PARAMETERIZED APPROXIMATION SCHEMES FOR STEINER TREES WITH SMALL NUMBER OF STEINER VERTICES *

PAVEL DVOŘÁK†, ANDREAS E. FELDMANN‡, DUŠAN KNOP§, TOMÁŠ MASAŘÍK¶, TOMÁŠ TOUFAR†, AND PAVEL VESELÝ‖

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 parameterization. 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 considered 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 approximating within any function of the studied parameter is W[1]-hard. Nevertheless, we show that an EPAS exists for Unweighted Directed Steiner Tree, but a PSAKS does not. We also 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.

Key words. Steiner Tree, Steiner Forest, Approximation Algorithms, Parameterized Algorithms

AMS subject classifications. Mathematics of computing → Combinatorics, Mathematics of computing → Graph theory, Theory of computation → Parameterized complexity and exact algorithms

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

*Submitted to the editors November 16, 2020. The research was done while the authors were at Charles University. An extended abstract of this manuscript appeared at STACS 2018 [16].

Funding: This work was partially supported by the project SVV–2017–260452. T. Masařík was supported by project 17-09142S of GAČR. A.E. Feldmann was supported by the Czech Science Foundation GAČR (grant #17-10090Y), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004). A.E. Feldmann, T. Masařík, T. Toufar, and P. Veselý were supported by the project GAUK 1514217. D. Knop is supported by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". P. Veselý was supported by European Research Council grant ERC-2014-CoG 647557. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787.

†Computer Science Institute, Charles University, Prague (koblich@iuuk.mff.cuni.cz, toufar@iuuk.mff.cuni.cz).

‡Department of Applied Mathematics, Charles University, Prague, Czech Republic (feldmann.a.e@gmail.com).

§Faculty of Information Technology, Czech Technical University, Prague, Czech Republic (dusan.knop@gmail.com).

¶Faculty of Mathematics, Informatics and Mechanics of University of Warsaw, Poland & Department of Applied Mathematics, Charles University, Prague, Czech Republic (masarik@kam.mff.cuni.cz).

‖Department of Computer Science, University of Warwick, Coventry, UK (pavel.vesely@warwick.ac.uk).

original NP-hard problems listed by Karp [26] 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 [24]).

Two popular ways to handle the seeming intractability of NP-hard problems are to design *approximation* [33] and *parameterized* [11] 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 [10], 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 idea is to isolate the exponential runtime of an NP-hard problem to the parameter. That is, the optimum solution is computed in time $f(p) \cdot n^{\mathrm{O}(1)}$, 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. Here, the choice of the parameter is crucial, and a problem may be FPT for some parameters, but not for others. 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 [15] that the Steiner Tree problem is FPT for this parameter, as the problem can be solved in time $3^{|R|} \cdot n^{\mathrm{O}(1)}$ if $n = |V|$. A more recent algorithm by Fuchs et al. [19] obtains runtime $(2+\delta)^{|R|} \cdot n^{O_\delta(1)}$ for any constant $\delta > 0$. This can be improved to $2^{|R|} \cdot n^{\mathrm{O}(1)}$ if the input graph is unweighted via the algorithm of Nederlof [29] (using 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 [14] 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 [25, 32].

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 parameterization.[1] We show that there is an *efficient parameterized approximation scheme* (EPAS), which for any $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation in time $f(p, \varepsilon) \cdot n^{\mathrm{O}(1)}$ for a 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). In fact, as summarized in the following theorem, our algorithm computes an approximation to the cheapest tree having at most p Steiner vertices, even if better solutions with more Steiner vertices exist.

Theorem 1. *There is an algorithm for* Steiner Tree*, which given an edge-weighted undirected graph $G = (V, E)$, terminal set $R \subseteq V$, $\varepsilon > 0$, and an integer p,*

[1] This area has recently received growing interest (cf. the Parameterized Approximation Algorithms Workshop)

computes a $(1+\varepsilon)$*-approximation to the cheapest Steiner tree* $T \subseteq G$ *with* $p \geq |V(T) \setminus R|$ *in time* $2^{\mathrm{O}(p^2/\varepsilon^4)} \cdot n^{\mathrm{O}(1)}$. [2]

It is worth noting that here we treat the actual value of p as a parameter; not as a "hard constraint". That is, the solution returned by our algorithm may contain more than p Steiner vertices and only its quality (cost) is compared to the cost of the cheapest solution that contains at most p Steiner vertices. This is true for all our approximation algorithms.

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 $\mathrm{O}\left(\log^{2-\varepsilon}(n)\right)$-approximation exists unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathrm{polylog}(n)})$ [21]. 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 $((1-\varepsilon)\ln n)$-approximation algorithm is possible unless $\mathsf{P} = \mathsf{NP}$ [12, 21]. At the same time, even Unweighted Directed Steiner Tree is $\mathsf{W}[2]$-hard for our considered parameter p [25, 28], just as for the undirected case. For this reason, all previous results have focused on restricted inputs: Jones et al. [25] 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, where degeneracy is measured in the underlying undirected graph, 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, the cost of a Steiner arborescence is the number of contained 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* $r \in R$*,* $\varepsilon > 0$*, and integer* p*, computes a* $(1+\varepsilon)$*-approximation to the cheapest Steiner arborescence* $T \subseteq G$ *with* $p \geq |V(T) \setminus R|$ *in time* $2^{p^2/\varepsilon} \cdot n^{\mathrm{O}(1)}$. [2]

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 computable function* f*, it is* $\mathsf{W}[1]$*-hard to compute an* $f(p)$*-approximation of the optimum Steiner arborescence* T *for* Directed Steiner

[2] If the input to this optimization problem is malformed (e.g., if p is smaller than the number of Steiner vertices of any feasible solution) then the output of the algorithm can be arbitrary (cf. [27])

Tree *parameterized by* $p = |V(T) \setminus R|$*, if the input graph is arc-weighted.*

Another variant of Steiner Tree is the Node Weighted Steiner Tree problem, in which the Steiner vertices have weights, instead of the edges. The aim is to minimize the total weight of the Steiner vertices in the computed solution. A similar reduction as the one used to prove Theorem 3 (from Dominating Set) shows that also in this case computing any $f(p)$-approximation is W[1]-hard, even if all Steiner vertices have unit weight.

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 $\{s_1, s_1'\}, \ldots, \{s_k, s_k'\}$ of terminal pairs, i.e., the set of terminals is $R = \{s_i, s_i' \mid 1 \leq i \leq k\}$. A *Steiner forest* is a forest F in G for which each $\{s_i, s_i'\}$ 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 parameterizing 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 subsection 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. [33]). 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 mentioned reduction from Steiner Tree (cf. subsection 2.1). Due to this, it is not possible to have a parameterized 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* $\{s_1, s_1'\}, \ldots, \{s_k, s_k'\} \subseteq V$ *of terminal pairs,* $\varepsilon > 0$*, and integers* p, c*, computes a* $(1 + \varepsilon)$*-approximation to the cheapest Steiner forest* $F \subseteq G$ *with at most* c *connected components and* $p \geq |V(F) \setminus R|$ *where* $R = \{s_i, s_i' \mid 1 \leq i \leq k\}$*, in time* $(2c)^{\mathrm{O}((p+c)^2/\varepsilon^4)} \cdot n^{\mathrm{O}(1)}$*.* [2]

As mentioned for Theorem 1, our algorithm might compute an approximate solution with more than p Steiner vertices. Analogously, it may also compute a forest with more than c components, even if its cost is compared to the best one containing at most p Steiner vertices and c components only.

A topic tightly connected to parameterized algorithms is kernelization. We here use the framework of Lokshtanov et al. [27], who also give a thorough introduction to the topic (see subsection 2.2 for 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 efficiently convert an optimum solution to the kernel

into an optimum solution to the input instance.

A fundamental result of parameterized complexity says that a problem is FPT if and only if it has a kernelization algorithm [11]. 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 shifted to special cases again. By a folklore result, Steiner Tree is FPT for our parameter p if the input graph is planar (cf. [25]). 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 [32] 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* (approximate) 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 parameterized 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. subsection 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 of terminal pairs $\{s_1, s'_1\}, \ldots, \{s_k, s'_k\} \subseteq V$, and integers p, c, computes an approximate kernel of size $((p+c)/\varepsilon)^{2^{O(1/\varepsilon)}}$, if for the optimum Steiner forest $F \subseteq G$, we have $p \geq |V(F) \setminus R|$ where $R = \{s_i, s'_i \mid 1 \leq i \leq k\}$, the number of connected components of F is at most c, and $\varepsilon > 0$.* [2]

Analogous to approximation schemes, it is possible to distinguish between efficient and non-efficient kernelization schemes: A PSAKS is *size 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 a size 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 or not a size 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. In fact, we prove the following theorem.

Theorem 6. *No $(2-\varepsilon)$-approximate kernelization algorithm exists for the* Un-

WEIGHTED DIRECTED STEINER TREE *problem parameterized by the number* $p = |V(T) \setminus R|$ *of Steiner vertices in the optimum Steiner arborescence* T *for any* $\varepsilon > 0$, *unless* $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$.

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 neighbors 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 [25, 32] 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. [32]) 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 in the remaining graph. This algorithm is parameterized by the number of terminals, which now is bounded by our parameter. For the variants of STEINER TREE considered in our positive results, such FPT algorithms can easily be obtained from the ones for STEINER TREE [2, 15, 19]. 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 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 subsection 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 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. [27], 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.

To obtain our inapproximability result of Theorem 3, we use a reduction from the Dominating Set problem. It was recently shown by Srikanta et al. [31] that this problem does not admit parameterized $f(k)$-approximation algorithms for any function f, if the parameter k 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. To prove Theorem 6 we use a cross composition from the Set Cover problem, for which Dinur and Steurer [12] proved that it is NP-hard to compute a $(1-\varepsilon)\ln(n)$-approximation. We are able to preserve only a constant gap; thus, we leave open whether stronger non-constant lower bounds are obtainable, or whether Unweighted Directed Steiner Tree has a polynomial size α-approximate kernel for some constant $\alpha \geq 2$.

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 general input graphs, Zelikovsky [34] gave the first polynomial time approximation algorithm for Steiner Tree with a better ratio than 2 (which can easily be obtained by computing an MST on the terminal set). His algorithm is based on repeatedly contracting stars with three terminals each, in the metric closure of the graph, which yields a 11/6-approximation. This line of work led to the Borchers and Du [4] Theorem, which states that for every Steiner Tree instance with terminal set R and every $\varepsilon > 0$ there exists a set of sub-trees (so-called *full components*) on at most $2^{O(1/\varepsilon)}$ terminals from R each and with all leaves being terminals, such that their union forms a Steiner tree for R of cost at most $1+\varepsilon$ times the optimum. As a consequence, it is possible to compute all full components with at most $2^{O(1/\varepsilon)}$ terminals (using an FPT algorithm parameterized by the number of terminals [15, 19]), and then find a subset of the precomputed solutions, in order to approximate the optimum. This method is the basis of most modern Steiner Tree approximation algorithms, and is for instance leveraged in the currently best $(\ln(4)+\varepsilon)$-approximation algorithm of Byrka et al. [6]. The Borchers and Du [4] Theorem can also be interpreted in terms of approximate kernelization schemes, as Lokshtanov et al. [27] point out (cf. proof of Theorem 5). It is interesting to note that our algorithms are also based on finding good sub-trees. However, while computing optimum full components is NP-hard, the sub-trees we compute in each step can be found in polynomial time, regardless of how many terminals they contain.

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 [17]. If the Unweighted Steiner Tree problem is parameterized by the solution size, it is known [13] that no polynomial (exact) kernel exists, unless NP $\subseteq$ coNP/Poly. If the input is restricted to planar or bounded-genus graphs it was shown that polynomial kernels do exist for this parameterization [30]. It was later shown [32] that for planar graphs this is even true for our smaller 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 $O(|R|^{\varepsilon})$-approximation can be computed in polynomial time [7], and an $O(\log^2 n)$-approximation in quasi-polynomial time [7]. Feldmann and Marx [18] consider the Directed Steiner Network problem, which is the directed variant of Steiner Forest (i.e., a generalization of Directed Steiner Tree). They give a dichotomy result, proving that the problem parameterized by $|R|$ is FPT whenever the terminal pairs induce a graph that is a caterpillar with a constant number of additional edges, and otherwise the problem is W[1]-hard. Among the W[1]-hard cases is the Strongly Connected Steiner Subgraph problem (for which the hardness was originally established by Guo et al. [20]), in which all terminals need to be strongly connected. For this problem a 2-approximation is obtainable [8] when parameterizing by $|R|$, and a recent result shows that this is the best possible [9] under the Gap Exponential Time Hypothesis.

In the same paper, Chitnis et al. [9] also consider the Bidirected Steiner Network problem, which is the directed variant of Steiner Forest on *bidirected* input graphs, i.e., directed graphs in which for every edge uv the reverse edge vu exists as well and has the same cost. These graphs model inputs that lie between the undirected and directed settings. From Theorems 1 and 5, it is not hard to see that the Bidirected Steiner Tree problem (i.e., Directed Steiner Tree on bidirected inputs) has both an EPAS and a PSAKS for our parameter p, by reducing the problem to the undirected setting. Since the PSAKS for parameter p follows from the PSAKS for parameter $|R|$ given by Lokshtanov et al. [27], it is interesting to note that for parameter $|R|$, Chitnis et al. [9] provide both a PSAKS and a parameterized approximation scheme for the Bidirected Steiner Network problem whenever the optimum solution is planar. This is achieved by generalizing the Borchers and Du [4] Theorem to this setting. As this is a generalization of Bidirected Steiner Tree, it is natural to ask whether corresponding algorithms also exist for our parameter p in the more general setting considered in [9].

2. Preliminaries.

2.1. Reducing Steiner tree to Steiner forest. By a folklore result, we may reduce the Steiner Tree problem to Steiner Forest. For this we pick 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, $R = V$ 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. [27]. See the latter reference for a thorough introduction to the topic.

For an optimization problem, a *polynomial time preprocessing 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 $|I'| + p' \leq f(p)$ 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 preprocessing 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 preprocessing 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 preprocessing 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 [27], 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 preprocessing algorithm (which is not necessarily the case when using the non-strict definition; see [27]).

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 there is a Steiner forest with 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. [27] and obtain a polynomial size lossy kernel (Theorem 5), or we can run an exact FPT algorithm derived from Fuchs et al. [19] 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 $\{s_1, s_1'\}, \ldots, \{s_k, s_k'\}$ of pairs of terminals.
Solution:	A Steiner forest $F \subseteq G$ containing an s_i-s_i' path for every $i \in [k]$

3.1. Algorithm description. We first rescale all weights so that every edge has weight strictly greater than 1. Using a standard preprocessing procedure, we also take the metric closure of the input graph, i.e., every edge of the graph is present and its weight is equal to the shortest path distance of the endpoints in the original input graph. It is easy to see (and folklore) that solving Steiner Forest in the metric closure is equivalent to solving it for the original input graph. Moreover, every solution still exists as a subgraph in the metric closure, so that our parameters remain unchanged.

Then, in each step of our algorithm we pick a star, add it to the solution, and contract the star in the current graph. After the contraction, the edge weights may not obey the triangle inequality anymore. However, this is not needed for our algorithm. Instead, we only need that the graph is always complete, so that a star to contract can always be found. 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.

Definition 7. *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 in $V(C)$, where $V(C)$ is the set of vertices in 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, and that due to rescaling of edge weights each star has ratio strictly greater than 1. Observe also that the ratio of a star is similar to the average weight of an edge in the star. However the ratio is skewed due to the subtraction of 1 in the denominator. In particular, for two stars of the same average weight, the one with more terminals will have the smaller ratio. Thus, in this sense, picking a star with small ratio favors large stars.

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). Since we assume that our input is a complete graph, a star containing two terminals always exists (except in the trivial case when there is only one terminal). Moreover, due to the following lemma, a star with the best ratio has a simple form: It consists of the cheapest ℓ edges incident to its center vertex such that all leaves are terminals. As there are n possible center vertices and at most n incident edges to each center which can be sorted in time $\mathrm{O}(n \log n)$, the best ratio star can be found in time $\mathrm{O}\left(n^2 \log n\right)$.

Lemma 8. *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 of vq_i. ∎

We now formally describe different graphs resulting from each contraction step t, together with their terminal pairs. 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 in $V(C_t)$, removing all resulting loops, and among the resulting parallel edges we delete all but the lightest one with respect to their weights. We also adjust the terminal pairs in a straightforward way: Let v be the vertex of G_{t+1} resulting from contracting C_t. If G_t had a terminal pair $\{s, s'\}$ such that s is incident to some edge of C_t while s' is not (i.e., $s \in V(C_t)$ and $s' \notin V(C_t)$), then we introduce the terminal pair $\{v, s'\}$ for G_{t+1}. Also every terminal pair $\{s, s'\}$ of G_t for which neither s nor s' is incident to any edge of C_t is introduced as a terminal pair of G_{t+1}. Somewhat counter-intuitively, we also introduce the (trivial) terminal pair $\{v, v\}$ for G_{t+1} if there was a terminal pair in G_t for which both s and s' were incident to edges of C_t. In particular, this means that v can be a leaf of a contracted star in a subsequent step, even though the solution might not require any connection from v to some other terminal. The reason we need to keep v as a terminal is that otherwise the number of Steiner vertices of the considered solution, i.e., our parameter p, might increase. Still, our analysis below shows that contracting such a trivial terminal v in a best-ratio star will not cause any problems.

The number of terminals in any given instance with terminal pairs $\{s_1, s_1'\}, \ldots, \{s_k, s_k'\}$ is the size of the set $R = \{s_i, s_i' \mid 1 \leq i \leq k\}$. This in particular means that if a terminal appears in several pairs or is in a trivial terminal pair, it is only counted once. The algorithm stops contracting best-ratio stars when there are less than τ terminals left in R; we have $\tau = \mathrm{O}\left((p+c)^2/\varepsilon^4\right)$, but we specify the precise value of τ in the analysis below. 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. Algorithm 3.1 gives a pseudo-code of the resulting algorithm.

3.2. Analysis. For the purpose of analysis, we consider a solution in the current graph G_t that originates from a 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^* a solution in G_0 with at most p Steiner vertices and at most c components, i.e., F_0^* is a Steiner forest containing an s_i-s_i' path for any i. We remark that F_0^* may or may not be an optimum solution of G_0, and that we think of F_0^* as a subgraph of G_0, isomorphic to a forest, without isolated vertices.

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} (as defined above). F_{t+1}^* will again be a subgraph of G_{t+1} without isolated vertices. 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 8 and thus part of the solution F_t^*) and possibly also the center v of C_t if it is in F_t^*. (Note that if v is not a terminal, it may not be a part of the solution F_t^*, which does not contain isolated vertices.) This results in a vertex v' and a solution F_{t+1}' for G_{t+1}, which however may well contain some cycles or loops.

We now want to delete edges incident to v' in such a way that we are left with an

Algorithm 3.1 An algorithm for solving Steiner Forest. If we stop before line 11 we obtain the reduced instance.

input : undirected graph $G = (V, E)$, list of terminal pairs $\{s_1, s'_1\}, \ldots, \{s_k, s'_k\}$, edge weights $w(e) \in \mathbb{R}_0^+$

output : a forest $F \subseteq G$ that contains an s_i-s'_i path for any $i \in \{1, \ldots, k\}$

1 **Function** `BestStar(`v`)`
2 **if** v *is a terminal* **then** $z \leftarrow 1$;
3 **else** $z \leftarrow 0$;
4 $q_1, \ldots, q_k \leftarrow$ terminals adjacent to v sorted by the weight of edge vq_i `/* `$k \geq |R|-1$ `in the metric closure */`
5 **for** i *in* $2 - z, \ldots, k$ **do**
6 $r_i \leftarrow \sum_{j=1}^{i} w(vq_j)/(i + z - 1)$
7 **return** *edges* $\{vq_1, \ldots, vq_i\}$ *of a star with the smallest* r_i
8 **while** $|R| \geq \tau$ **do**
9 $C \leftarrow \arg\min\{w(C_v) \mid C_v \leftarrow$ `BestStar(`v`)`$, v \in V\}$ `/* a star exists in the metric closure */`
10 Contract C, then remove loops and among parallel edges, keep only the lightest. Adjust the terminal pairs accordingly.
11 Run FPT algorithm parameterized by the number of terminals and connected components

acyclic feasible solution for G_{t+1}. Let Q_t denote the set of terminals in $V(C_t)$. We repeat the following simple step to find an inclusion-wise minimal feedback edge set D_t of F'_{t+1} that is incident to v': As long as there is a cycle K in F'_{t+1} (possibly, K is a loop), remove from F'_{t+1} an edge e of K such that in solution F^*_t, edge e is incident to Q_t (thus, in particular, e is incident to v' in F'_{t+1}). We claim that K must contain an edge e that is incident to a terminal in Q_t in solution F^*_t. Indeed, observe first that K must contain v', since otherwise K appears in F^*_t, which contradicts the acyclicity of F^*_t. Recall that the only vertex of $V(C_t)$ that may be a Steiner vertex is the center v of the star C_t. If K is a loop, then the only edge e of K connects two vertices in $V(C_t)$, so e is incident to Q_t. Otherwise, K contains two edges e' and e'' incident to v' that do not connect two vertices in $V(C_t)$, because edges connecting vertices in $V(C_t)$ become loops after the contraction. Since both e' and e'' cannot be incident to v in F^*_t (otherwise, K would be a cycle in F^*_t), one of e' or e'' must be incident to Q_t, which shows the claim. It follows that the above procedure is well-defined.

Once there is no cycle in F'_{t+1}, we set $F^*_{t+1} := F'_{t+1}$, 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. Let $D_t := E(F^*_t) \setminus E(F^*_{t+1})$ be the set of edges that were deleted from the solution. (We remark that we do not optimize the total length of edges in D_t.)

To show that our algorithm only loses an ε-factor compared to the cost of the solution F^*_0, we will compare the cost of the edges C_t contracted by our algorithm to the set D_t of deleted edges of F^*_t. Note 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 Figure 2) and we have to account for them differently.

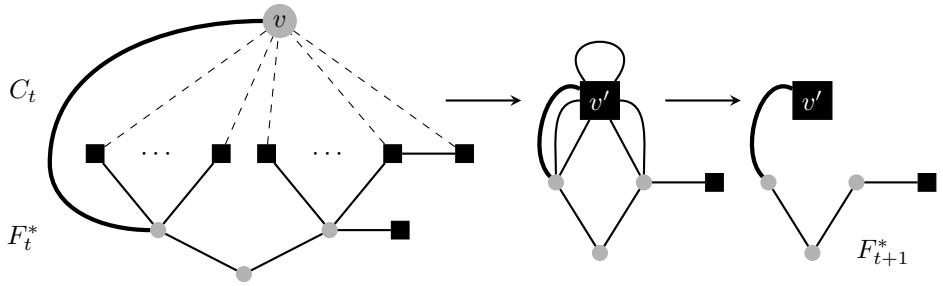

FIG. 1. *An example of creating F_{t+1}^* from F_t^* after contracting C_t. Solid edges (including the thick one) belong to solutions F_{t+1}^* and F_t^*, while edges in C_t are dashed. Note that in this example, no edge in C_t belongs to F_t^*, although this is not true in general. Set D_t consists of all edges deleted in the second step, i.e., all edges incident to v', except for the thick edge, which cannot be in D_t because it is not incident to any terminal.*

DEFINITION 9. *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 contractions is bounded by an ε-fraction of the weight of F_0^*. We start by proving that if the set Q_t of terminals in a contracted edge set C_t is sufficiently large, then the contraction is good. Intuitively, this means that skewing the ratio such that large stars are favored (compared to just picking the star with the smallest average weight) tends to result in good contractions. We define

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

LEMMA 10. *If $|Q_t| \geq \lambda$, then the contracted edge set C_t is good.*

Proof. Let $r = w(C_t)/(|Q_t|-1)$ be the ratio of the contracted star, and let ℓ' be the number of deleted edges in D_t 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_t^* (of which D_t is a subset) can only be heavier than the corresponding edge in the current graph G_t.

Let $u_1, \ldots, u_q$ be the Steiner vertices adjacent to edges in D_t, and let ℓ_i be the number of edges in D_t incident to one such Steiner vertex u_i (see Figure 3). Since D_t is a feedback edge set in which any edge was incident to a terminal in Q_t before the contraction, there is no edge in D_t which connects two Steiner vertices. Consider the star spanned by the ℓ_i edges of D_t 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_t and the algorithm chose a star with the minimum ratio in G_t. Thus, the weight of edges in D_t 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_t not incident to any Steiner vertex u_i connects two terminals. Therefore, we have $\ell' + \sum_{i=1}^{q} \ell_i = |D_t|$ as any edge in D_t is incident to a terminal in Q_t and we thus do not count any edge twice.

We observe next that from the construction of F_t^* we get that there are at least $|Q_t| - c$ edges in D_t. Recall that we contracted terminals in Q_t in the forest F_t^* which has at most c connected components in order to obtain F_{t+1}^*. Indeed, a forest on n vertices and c components has $n - c$ 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

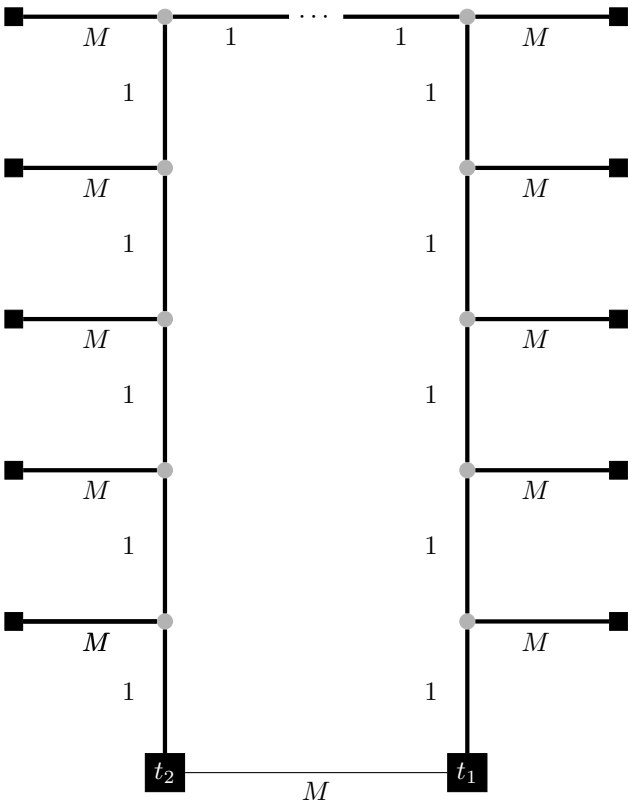

FIG. 2. *An example of a bad contraction. The optimum solution consists of the thick edges. The numbers of terminals and the weight M can be arbitrarily large. Note that any edge in the metric closure between any two terminals has length of at least M if there are at least $M+1$ terminals. The star centered at t_1 and containing the incident terminal t_2 has ratio M, while every other star in the metric closure of the graph has ratio slightly more than M. By contracting the star t_1, t_2 we create a cycle in the optimum solution containing edges of weight 1 only. Thus, for a sufficiently large value of M the contraction cannot be charged.*

a Steiner vertex present in F_t^*), and we decrease the number of components by at most $c-1$. Let z be the number of vertices in F_t^*. We conclude that the forest F_{t+1}^* has at most $z-|Q_t|+1$ vertices and at least 1 connected component. Thus, there are at most $z-|Q_t|$ edges in F_{t+1}^* and we get that $|D_t| \geq z-c-(z-|Q_t|) = |Q_t|-c$.

Since F_t^* contains at most p Steiner vertices we have $q \leq p$, and we obtain

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

Finally, using $|Q_t| \geq \lambda$ we bound $w(C_t)$ by $(1+\varepsilon)w(D_t)$ as follows:

$$\begin{aligned}(1+\varepsilon)w(D_t) &\geq (1+\varepsilon)r\big(|Q_t| - p - c\big) = r|Q_t| + r\big(\varepsilon|Q_t| - (1+\varepsilon)(p+c)\big)\\ &\geq w(C_t) + r\left(\varepsilon\frac{(1+\varepsilon)(p+c)}{\varepsilon} - (1+\varepsilon)(p+c)\right) = w(C_t)\,. \quad \square\end{aligned}$$

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

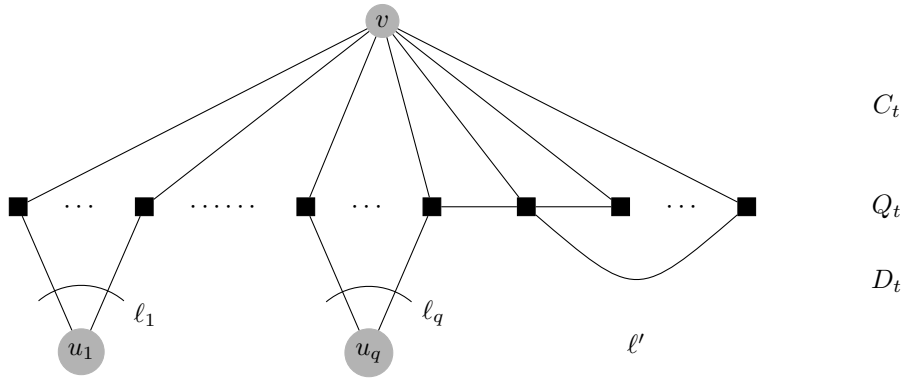

FIG. 3. *The contracted star C_t and a part of the optimal solution spanned by the terminals Q_t of the star C_t.*

contractions with ratio in an interval $\big((1+\delta)^i;(1+\delta)^{i+1}\big]$ 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 $r \in \big((1+\delta)^i;(1+\delta)^{i+1}\big]$, an *$i$-contraction*, and define

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

LEMMA 11. *For any integer i the number of bad single-component i-contractions is at most κ.*

Proof. Let us focus on bad single-component i-contractions only, which we here just call bad i-contractions for brevity. Suppose for a contradiction that the number of bad i-contractions is larger than κ. The plan is to show that at each of the κ steps t in which a bad i-contraction happens, there must be a cheap edge e_t in the corresponding set D_t. Since the deleted sets D_t are disjoint, all of these edges are also present in $G_{\tilde{t}}$ of the first step $\tilde{t}$ with a bad i-contraction, i.e., $\tilde{t}$ is the minimum among all t for which $w(C_t) > (1+\varepsilon)w(D_t)$ and $w(C_t)/\big(|Q_t|-1\big) \in \big((1+\delta)^i;(1+\delta)^{i+1}\big]$ and the contraction is single-component. We then show that among all the cheap edges in $G_{\tilde{t}}$ there is a "light" star 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 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 $w(C_t) \leq (1+\delta)^{i+1}\big(|Q_t|-1\big)$ as the ratio of C_t is in interval i. Putting it together and using the definition of δ we obtain

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

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, 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 connect 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_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 solution from which $F_{\tilde{t}}^*$ is derived contained at most p Steiner vertices. Summarizing, we assume that there are more than κ bad single-component i-contractions, each of which contributes one edge e_t that is incident to a Steiner vertex, and we remove less than p edges e_t that connect two Steiner vertices, which implies that set S of the remaining edges e_t satisfies $|S| > \kappa - p$.

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 $|C|(1+\delta)^{i-1}/\big(|C|-1\big)$. Since $|C|/\big(|C|-1\big) \leq (1+\delta)$ (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 remark that the proof of Lemma 11 does not use that the number of terminals in a bad i-contraction is bounded by λ, as shown in Lemma 10. Instead we will 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 $(1+\delta)^{j+1}$. As there are at most κ bad single-component contractions in each interval and at most $c-1$ (bad) multiple-component contractions, and as the interval size grows exponentially, we can upper bound the total weight of bad contractions in terms of κ, c, λ, and $(1+\delta)^j$. We can also lower bound the weight of $w(F_0^*)$ in terms of $(1+\delta)^j$ and the lower bound τ on the number of terminals in the graph. If τ is large enough, then the total weight of edge sets C_t of bad contractions is at most $\varepsilon \cdot w(F_0^*)$. These ideas are summarized in the next lemma.

LEMMA 12. *Let j be the largest interval of any contraction during the whole run of the algorithm and let W_B be the total weight of the union edge sets C_t of bad contractions. Then, the following holds.*
(1) $W_B \leq (\kappa + c) \cdot \lambda \cdot (1+\delta)^{j+2}/\delta$.
(2) $w(F_0^*) \geq (1+\delta)^j \cdot (\tau - c - p)$.
(3) Let

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

then $W_B \leq \varepsilon \cdot w(F_0^*)$.

Proof. To prove (1), observe first that if C_t is a 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, there are less than c bad multiple-component contractions. Each of them has at most λ terminals by Lemma 10 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 $(1+\delta)^{j+1} \cdot c \cdot \lambda$.

Note that it follows from Lemmas 10 and 11 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 possibly does a contraction:

$$\begin{aligned}
&\kappa \cdot \lambda \cdot \sum_{1 \le i \le j} (1+\delta)^{i+1} + c \cdot \lambda \cdot (1+\delta)^{j+1} \\
&= \kappa \cdot \lambda \cdot \frac{(1+\delta)^{j+2} - (1+\delta)}{(1+\delta) - 1} + c \cdot \lambda \cdot (1+\delta)^{j+1} \\
&\le (\kappa + c) \cdot \lambda \cdot \frac{(1+\delta)^{j+2}}{\delta} .
\end{aligned}$$

This proves (1).

For (2), when our algorithm contracted a star having ratio $r \ge (1+\delta)^j$ in the largest interval j in some step t, all stars in G_t with at least two terminals had ratio at least r. Let $v_1, \ldots, v_q$ be the Steiner vertices of F_t^* and $u_1, \ldots, u_{q'}$ be Steiner vertices of F_t^* which are connected to at least one terminal. Thus, if ℓ_i is the number of terminals adjacent to u_i in F_t^*, then these terminals together with u_i form a star of weight at least $r \cdot (\ell_i - 1)$ if $\ell_i \ge 2$, since no edge in F_t^* is lighter than the corresponding edge of G_t. If $\ell_i = 1$ then lower bound $r \cdot (\ell_i - 1) = 0$ on the weight trivially holds. Similarly, all edges between terminals in F_t^* have weight at least r; let ℓ' be the number of such edges.

Since F_t^* has at least τ terminals in step t (otherwise the algorithm would have terminated), it contains q Steiner vertices, and has at most c components, the total number of edges of F_t^* is $\tau + q - c$. Those of its edges that connect two Steiner vertices form a forest on at most q vertices, and there can therefore be at most $q - 1$ such edges. Hence the number of edges in F_t^* that are incident to a terminal is $\ell' + \sum_{i=1}^{q'} \ell_i \ge \tau + q - c - (q-1) \ge \tau - c$. Using $p \ge q'$, the total weight of edges in F_t^* is at least

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

This shows (2) as $w(F_t^*) \le w(F_0^*)$.

To get (3), by (2) and using the value of τ we have

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

which is the upper bound on W_B by (1). ∎

The above lemma can now be used to prove that all the contractions put together (with ε scaled appropriately) form a $(1+\varepsilon)$-approximate preprocessing procedure with respect to F_0^* (cf. subsection 2.2).

LEMMA 13. *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 preprocessing algorithm with respect to F_0^*.*

Proof. 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 $C_0, C_1, \ldots$ to F_H in order to compute a Steiner forest F_G in the input graph G_0. We need to show that, if F_H is a β-approximation to the optimum solution F_H^* in H, the resulting forest F_G is a $\big((1+2\varepsilon)\beta\big)$-approximation to the solution F_0^* of G_0.

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, we use Lemma 12 to bound the weight of F_G by

$$w(F_G) = \sum_{\text{good } t} w(C_t) + \sum_{\text{bad } t} w(C_t) + w(F_H) \leq \sum_{\text{good } t} (1+\varepsilon)w(D_t) + \varepsilon \cdot w(F_0^*) + \beta \cdot w(F_H^*).$$

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 $\sum_{\text{good } t} w(D_t) \leq \sum_t w(D_t) \leq w(F_0^*) - w(F_{\tilde{t}}^*)$. Therefore, the above bound becomes

$$\begin{aligned} w(F_G) &\leq (1+\varepsilon)\left(w(F_0^*) - w(F_{\tilde{t}}^*)\right) + \varepsilon \cdot w(F_0^*) + \beta \cdot w(F_{\tilde{t}}^*) \\ &\leq (1+\varepsilon)\beta\left(w(F_0^*) - w(F_{\tilde{t}}^*) + w(F_{\tilde{t}}^*)\right) + \varepsilon \cdot w(F_0^*) \leq (1+2\varepsilon)\beta \cdot w(F_0^*)\,, \end{aligned}$$

which proves the claim. ∎

Note that in case the given p is smaller than the number of Steiner vertices in F_0^*, or c is smaller than the number of connected components in F_0^*, the algorithm still outputs a Steiner forest, but the approximation factor may be arbitrary. Finally, we provide proofs of Theorems 4 and 5.

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 [19] 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 [19] on each subset to compute an optimum Steiner tree connecting it. To bound the number of partitions of the terminal set, recall that the input to our algorithm has an integer c upper-bounding the number of components in a solution with which we compare our solution. Thus, each terminal can be in one of at most c components, so there are at most $c^{|R|}$ partitions of the terminal set R that need to be considered. The algorithm of [19] runs in time $(2+\delta)^{|R|} \cdot n^{\mathrm{O}(1)}$ for any constant $\delta > 0$, and this results in an algorithm with runtime $((2+\delta)c)^{|R|} \cdot n^{\mathrm{O}(1)}$ to solve Steiner Forest. We run this algorithm on the Steiner Forest instance that our preprocessing algorithm of Lemma 13 computes, in order to obtain Theorem 4. ∎

To obtain Theorem 5 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 [15, 19], this problem does not admit polynomial (exact) kernels [13], unless NP ⊆ coNP/Poly. However, as shown by Lokshtanov et al. [27], the Borchers and Du [4] Theorem can be reinterpreted to show that a PSAKS exists. Obtaining a PSAKS for Steiner Forest can be done in essentially the same way as described in [27], and together with Lemma 13 this gives a PSAKS for our choice of parameters.

Proof of Theorem 5. The Borchers and Du [4] Theorem states that for any optimum Steiner tree T on terminal set R there exists a collection of trees $T_1, \ldots, T_k$, such

that all leaves of each tree belong to R, each T_i contains $2^{O(1/\varepsilon)}$ terminals of R, and the union $\bigcup_{i=1}^{k} T_i$ is a $(1+\varepsilon)$-approximation of T. This theorem can also be applied to each tree in the optimum Steiner Forest solution, since each such tree must be an optimum Steiner tree for its contained terminal set.

In particular, to compute a kernel, first we take the metric closure of the graph with τ terminals computed by our algorithm, so that any minimum cost tree connecting $2^{O(1/\varepsilon)}$ terminals can be assumed to only contain $2^{O(1/\varepsilon)}$ Steiner vertices as well. We then compute an optimum Steiner tree for each subset of R of size at most $2^{O(1/\varepsilon)}$. This is done using an FPT algorithm parameterized by the number of terminals, which takes polynomial time if ε is a constant. Within the union of all computed Steiner trees exists a $(1+\varepsilon)$-approximate Steiner forest due to the Borchers and Du [4] Theorem, and the total number of vertices in this union is $|R|^{2^{O(1/\varepsilon)}}$. However, the union is not of polynomial size in $|R|$ yet, due to the edge lengths. Lokshtanov et al. [27] show that it is possible to round the edge lengths in such a way that the cost of every Steiner tree grows by at most a factor of $(1+\varepsilon)$, and the edge lengths can be encoded using at most $O(\log(|R|) + \log(1/\varepsilon))$ bits. For this an estimate on the cost of the optimum solution is needed, which can be obtained using the polynomial time 2-approximation algorithm for Steiner Forest by Agrawal et al. [1].

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 by Lemma 13. Hence, the union of all computed solutions for terminal sets of size at most $2^{O(1/\varepsilon)}$ with rounded edge lengths is a polynomial-sized $(1+\varepsilon)$-approximate kernel for Steiner Forest. ∎

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.

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 $v \in R$.

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 H, and prove that the number of terminals in H is bounded by a function of our parameter p and the approximation ratio $(1+\varepsilon)$. On H we use the algorithm of Nederlof [29] 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, any feasible solution must connect this terminal as well using at least one arc—this argument is formalized in Lemma 14 (cf. subsection 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 14. *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.*

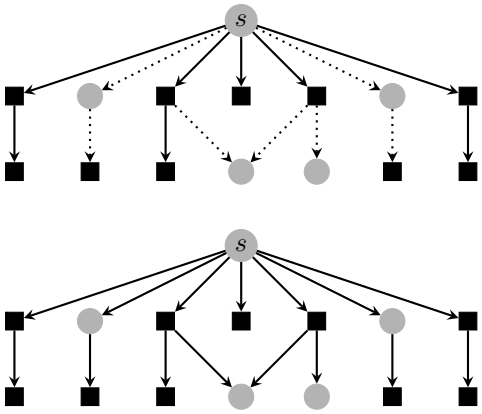

FIG. 4. *An example of extended neighborhood of Steiner vertex s. The set $N^0_{Ext}(s)$ is depicted on the top using full arcs, while the vertices connected by dotted arcs are not a part of this set. The set $N^1_{Ext}(s)$ is depicted on the bottom using full arcs.*

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. ∎

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 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 $N^k_{\mathrm{Ext}}(v)$ the set of all k-extended neighbors of v, and call it the *k-extended neighborhood* of v (see Figure 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 15. *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 $V(T) \subseteq N^p_{Ext}(r)$.*

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 $\left|N^0_{Ext}(s)\right| \geq p/\varepsilon$ and $s \in N^p_{Ext}(r)$, 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 $N^0_{Ext}(s)$ and P in G, and declare the resulting vertex the new root.*

LEMMA 16. *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 $N^0_{\mathrm{Ext}}(s)$ 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 $N^p_{\mathrm{Ext}}(r)$. Finally, for each $s \in N^p_{\mathrm{Ext}}(r)$ we compute $N^0_{\mathrm{Ext}}(s)$ 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 $N^0_{\mathrm{Ext}}(s)$. The solution lifting algorithm first computes an arborescence A of Q rooted in r (e.g., by a depth-first search). Define T_G as the union of T_H and A. 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 $N^0_{\mathrm{Ext}}(s) \cup V(P)$). 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 $N^0_{\mathrm{Ext}}(s) \cup V(P)$ (not counting the root) and let T^*_G be an optimal solution in G. 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 $N^0_{\mathrm{Ext}}(s) \cup V(P)$ (and thus not in H) as their head, identifying all vertices in $N^0_{\mathrm{Ext}}(s) \cup V(P)$ 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. ∎

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 17. *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 a 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 $N^0_{\mathrm{Ext}}(r) \cap R = \emptyset$. As Reduction Rule R2 is not applicable to H, it holds that $\left|N^0_{\mathrm{Ext}}(s) \cap R\right| < p/\varepsilon$ for every Steiner vertex $s \in N^p_{\mathrm{Ext}}(r)$. 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. ∎

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 Reduction Rules R1 and R2. 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 14 and 16. The algorithm is summarized in Algorithm 4.1.

Algorithm 4.1 Algorithm for solving Directed Steiner Tree. As explained earlier, all steps except line 10 can be implemented in polynomial time.

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** $R \setminus N^p_{Ext}(r) \neq \emptyset$ **then** `/* no solution with at most` p `Steiner vertices */`
2 **return** "no"
3 **while** *Reduction Rule R1 or R2 is applicable* **do**
4 **if** *there is an arc from r to $v \in R$* **then** `/* Reduction Rule R1 */`
5 Contract the arc (r, v), and declare the resulting vertex the new root.
6 **if** *there exists $s \in V \setminus R$ with $s \in N^p_{Ext}(r)$ and $\left|N^0_{Ext}(s)\right| \geq p/\epsilon$* **then** `/* Reduction Rule R2 */`
7 Find an $r \to s$ path P with at most p Steiner vertices. Contract the subgraph of G induced by $N^0_{\mathrm{Ext}}(s)$ 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 **return** "no"
10 Run the FPT algorithm of [29]; let T be the returned solution.
11 In the reverse order of application of Reduction Rules R1 and R2:
12 Revert the contraction of the reduction rule.
13 Run the solution lifting algorithm for the reduction rule on T.
14 Store the resulting arborescence in T.
15 **return** T

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 17 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 [29], which runs in time $2^{|R|} \cdot n^{\mathrm{O}(1)}$ where R is the current set of terminals with size at most

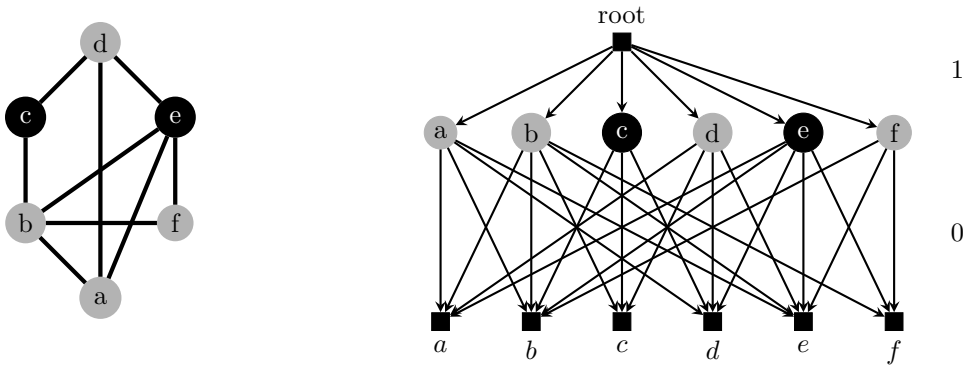

FIG. 5. *An example for the reduction. A graph G with its dominating set $U = \{c, e\}$ on the left. The corresponding instance of* DIRECTED STEINER TREE *to the right.*

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 $2^{p^2/\varepsilon} \cdot n^{\mathrm{O}(1)}$. The approximation guarantee and correctness of the obtained solution follow from Lemmas 14 and 16. □

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 of Srikanta et al. [31], there is no $f(b)$-approximation algorithm for the DOMINATING SET problem, even when parameterizing by the size b 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 neighbor 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 Figure 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 neighbor 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 $f(p)$-approximation algorithm for the DIRECTED STEINER TREE problem for parameter p and a computable function

f. Then, we would obtain a parameterized $f(b)$-approximation algorithm for the Dominating Set problem parameterized by the size b of the solution. This would imply $\mathsf{W}[1] = \mathsf{FPT}$ by [31]. ∎

6. Refuting a PSAKS for Unweighted Directed Steiner Tree. In this section, we prove that the Unweighted Directed Steiner Tree problem does not admit a $(2-\varepsilon)$-approximate polynomial kernel for any constant $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$. We use a framework for proving lower bounds on approximate polynomial kernels by Lokshtanov et al. [27] and present an α-gap cross composition (for $\alpha = 2-\varepsilon$). For the composition, we need to define a polynomial equivalence.

Definition 18. *An equivalence relation $\equiv$ on Σ^*, where Σ is a finite alphabet, is called a* polynomial equivalence relation *if*

1. *The equivalence of any $x, y \in \Sigma^*$ can be checked in time polynomial in $|x|+|y|$.*
2. *Any finite set $S \subseteq \Sigma^*$ has at most $(\max_{x\in S} |x|)^{O(1)}$ equivalence classes.*

Now we explain how the composition works. Let $L \subseteq \Sigma^*$ be a language, and let $x_1, \ldots, x_t \in \Sigma^*$ be strings belonging to the same class of some polynomial equivalence $\equiv$. The composition, given $x_1, \ldots x_t$, runs in time polynomial in $\sum_{i=1}^{t} |x_i|$ and computes $c \in \mathbb{R}$ and an instance (G, R, p) of the Unweighted Directed Steiner Tree problem parameterized by p such that:

1. If $x_i \in L$ for some $1 \leq i \leq t$, then G contains a Steiner arborescence containing at most c arcs.
2. If $x_i \notin L$ for all $1 \leq i \leq t$, then any Steiner arborescene of G contains at least $\alpha \cdot c$ arcs.
3. The parameter p is bounded by a polynomial in $\log t + \max_{1\leq i\leq t} |x_i|$.

By the framework of Lokshtanov et al. [27], if L is an NP-hard language, then the Unweighted Directed Steiner Tree problem does not admit a polynomial-sized α-approximate kernel for parameter p, unless $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$. We use the Set Cover problem as the language L.

Set Cover	
Input:	A universe U, a set $\mathcal{P}$ of subsets of U, and a positive integer b.
Solution:	A set $\mathcal{C} \subseteq \mathcal{P}$ such that $\lvert\mathcal{C}\rvert \leq b$ and $U = \bigcup_{C\in\mathcal{C}} C$.

We call b the *budget*. Let $\mathcal{I}_1, \ldots, \mathcal{I}_t$ be instances of the Set Cover problem. We define the polynomial equivalence $\equiv$ as follows. Two Set Cover instances $(U_1, \mathcal{P}_1, b_1)$ and $(U_2, \mathcal{P}_2, b_2)$ are equivalent in $\equiv$ if $|U_1| = |U_2| = n$, $|\mathcal{P}_1| = |\mathcal{P}_2| = m$ and $b_1 = b_2 = b$. Thus, we can suppose that all instances $\mathcal{I}_1, \ldots, \mathcal{I}_t$ are over the same universe U. It is straightforward to verify that the relation $\equiv$ is a polynomial equivalence relation.

We can also suppose that m is polynomial in n and either each instance $\mathcal{I}_i$ has a set cover of size at most b or each set cover has size at least γb for arbitrary constant γ (actually γ can be $O(\log n)$ but we do not need this here). By a result of Dinur and Steurer [12], the Set Cover problem is still NP-hard in this case.

The first step of the α-gap cross composition is to convert each instance $\mathcal{I}_k$ to an instance G_k of the Unweighted Directed Steiner Tree problem. The construction is similar to the reduction in the proof of Theorem 3. Let $\mathcal{I}_k = (U, \mathcal{P}_k, b)$ be an instance of Set Cover. We create a terminal root r_k and for each $S_j^k \in \mathcal{P}_k$ we create a Steiner vertex s_j^k. We add a directed path of length n from r_k to each s_j^k. Then we create a terminal t_i^k for every $i \in U$ and create the incidence graph of U and $\mathcal{P}$, i.e., we add an arc (s_j^k, t_i^k) if $i \in S_j^k$ where S_j^k is the set in $\mathcal{P}$ corresponding to s_j^k.

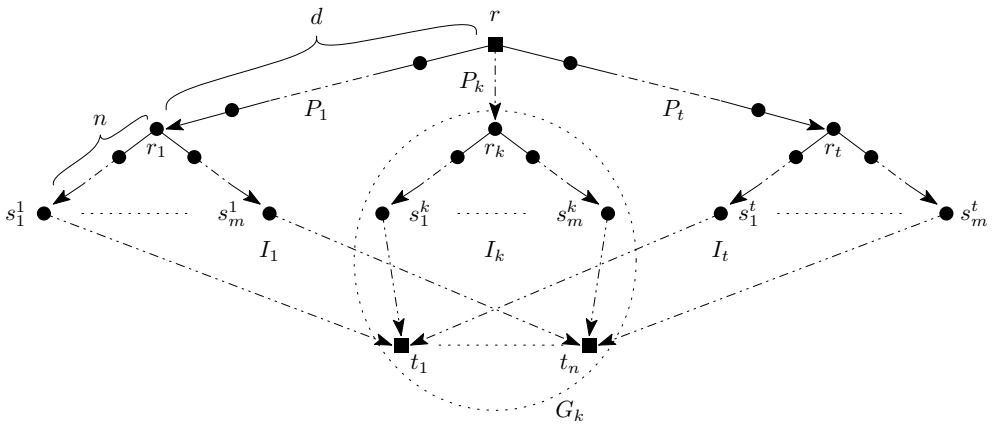

FIG. 6. *Sketch of the $(2-\varepsilon)$-gap cross composition. All arcs are oriented in "top-down" direction from the root r to terminals t_i. The graph I_k is an incident graph of the instance $\mathcal{I}_k = (U, \mathcal{P}_k, b)$ of the* SET COVER *problem. The graph G_k is the graph I_k with paths from the vertex r_k to the vertices $s_1^k, \ldots, s_m^k$.*

If $\mathcal{I}_k$ is a *yes*-instance (it has a set cover of size at most b), then G_k has a Steiner arborescence with $bn + n = (b+1)n$ arcs. On the other hand, if $\mathcal{I}_k$ is a *no*-instance (each set cover has size at least γb), then each Steiner arborescence of G_k has more than $\gamma bn + n$ arcs.

Now we combine all G_k into one instance G of the UNWEIGHTED DIRECTED STEINER TREE problem. First, we create a root r of the instance G. We connect r and each vertex r_k (the root of the instance G_k) by a directed path P_k of length d (the value of d will be determined later). Thus, the root r has degree t. Finally, we identify all terminals t_i^k of all graphs G_k corresponding to the same element i in U into one terminal t_i, i.e., the graph G has n terminals apart from the root. See Figure 6 for a sketch of the composition.

LEMMA 19. *If for some k, $\mathcal{I}_k$ is a yes-instance then G has a Steiner arborescence with at most $d + (b+1)n$ arcs.*

Proof. Let $\mathcal{C}$ be a set cover of $\mathcal{I}_k$ of size at most b. The arborescence T contains the path P_k from r to r_k, thus it contains d arcs. Let S be Steiner vertices in G_k corresponding to the sets in $\mathcal{C}$. We add to T all the paths from r_k to Steiner vertices in S; as $|S| \leq |\mathcal{C}| \leq b$, these paths have at most bn arcs. Since $\mathcal{C}$ is a set cover, there are n arcs from Steiner vertices in S to all terminals of G and we add them to T. Thus, T connects the root r of G to all the terminals of G and it has $d + (b+1)n$ arcs. ∎

LEMMA 20. *Let T be a Steiner arborescence of G. Suppose T contains two distinct paths P_i and P_j. Then, T has at least $2d + 3n$ arcs.*

Proof. The paths P_i and P_j are edge disjoint and each contains d arcs. Further, T contains at least n arcs from r_i to some Steiner vertex in G_i and at least n arcs from r_j to some Steiner vertex in G_j. Finally, we still need n arcs to connect the terminals and we get

$$|E(T)| \geq 2d + 3n.$$

∎

LEMMA 21. *Let T be a Steiner arborescence of G such that T contains only one path P_k. If all instances $\mathcal{I}_1, \ldots, \mathcal{I}_t$ are no-instances, then any Steiner arborescence of*

G has at least $d + n(\gamma b + 1)$ arcs.

Proof. Let T' be an arborescence we get from T when we remove the path P_k. Since $V(P_k) \cap V(T') = \{r_k\}$, the arborescence T' is a Steiner arborescence of G_k. Thus, the arborescence T' has at least $n(\gamma b + 1)$ arcs, because the instance $\mathcal{I}_k$ is a *no*-instance. Adding the d edges of P_k, we obtain the claimed bound. ∎

Now we calculate the value of d. We set d large enough so that Steiner arborescences which contain more than one path P_k are bigger than Steiner arborescences which contain only one such path. Formally, by the above two lemmas we want

$$2d + 3n \geq d + n(\gamma b + 1).$$

Thus, we set $d = n(\gamma b - 2)$ and we get the following corollary of Lemma 20 and Lemma 21.

COROLLARY 22. *If all instances $\mathcal{I}_1, \ldots, \mathcal{I}_t$ are no-instances, then each Steiner arborescence of G has at least $n(2\gamma b - 1)$ arcs.*

OBSERVATION 23. *The graph G has a Steiner arborescence T with at most $d+n^2+n$ Steiner vertices.*

Proof. We take a path P_k from r into an arbitrary vertex r_k (with d Steiner vertices) and an arbitrary Steiner arborescence in G_k (with at most $n^2 + n$ Steiner vertices—from a trivial set cover when each element is covered by its own set). ∎

Thus, our parameter p of G (the number of Steiner vertices in the optimum) is bounded by a polynomial in n, as $d = n(\gamma b - 2)$ and $b \leq n$. If there is a *yes*-instance among $\mathcal{I}_1, \ldots, \mathcal{I}_t$, then by Lemma 19 we know that the optimal Steiner arborescence of G has at most $d + (b+1)n = n\big((\gamma+1)b - 1\big)$ arcs. If there are *no*-instances among $\mathcal{I}_1, \ldots, \mathcal{I}_t$ only, then by Corollary 22 the optimal Steiner arborescence of G has at least $n(2\gamma b - 1)$ arcs. This means that

$$\frac{n(2\gamma b - 1)}{n\big((\gamma+1)b - 1\big)} \geq 2 - \varepsilon$$

for γ large enough. Thus, for any constant $\varepsilon > 0$ we created a $(2-\varepsilon)$-gap cross composition from the SET COVER problem to the UNWEIGHTED DIRECTED STEINER TREE problem parameterized by the number of Steiner vertices in the optimum. This refutes the existence of polynomial-sized $(2-\varepsilon)$-approximate kernels for this problem, unless $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$, and proves Theorem 6.

7. Conclusions and open problems. Recently, it was shown that contracting stars not only leads to parameterized approximation schemes for the STEINER TREE problem, as outlined in this paper, but also behaves well in practical computations [22, 23]. In fact, this idea was used as a heuristic, which significantly improves approximations of minimum spanning trees. The implementation of this idea together with only a few additional heuristics was awarded the 4th place (out of 24) in the PACE challenge 2018 in the very competitive Track C [3].

From our theoretical work, we leave the following open problems:

- The runtimes of our approximation schemes may be improvable. In particular, we conjecture that a linear dependence on our parameter p should suffice in the exponent of both algorithms in Theorems 1 and 2. It would also be very interesting to obtain runtime lower bounds for our approximation schemes under some reasonable complexity assumption.

- Given that we obtain a PSAKS for the Steiner Tree problem, but not for the Unweighted Directed Steiner Tree problem (even though we show an EPAS for each of them), one remaining open question is what the best approximation ratio obtainable by a polynomial-sized kernel is for the latter. Namely, is there a polynomial-sized α-approximate kernel for Unweighted Directed Steiner Tree for some constant $\alpha \geq 2$?
- As mentioned in subsection 1.2, a parameterized approximation scheme and a PSAKS exist for the Bidirected Steiner Network problem with planar optimum [9] for parameter $|R|$. The PSAKS uses a generalization of the PSAKS for Steiner Tree with parameter $|R|$ by Lokshtanov et al. [27]. Hence, it is natural to ask whether or not this is also the case for our parameter p, i.e., whether or not there is a parameterized approximation scheme and/or a PSAKS for Bidirected Steiner Network with planar optimum when parameterized by p.

Acknowledgments. We would like to thank Michael Lampis and Édouard Bonnet for helpful discussions on the problem. Also, we thank Jiří Sgall and Martin Böhm for finding a mistake in a preliminary version of the proof of Lemma 11.

References.

[1] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, jun 1995, doi:10.1137/s0097539792236237.

[2] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC 07*. ACM Press, 2007, doi:10.1145/1250790.1250801.

[3] Édouard Bonnet and Florian Sikora. The PACE 2018 Parameterized Algorithms and Computational Experiments Challenge: The Third Iteration. In Christophe Paul and Michal Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, volume 115 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-084-2, doi:10.4230/LIPIcs.IPEC.2018.26.

[4] Al Borchers and Ding-Zhu Du. The k-Steiner Ratio in Graphs. *SIAM Journal on Computing*, 26(3):857–869, 1997, doi:10.1137/S0097539795281086.

[5] Glencora Borradaile, Philip Klein, and Claire Mathieu. AnO(nlogn) approximation scheme for steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5(3): 1–31, jul 2009, doi:10.1145/1541885.1541892.

[6] Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoss, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60 (1):1–33, feb 2013, doi:10.1145/2432622.2432628.

[7] Moses Charikar, Chandra Chekuri, To yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, oct 1999, doi:10.1006/jagm.1999.1042.

[8] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-parameter and approximation algorithms: A new look. In *Parameterized and Exact Computation*, pages 110–122. Springer International Publishing, 2013, doi:10.1007/978-3-319-03898-8_11.

[9] Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized Approximation Algorithms for Bidirected Steiner Network Problems. In *26th*

Annual European Symposium on Algorithms (ESA 2018), volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-081-1, doi:10.4230/LIPIcs.ESA.2018.20.

[10] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of the steiner tree problem on graphs. In *Algorithm Theory — SWAT 2002*, pages 170–179. Springer Berlin Heidelberg, 2002, doi:10.1007/3-540-45471-3_18.

[11] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015, doi:10.1007/978-3-319-21275-3.

[12] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing - STOC 14*. ACM Press, 2014, doi:10.1145/2591796.2591884.

[13] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM Transactions on Algorithms*, 11(2):1–20, oct 2014, doi:10.1145/2650261.

[14] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer New York, 1999, doi:10.1007/978-1-4612-0515-9.

[15] Stuart E. Dreyfus and Robert A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971, doi:10.1002/net.3230010302.

[16] Pavel Dvořák, Andreas Emil Feldmann, Dušan Knop, Tomáš Masařík, Tomáš Toufar, and Pavel Veselý. Parameterized approximation schemes for steiner trees with small number of steiner vertices. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 26:1–26:15, 2018, doi:10.4230/LIPIcs.STACS.2018.26.

[17] David Eisenstat, Philip Klein, and Claire Mathieu. An efficient polynomial-time approximation scheme for steiner forest in planar graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 626–638. Society for Industrial and Applied Mathematics, jan 2012, doi:10.1137/1.9781611973099.53.

[18] Andreas Emil Feldmann and Dániel Marx. The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55, pages 27:1–27:14, 2016, doi:10.4230/LIPIcs.ICALP.2016.27.

[19] Bernhard Fuchs, Walter Kern, D Molle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007, doi:10.1007/s00224-007-1324-4.

[20] Jiong Guo, Rolf Niedermeier, and Ondrej Suchý. Parameterized complexity of arc-weighted directed Steiner problems. *SIAM J. Discrete Math.*, 25(2):583–599, 2011, doi:10.1137/100794560.

[21] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing - STOC 03*, pages 585–594. ACM Press, 2003, doi:10.1145/780542.780628.

[22] Radek Hušek, Tomáš Toufar, Dušan Knop, Tomáš Masařík, and Eduard Eiben. Steiner tree heuristics for PACE 2018 challenge track C, 2018. https://github.com/goderik01/PACE2018.

[23] Radek Hušek, Dušan Knop, and Tomáš Masařík. Approximation algorithms for steiner tree based on star contractions: A unified view. In *15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*, pages 16:1–16:18, 2020, doi:10.4230/LIPIcs.IPEC.2020.16.

[24] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992, doi:10.1016/s0167-5060(08)x7008-6.

[25] Mark Jones, Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Ondřej Suchý. Parameterized complexity of directed steiner tree on sparse graphs. *SIAM J. Discret. Math.*, 31(2):1294–1327, 2017, doi:10.1137/15M103618X. https://doi.org/10.1137/15M103618X.

[26] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum, 1972, doi:10.1007/978-1-4684-2001-2_9.

[27] Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2017*, pages 224–237. ACM Press, 2017, doi:10.1145/3055399.3055456.

[28] 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, oct 2007, doi:10.1007/s00224-007-9089-3.

[29] Jesper Nederlof. Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 713–725, 2009, doi:10.1007/978-3-642-02927-1_59.

[30] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, oct 2014, doi:10.1109/focs.2014.37.

[31] Karthik C. Srikanta, Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5), August 2019. ISSN 0004-5411, doi:10.1145/3325116.

[32] Ondřej Suchý. Extending the kernel for planar steiner tree to the number of steiner vertices. *Algorithmica*, 79(1):189–210, 2017, doi:10.1007/s00453-016-0249-1.

[33] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011, doi:10.1017/cbo9780511921735.

[34] Alexander Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993, doi:10.1007/BF01187035.