtain

$$w(D_t) < \frac{(1+\delta)^{i+1}}{1+\varepsilon}(|Q_t| - 1) = (1+\delta)^{i-1}(|Q_t| - 1).$$

Because C_t is single-component, we have $|D_t| \geq |Q_t| - 1$ and therefore there is an edge $e_t \in D_t$ with weight at most $(1+\delta)^{i-1}$, which proves the claim.

Note that the edge e_t also exists at time step $\tilde{t}$, as $\tilde{t} \leq t$ and F_t^* is obtained from $F_{\tilde{t}}^*$ by a sequence of edge contractions and deletions. At time $\tilde{t}$ it cannot be that e_t connects two terminals,

10

since we assume that the algorithm picked a star of ratio more than $(1 + \delta)^i$ in step $\tilde{t}$ (recall that each edge connecting two terminals is a star with ratio equal to its weight). It may happen though that e_t connects two Steiner vertices in step $\tilde{t}$. We discard any such edge e_t that connects two Steiner vertices in step $\tilde{t}$. That is, let S be the set of light edges e_t that lead between a Steiner vertex and a terminal in step $\tilde{t}$. Note that edges e_t and $e_{t'}$ for steps $t < t'$ with bad i-contractions are distinct, because $D_t \cap D_{t'} = \emptyset$ as all edges in D_t are deleted from $F_{\tilde{t}}^*$. There are at most $p - 1$ edges $e_t \notin S$ connecting two Steiner vertices in $F_{\tilde{t}}^*$, since $F_{\tilde{t}}^*$ is a forest and the optimum, from which $F_{\tilde{t}}^*$ is derived, contained at most p Steiner vertices. As we assume that there are more than κ bad single-component i-contractions, we have $|S| > \kappa - p$.

At step $\tilde{t}$ there must be a Steiner vertex v in $F_{\tilde{t}}^*$ incident to at least $|S|/p > (\kappa - p)/p \geq (1 + \delta)/\delta$ edges in S. Consider a star C with v as the center and with edges from S that are incident to v; we have $|C| \geq (1 + \delta)/\delta$. The ratio of this star is at most $|C|(1 + \delta)^{i-1}/(|C| - 1)$. Since $|C|/(|C| - 1) \leq (1 + \delta)$ (by a routine calculation) we get that the ratio of C is at most $(1 + \delta)^i$ which is a contradiction to the assumption that the algorithm does an i-contraction in step $\tilde{t}$. □

We also need a bound on number of bad multiple-component edge sets.

Lemma 11. *The number of steps t in which a bad multiple-component edge set C_t is contracted is at most $c - 1$.*

Proof. If C_t is a bad multiple-component edge set, F_{t+1}^* must have at least one component fewer than F_t^*. Since F_0^* has at most c components, the bound follows. □

We remark that the proofs of Lemmas 10 and 11 do not use that the number of terminals in a bad i-contraction is bounded by λ, as shown in Lemma 9. Instead we bound the total weight of bad contractions in terms of λ. For this let j be the largest interval of any contraction during the whole run of the algorithm, i.e., the ratio of every contracted star is at most $(1 + \delta)^{j+1}$.

Lemma 12. *The total weight of bad edge sets C_t is at most*

$$(\kappa + c) \cdot \lambda \cdot \frac{(1 + \delta)^{j+2}}{\delta} \,.$$

Proof. By Lemma 11, there are less than c bad multiple-component contractions. Each of them has at most λ terminals by Lemma 9 and has ratio at most $(1 + \delta)^{j+1}$ by the choice of j. Thus, the total weight of all bad multiple-component contractions can be bounded by $(1 + \delta)^j \cdot \lambda \cdot c$.

Note that it follows from Lemmas 9 and 10 that the total weight of bad single-component i-contractions is at most $\kappa \cdot \lambda \cdot (1 + \delta)^{i+1}$. The bound on the total weight of bad contractions follows by summing over all intervals in which the algorithm does a contraction:

$$\kappa \cdot \lambda \cdot \sum_{i \leq j} (1 + \delta)^{i+1} + c \cdot \lambda \cdot (1 + \delta)^j = \kappa \cdot \lambda \cdot \frac{(1 + \delta)^{j+2} - 1}{(1 + \delta) - 1} + c \cdot \lambda \cdot (1 + \delta)^j \leq (\kappa + c) \cdot \lambda \cdot \frac{(1 + \delta)^{j+2}}{\delta} \,.$$

□

To bound the approximation ratio, as usual we need a suitable lower bound on the cost of the optimum solution.

Lemma 13. $w(F_0^*) \geq (1 + \delta)^j \cdot (\tau - 2p - c)$.

Proof. When our algorithm contracted a star having ratio $r \geq (1 + \delta)^j$ in the largest interval j in some step t, all stars in F_t^* with Steiner vertices $v_1, \ldots, v_q$ as centers had ratios at least r. Thus if ℓ_i is the number of terminals incident to v_i in F_t^*, then these terminals together with v_i form a star of weight at least $r \cdot (\ell_i - 1)$. Similarly, all edges between terminals in F_t^* have weight at least r; let ℓ' be the number of such edges.

11

Since there are at least τ terminals in step t (otherwise the algorithm would have terminated), and at most $q-1$ of edges in F_t^* connect two Steiner vertices, we have $\ell' + \sum_{i=1}^q \ell_i \geq \tau - c - (q-1) \geq \tau - c - p$ as $p \geq q$. The total weight of edges in F_t^* is thus at least

$$\ell' r + \sum_{i=1}^q r \cdot (\ell_i - 1) \geq r \cdot (\tau - 2p - c) \geq (1+\delta)^j \cdot (\tau - 2p - c).$$

This proves the lemma as $w(F_t^*) \leq w(F_0^*)$. □

Finally, we show that the total weight of bad contracted edge sets is very small compared to the weight of the optimum, provided that there are still many terminals in the graph during the last contraction. More precisely, we set

$$\tau := (\kappa + c) \cdot \lambda \cdot \frac{(1+\delta)^2}{\varepsilon\delta} + 2p + c.$$

Lemma 14. *The total weight of bad edge sets C_t is at most $\varepsilon \cdot w(F_0^*)$.*

Proof. By Lemma 13 and using the value of τ we have

$$\varepsilon \cdot w(F_0^*) \geq \varepsilon(1+\delta)^j \cdot (\tau - 2p - c) \geq \varepsilon(1+\delta)^j \cdot (\kappa + c) \cdot \lambda \cdot \frac{(1+\delta)^2}{\varepsilon\delta} = (\kappa + c) \cdot \lambda \cdot \frac{(1+\delta)^{j+2}}{\delta},$$

which is the upper bound on the total weight of bad edges sets by Lemma 12. □

The above lemmas can now be used to prove that all the contractions put together (by scaling ε) form a $(1+\varepsilon)$-approximate pre-processing procedure (cf. Section 2.2).

Lemma 15. *The algorithm outputs an instance with $\tau \in \mathrm{O}\left((p+c)^2/\varepsilon^4\right)$ terminals and (together with the solution lifting algorithm) it is a $(1 + 2\varepsilon)$-approximate polynomial time pre-processing algorithm. In case the given p is smaller than the number of Steiner vertices in the optimum, or c is smaller than its number of connected components, the algorithm still outputs a Steiner forest, but the approximation factor may be arbitrary.*

Proof. By Lemma 7 each step of the algorithm can be executed since $\tau > p$. Thus the upper bound on the number of terminals follows directly from the description of the algorithm. To bound the running time, we already noted that finding a minimum ratio star to contract can be done in $\mathrm{O}\left(n^2 \log n\right)$ time. Since such a star with at least two vertices is contracted in each step t to form the next graph G_{t+1}, the total time used for contractions until only τ terminals are left is polynomial in n.

Let us focus on the $(1 + 2\varepsilon)$-approximate part. Let $H = G_{\tilde{t}}$ be the graph left after the last contraction step $\tilde{t}$, and let F_H be a Steiner forest for the remaining terminal pairs. The solution lifting algorithm simply adds all contracted edge sets $C_0, C_1, \dots$ to F_H in order to compute a Steiner forest F_G in the input graph G. We need to show that, if F_H is a β-approximation to the optimum F_H^* in H, the resulting forest F_G is a $((1 + 2\varepsilon)\beta)$-approximation to the optimum $F_G^* = F_0^*$ of G.

Let us call a step t of the algorithm *good* (*bad*) if the corresponding contracted edge set C_t is good (bad). As all sets C_t are disjoint, using Lemmas 9 and 14 the weight of F_G can be bounded by

$$w(F_G) = \sum_{\text{good } t} w(C_t) + \sum_{\text{bad } t} w(C_t) + w(F_H) \leq \sum_{\text{good } t} (1+\varepsilon) w(D_t) + \varepsilon \cdot w(F_G^*) + \beta \cdot w(F_H^*).$$

The forest $F_{\tilde{t}}^*$ left after the last contraction corresponds to a feasible solution in H. As the edge weights might be less expensive in H than in $F_{\tilde{t}}^*$, we have $w(F_H^*) \leq w(F_{\tilde{t}}^*)$. At the same time, the deleted sets D_t and the edges of $F_{\tilde{t}}^*$ are disjoint, so that $\sum_{\text{good } t} w(D_t) \leq \sum_t w(D_t) \leq w(F_G^*) - w(F_{\tilde{t}}^*)$. Therefore the above bound becomes

$$w(F_G) \leq (1+\varepsilon)\left(w(F_G^*) - w(F_{\tilde{t}}^*)\right) + \varepsilon \cdot w(F_G^*) + \beta \cdot w(F_{\tilde{t}}^*)$$
$$\leq (1+\varepsilon)\beta\left(w(F_G^*) - w(F_{\tilde{t}}^*) + w(F_{\tilde{t}}^*)\right) + \varepsilon \cdot w(F_G^*) \leq (1 + 2\varepsilon)\beta \cdot w(F_G^*),$$

which proves the claim. □

```
input    : undirected graph $G = (V, E)$,
            list of terminal pairs $\{s_1, t_1\}, \dots, \{s_k, t_k\}$, edge weights $w(e) \in \mathbb{R}_0^+$
output   : a forest $F \subseteq G$ such that it contains every $s_i$-$t_i$ path.
1  Function BestStar($v$)
2  |   if $v$ is a terminal then $z \leftarrow 1$
3  |   else $z \leftarrow 0$
4  |   $q_1, \dots, q_k \leftarrow$ terminals adjacent to $v$ sorted by the weight of edge $vq_i$
5  |   for $i$ in $1, \dots, k$ do
6  |   |   $r_i \leftarrow \sum_{j=1}^{i} w(vq_i)/(i + z - 1)$
7  |   return edges $\{vq_1, \dots, vq_i\}$ of star with the smallest $r_i$ /* which exists by Lemma 7 */
8  while number of terminals is at least $\tau$ do
9  |   $C \leftarrow \arg\min\{w(C_v) \mid C_v \leftarrow \text{BestStar}(v), v \in V\}$
10 |   Contract $C$
11 Run FPT algorithm parametrized by the number of terminals
```

Algorithm 1: An algorithm for solving STEINER FOREST. If we stop before Line 11 we obtain the reduced instance.

Algorithm 1 gives a pseudo-code of the resulting algorithm.

Proof of Theorem 4. Obtaining an FPT algorithm for STEINER FOREST parameterized by the number of terminals and connected components is not hard given an FPT algorithm as the one given in [16] for STEINER TREE: we only need to guess the sets of terminals that form connected components in the optimum Steiner forest. We can then invoke the algorithm of [16] on each subset to compute an optimum Steiner tree connecting it. The input to our algorithm is an integer c upper-bounding the number of components of the optimum. Thus each terminal can be in one of at most c components, so that there are $c^{|R|}$ partitions of the terminal set R that need to be considered. The algorithm of [16] runs in time $3^{|R|} \cdot n^{O(1)}$, and so this results in an algorithm with runtime $(3c)^{|R|} \cdot n^{O(1)}$ to solve STEINER FOREST. We may run this algorithm on the STEINER FOREST instance that our pre-processing algorithm of Lemma 15 computes, in order to obtain Theorem 4. □

Proof of Theorem 5. To obtain our result on lossy kernels we rely on the fact that a PSAKS exists for STEINER TREE parameterized by the number of terminals. It is known that despite being FPT [16], this problem does not admit polynomial (exact) kernels [14], unless NP $\subseteq$ coNP/Poly. However, as shown by Lokshtanov et al. [28], the Borchers and Du Theorem [4] can be reinterpreted to show that a PSAKS exists. The theorem states that for any optimum Steiner tree T on terminal set R there exists a collection of trees $T_1, \dots, T_k$, each of which contains $2^{O(1/\varepsilon)}$ terminals of R, and for which the union $\bigcup_{i=1}^{k} T_i$ is a $(1 + \varepsilon)$-approximation of T. To obtain a kernel we first preprocess the input graph by removing any vertex that is far away from any terminal and can therefore never be part of an optimum Steiner tree. For this a rough estimate of the cost of an optimum Steiner tree is needed, which can be obtained in polynomial time using existing constant approximation algorithms for STEINER TREE. The next step is to take the metric closure of the remaining graph, so that any minimum cost tree connecting $2^{O(1/\varepsilon)}$ terminals only contains $2^{O(1/\varepsilon)}$ Steiner vertices as well. We then compute an optimum Steiner tree for each subset of R of size $2^{O(1/\varepsilon)}$ and take their union. Within this union exists the $(1 + \varepsilon)$-approximate Steiner tree due to the Borchers and Du Theorem [4], and the total number of vertices in this union is $|R|^{2^{O(1/\varepsilon)}}$. It is still necessary to round the edge weights of the resulting instance. It can be shown [28] that this is possible using at most $O(\log(|R|) + \log(1/\varepsilon))$ bits, and so that any Steiner tree in the resulting approximate kernel corresponds to a Steiner tree in the original instance that costs at most a factor of $(1 + \varepsilon)$ more.

Obtaining a PSAKS for STEINER FOREST can be done in essentially the same way: the Borchers and Du Theorem [4] can be applied to each tree in the optimum Steiner forest, and a rough estimate of the cost of an optimum Steiner forest can be obtained via the polynomial time 2-approximation of Agrawal et al. [1]. Using the same steps as for STEINER TREE, if the STEINER FOREST instance

contains $|R|$ terminal pairs, we obtain an approximate kernel of size $|R|^{2^{\mathrm{O}(1/\varepsilon)}}$ in which every optimum Steiner forest corresponds to a Steiner forest in the original instance of cost at most a factor of $(1 + \varepsilon)$ more. Since the number of terminals in the instance that we obtain after exhaustively applying our contractions is bounded in terms of our parameters p, c, and ε, we may compute an approximate kernel for these parameters by Lemma 15. This implies Theorem 5. $\quad\square$

4 The unweighted directed Steiner tree problem

In this section we provide an EPAS for the UNWEIGHTED DIRECTED STEINER TREE problem, in which each arc has unit weight.

<div>

UNWEIGHTED DIRECTED STEINER TREE

Input: A directed graph $G = (V, A)$, and a set R of terminals with a root terminal r.
Solution: A Steiner arborescence $T \subseteq G$ containing a directed path from r to each terminal $v \in R$.

</div>

The idea behind our algorithm given in this section is to reduce the number of terminals of the input instance via a set of reduction rules. That is, we would like to reduce the input graph G to a graph G', and prove that the number of terminals in G' is bounded by a function of our parameter p and the approximation ratio $(1+\varepsilon)$. On the reduced instance we can use the algorithm of Björklund et al. [2] to obtain an optimum solution.

Our first reduction rule represents the idea that a terminal in the immediate neighborhood of the root can be contracted to the root. Observe that in this case our algorithm has to pay 1 for connecting such a terminal to the root, however, an optimal solution must connect this terminal as well using at least one arc—this argument is formalized in Lemma 16 (cf. Section 2.2).

Reduction Rule R1. *If there is an arc from the root r to a terminal $v \in R$, we contract the arc (r, v), and declare the resulting vertex the new root.*

Lemma 16. *Reduction Rule R1 is 1-safe and can be implemented in polynomial time. Furthermore, there is a solution lifting algorithm running in polynomial time and returning a Steiner arborescence if it gets a Steiner arborescence of the reduced graph as input.*

Proof. The implementation of the reduction rule is straightforward. Let H be a graph resulting from G after the contraction of the arc (r, v) to the new root r', let T_H^* and T_G^* denote optimal Steiner arborescences for H and G, respectively, and let T_H be a Steiner arborescence in H.

Our solution lifting algorithm constructs a Steiner arborescence T_G in G by simply taking T_H and uncontracting (r, v) in it. Note that T_G spans all terminals, as T_H does in H and we added (r, v). Also T_G is an arborescence, since r has in-degree zero (as r' has), v has in-degree one, and T_G is clearly a tree. Thus T_G is a Steiner arborescence in G.

The solution lifting algorithm adds 1 to the solution value, so that $w(T_G) = w(T_H) + 1$. Note that $w(T_G^*) \geq w(T_H^*) + 1$ as the optimal solution in G must additionally connect v to r, i.e., it has to add some arc of cost 1. Finally we have

$$\frac{w(T_G)}{w(T_G^*)} \leq \frac{w(T_H) + 1}{w(T_H^*) + 1} \leq \max\left\{\frac{w(T_H)}{w(T_H^*)}; \frac{1}{1}\right\},$$

so that if T_H is a β-approximation of T_H^*, then T_G is a $(\max\{1; \beta\})$-approximation of T_G^*. Hence the rule is 1-safe. $\quad\square$

The idea behind our next reduction rule is the following. Assume there is a Steiner vertex s in the optimum arborescence T connected to many terminals with paths not containing any other Steiner vertices. We can then afford to buy all these paths emanating from s together with a path connecting the root to s. Formally, we say that a vertex u is a *k-extended neighbor* of some vertex v, if there exists a directed path P starting in v and ending in u, such that $V(P) \setminus \{v\}$ contains at
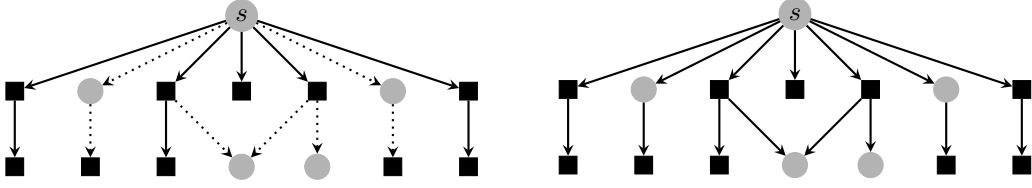
Figure 4: An example of extended neighborhood of Steiner vertex s. The set $N_{\text{Ext}}^0(s)$ is depicted on the left using full arcs, while the vertices connected by dotted arcs are not a part of this set. The set $N_{\text{Ext}}^1(s)$ is depicted on the right using full arcs.

most k Steiner vertices. Note that a vertex is always a k-extended neighbor of itself for any k, and that each of the above terminals connected to s in T is a 0-extended neighbor of s. We denote by $N_{\text{Ext}}^k(v)$ the set of all k-extended neighbors of v, and call it the k-*extended neighborhood* of v (see Fig. 4). By the following observation the Steiner vertex s of T lies in the p-extended neighborhood of the root r. Therefore there is a path containing at most p Steiner vertices connecting r to s.

Observation 17. *Let $G = (V, A)$ be a directed graph with root $r \in R$. Suppose there exists a Steiner arborescence $T \subseteq G$ with at most p Steiner vertices. It follows that $V(T) \subseteq N_{Ext}^p(r)$.*

In what follows we fix $\varepsilon > 0$. The second reduction rule contracts a path from r to a Steiner vertex s in the p-extended neighborhood of r together with the 0-extended neighborhood of s if this neighborhood is sufficiently large.

Reduction Rule R2. *If there exists a Steiner vertex s with $\left|N_{Ext}^0(s)\right| \geq p/\varepsilon$ and $s \in N_{Ext}^p(r)$ so that there is an $r \to s$ path P containing at most p Steiner vertices, then we contract the subgraph of G induced by $N_{Ext}^0(s)$ and P in G, and declare the resulting vertex the new root.*

Lemma 18. *Reduction Rule R2 is $(1 + \varepsilon)$-safe and can be implemented in polynomial time. Furthermore, there is a solution lifting algorithm running in polynomial time and returning a Steiner arborescence if it gets a Steiner arborescence of the reduced graph as input.*

Proof. Checking the applicability of Rule R2 and finding s together with $N_{\text{Ext}}^0(s)$ can be done in polynomial time as follows. We set arc lengths so that each arc ending at a terminal has length zero, while arcs ending at Steiner vertices have length one. Now a length of a directed path P from the root corresponds to the number of Steiner vertices in P. Then we run an algorithm for finding a shortest path from r to each vertex which allows us to find the set $N_{\text{Ext}}^p(r)$. Finally, for each $s \in N_{\text{Ext}}^p(r)$ we compute $N_{\text{Ext}}^0(s)$ by a simple breadth-first search.

We now specify the solution lifting algorithm. Denote by H the reduced graph obtained from G by applying R2. Let T_H be a solution of the reduced instance H and let T_H^* be an optimal solution in H. Consider the graph Q, which is the union of P and the subgraph of G induced by $N_{\text{Ext}}^0(s)$. The solution lifting algorithm first computes an arborescence A of Q rooted in r (e.g., by a depth-first search). It then takes T_H, uncontracts $N_{\text{Ext}}^0(s) \cup V(P)$, and adds the arcs in A to T_H; let T_G be the resulting graph. We show that T_G is a Steiner arborescence.

First observe that T_G spans all terminals as T_H contains all terminals in H and A is an arborescence containing all vertices in Q. Note that T_G is a tree as A is an arborescence of Q, T_H is a tree, and T_H contains at most one arc from the root in H to each vertex (recall that the root in H was created by contracting $N_{\text{Ext}}^0(s) \cup V(P)$). The root in T_G has clearly in-degree zero, while all other vertices have in-degree one, since this holds for H as T_H is an arborescence, and A is an arborescence of Q rooted in r. Thus T_G is a Steiner arborescence in G.

It remains to show the safeness of the rule. Let x be the total number of terminals in $N_{\text{Ext}}^0(s) \cup V(P)$ (not counting the root). Note that $w(T_G) \leq w(T_H) + x + p$. We obtain a solution for H of weight at most $w(T_G^*) - x$ by starting with T_G^*, removing x arcs each having one of the x non-root terminals in $N_{\text{Ext}}^0(s) \cup V(P)$ (and thus not in H) as their head, identifying all vertices in $N_{\text{Ext}}^0(s) \cup$

15

$V(P)$ with the new root, and removing loops and parallel arcs. Thus $w(T_G^*) \geq w(T_H^*) + x$ and we get

$$\frac{w(T_G)}{w(T_G^*)} \leq \frac{w(T_H) + x + p}{w(T_H^*) + x} \leq \max\left\{\frac{w(T_H)}{w(T_H^*)}; \frac{x+p}{x}\right\} \leq \max\left\{\frac{w(T_H)}{w(T_H^*)}; 1 + \varepsilon\right\}.$$

The last inequality is valid because $x \geq p/\varepsilon$. Thus if T_H is a β-approximation of T_H^*, then T_G is a $(\max\{1 + \varepsilon; \beta\})$-approximation of T_G^*, and so the reduction rule is $(1 + \varepsilon)$-safe. $\quad\square$

Now we prove that if none of the above reduction rules is applicable and our algorithm was provided with a correct value for parameter p, then the number of terminals in the reduced graph can be bounded by p^2/ε.

Lemma 19. *Let G be an instance of* Directed Steiner Tree, *and denote by H the graph obtained from G by exhaustive application of Reduction Rules R1 and R2. Suppose that there exists an optimal Steiner arborescence in G containing at most p Steiner vertices. It follows that the remaining terminal set R of H has size less than p^2/ε.*

Proof. Observe first that both our reduction rules use contractions in the underlying graph and thus if there was a solution T_G^* in G with at most p Steiner vertices, then there is a solution T_H^* in H again containing at most p Steiner vertices.

Since Reduction Rule R1 is not applicable to H, we conclude that $N_{\mathrm{Ext}}^0(r) \cap R = \emptyset$. As Reduction Rule R2 is not applicable to H it holds that $\left|N_{\mathrm{Ext}}^0(s) \cap R\right| < p/\varepsilon$ for every Steiner vertex $s \in N_{\mathrm{Ext}}^p(r)$. Therefore $|R| < p^2/\varepsilon$, since any terminal in H must be in the 0-extended neighborhood of some Steiner vertex in T_H^* and there are at most p Steiner vertices in T_H^*. $\quad\square$

The last step of the algorithm (cf. proof of Theorem 2) is to compute an optimum solution in the graph H obtained from the input graph G after exhaustively applying the two above reduction rules. From the resulting arborescence in H we obtain an arborescence in G by running the solution lifting algorithms for each reduction rule applied (in the reverse order); the existence and correctness of the solution lifting algorithms for our reduction rules is provided by Lemmas 16 and 18. The algorithm is summarized in Algorithm 2.

> **input** : directed graph $G = (V, A)$, terminals $R \subseteq V$, root $r \in R$, and integer p
> **output** : Steiner arborescence $T \subseteq G$, if p is at most the nr. of terminals in the optimum
>
> **1 if** $R \setminus N_{Ext}^p(r) \neq \emptyset$ **then** /* no solution with at most p Steiner vertices   */
> **2** $\quad$ **return** "no"
> **3 while** *Reduction Rule R1 or R2 is applicable* **do**
> **4** $\quad$ **if** *there is an arc from r to $v \in R$* **then** /* Reduction Rule R1   */
> **5** $\quad\quad$ Contract the arc (r, v), and declare the resulting vertex the new root.
> **6** $\quad$ **if** *there exists $s \in V \setminus R$ with $s \in N_{Ext}^p(r)$ and $\left|N_{Ext}^0(s)\right| \geq p/\epsilon$*
> $\quad\quad$ **then** /* Reduction Rule R2   */
> **7** $\quad\quad$ Find an $r \to s$ path P with at most p Steiner vertices. Contract the subgraph
> $\quad\quad\quad$ of G induced by $N_{\mathrm{Ext}}^0(s)$ and P, and declare the resulting vertex the new root.
> **8 if** $|R| > p^2/\varepsilon$ **then** /* no solution with at most p Steiner vertices   */
> **9** $\quad$ **return** "no"
> **10** Run the FPT algorithm of [2] and let T be the returned solution.
> **11** In the reverse order of application of Reduction Rules R1 and R2:
> **12** $\quad$ Revert the contraction of the reduction rule.
> **13** $\quad$ Run the solution lifting algorithm for the reduction rule on T.
> **14** $\quad$ Store the resulting arborescence in T.
> **15 return** T
>
> **Algorithm 2:** Algorithm for solving Directed Steiner Tree. As explained earlier, all steps except Line 10 can be implemented in polynomial time.
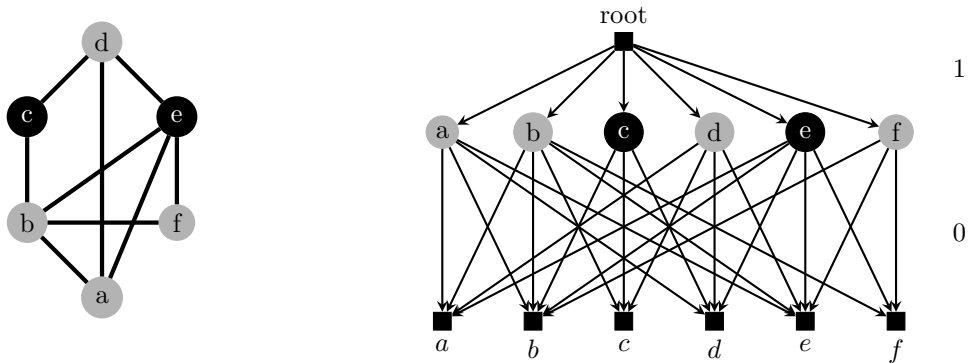
Figure 5: An example for the reduction. A graph G with its dominating set $U = \{c, e\}$ on the left. The corresponding instance of DIRECTED STEINER TREE to the right.

Proof of Theorem 2. If neither Reduction Rule R1 nor R2 is applicable and the current number of terminals exceeds the bound p^2/ε we can return "no" as it follows from Lemma 19 that no optimal solution with at most p Steiner vertices exists. If this is not the case we return an optimal solution using the algorithm of [2], which runs in time $2^{|R|} \cdot n^{O(1)}$ where R is the current set of terminals with size at most p^2/ε. As explained earlier both reduction rules can be implemented in polynomial time, together with their solution lifting algorithms. Thus the total running time is $2^{p^2/\varepsilon} \cdot n^{O(1)}$. The approximation guarantee and correctness of the obtained solution follow from Lemmas 16 and 18. $\square$

5 The weighted directed Steiner tree problem

Here we prove that the standard reduction from the DOMINATING SET problem to the DIRECTED STEINER TREE problem (with arc weights) translates into inapproximability of the latter problem. By a recent result [9], there is no constant approximation algorithm for the DOMINATING SET problem, even when parametrizing by the size of the optimum solution, unless $\mathsf{W}[1] = \mathsf{FPT}$.

DOMINATING SET	
Input:	an undirected graph $G = (V, E)$.
Solution:	the smallest *dominating set* $U \subseteq V$ for which every $v \in V$ either is in U or v has a neighbour in U.

Proof of Theorem 3. We give a parameterized reduction from the DOMINATING SET problem parameterized by the size of the solution U, which we denote by $b = |U|$.

For an overview of the reduction please refer to Fig. 5. Let $G = (V, E)$ be a graph in which we are searching for the smallest dominating set of size b and let $n = |V|$ and $m = |E|$. We create an instance of DIRECTED STEINER TREE having $2n + 1$ vertices and $n + 2m$ arcs as follows. There are n terminals, each corresponding to a vertex in V, one auxiliary terminal (the root), and n Steiner vertices again corresponding to vertices in V. There are arcs of two kinds. The first kind of arcs are of weight 1 and connect the root to each Steiner vertex, i.e., they are directed towards the Steiner vertices. The second kind of arcs are of weight 0 and connect the Steiner vertices with the terminals, directed towards the terminals. There is an arc from each Steiner vertex corresponding to a vertex $w \in V$ to every terminal corresponding to a vertex $v \in V$ if $v = w$ or v is a neighbour of w in G.

Observe that there is a dominating set of size b in G if and only if there is an arborescence connecting the root to all terminals of cost b. Note also that this arborescence contains b Steiner vertices. Thus we set the parameter p to value b.

Suppose that there is a parameterized α-approximation algorithm for the DIRECTED STEINER TREE problem for parameter p, where α is some constant. Then, we would obtain a parameterized

α-approximation algorithm for the DOMINATING SET problem parameterized by the size b of the solution. This would imply $\mathsf{W}[1] = \mathsf{FPT}$ by [9]. □

Acknowledgements. We would like to thank Michael Lampis and Édouard Bonnet for helpful discussions on the problem.

References

[1] Ajit Agrawal, Philip N. Klein, and R. Ravi. When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. *SIAM J. Comput.*, 24(3):440–456, 1995.

[2] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *STOC*, pages 67–74, 2007.

[3] Edouard Bonnet, Bruno Escoffier, EunJung Kim, and Vangelis T. Paschos. On subexponential and FPT-time inapproximability. In *IPEC*, pages 54–65, 2013.

[4] Al Borchers and Ding-Zhu Du. The k-Steiner Ratio in Graphs. *SIAM Journal on Computing*, 26(3):857–869, 1997.

[5] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5(3), 2009.

[6] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):6, 2013.

[7] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More. In *To appear in FOCS*, 2017.

[8] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33 (1):73–91, 1999.

[9] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *FOCS*, pages 505–514, 2016.

[10] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-parameter and approximation algorithms: A new look. In *IPEC*, pages 110–122, 2013.

[11] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized approximation algorithms for directed steiner network problems. *arXiv preprint arXiv:1707.06499*, 2017.

[12] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of the Steiner tree problem on graphs. In *SWAT 2002*, pages 170–179.

[13] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[14] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization Lower Bounds Through Colors and IDs. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014.

[15] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999.

[16] Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1(3): 195–207, 1971.

[17] Eduard Eiben, Mithilesh Kumar, Amer E Mouawad, and Fahad Panolan. Lossy kernels for connected dominating set on sparse graphs. *arXiv preprint arXiv:1706.09339*, 2017.

[18] David Eisenstat, Philip Klein, and Claire Mathieu. An efficient polynomial-time approximation scheme for Steiner forest in planar graphs. In *SODA 2012*, pages 626–638.

[19] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[20] Andreas Emil Feldmann. Fixed parameter approximations for k-center problems in low highway dimension graphs. In *ICALP*, pages 588–600, 2015.

[21] Andreas Emil Feldmann and Dániel Marx. The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems. In *ICALP*, pages 27:1–27:14, 2016.

[22] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *STOC*, pages 585–594, 2003.

[23] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.

[24] Mark Jones, Daniel Lokshtanov, MS Ramanujan, Saket Saurabh, and Ondřej Suchý. Parameterized complexity of directed steiner tree on sparse graphs. In *European Symposium on Algorithms (ESA)*, pages 671–682. Springer, 2013.

[25] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum, 1972.

[26] R Krithika, Pranabendu Misra, Ashutosh Rai, and Prafullkumar Tale. Lossy kernels for graph contraction problems. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 65. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[27] Michael Lampis. Parameterized approximation schemes using graph widths. In *ICALP*, pages 775–786, 2014.

[28] Daniel Lokshtanov, Fahad Panolan, MS Ramanujan, and Saket Saurabh. Lossy Kernelization. In *STOC*, pages 224–237, 2017.

[29] Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[30] Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory of Computing Systems*, 43(2): 234–253, 2008.

[31] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network Sparsification for Steiner Problems on Planar and Bounded-Genus Graphs. In *FOCS*, pages 276–285, 2014.

[32] Sebastian Siebertz. Lossy kernels for connected distance-r domination on nowhere dense graph classes. *arXiv preprint arXiv:1707.09819*, 2017.

[33] Ondřej Suchý. Extending the kernel for planar steiner tree to the number of steiner vertices. *Algorithmica*, 79(1):189–210, 2017.

[34] Andreas Wiese. A $(1 + \epsilon)$-approximation for unsplittable flow on a path in fixed-parameter running time. In *ICALP 2017*, pages 67:1–67:13.

[35] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.