
Proceedings Paper:
Feldmann, A.E. and Marx, D. (2016) The complexity landscape of fixed-parameter directed steiner network problems. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y. and Sangiorgi, D., (eds.) Leibniz International Proceedings in Informatics, LIPIcs. 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), 12-15 Jul 2016, Rome, Italy. Leibniz International Proceedings in Informatics, 55. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 27:1-27:14. ISBN: 978-3-95977-013-2. ISSN: 1868-8969. EISSN: 1868-8969.

https://doi.org/10.4230/LIPIcs.ICALP.2016.27

The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems*

Andreas Emil Feldmann[1] and Dániel Marx[2]

1 **Department of Applied Mathematics, Charles University, Prague, Czech Republic; and**
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
feldmann.a.e@gmail.com

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
dmarx@cs.bme.hu

Abstract

Given a directed graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of terminal pairs, the Directed Steiner Network problem asks for a minimum-cost subgraph of G that contains a directed $s_i \to t_i$ path for every $1 \leq i \leq k$. The special case Directed Steiner Tree (when we ask for paths from a root r to terminals $t_1, \ldots, t_k$) is known to be fixed-parameter tractable parameterized by the number of terminals, while the special case Strongly Connected Steiner Subgraph (when we ask for a path from every t_i to every other t_j) is known to be W[1]-hard parameterized by the number of terminals. We systematically explore the complexity landscape of directed Steiner problems to fully understand which other special cases are FPT or W[1]-hard. Formally, if $\mathcal{H}$ is a class of directed graphs, then we look at the special case of Directed Steiner Network where the list $(s_1, t_1), \ldots, (s_k, t_k)$ of requests form a directed graph that is a member of $\mathcal{H}$. Our main result is a complete characterization of the classes $\mathcal{H}$ resulting in fixed-parameter tractable special cases: we show that if every pattern in $\mathcal{H}$ has the combinatorial property of being "transitively equivalent to a bounded-length caterpillar with a bounded number of extra edges," then the problem is FPT, and it is W[1]-hard for *every* recursively enumerable $\mathcal{H}$ not having this property. This complete dichotomy unifies and generalizes the known results showing that Directed Steiner Tree is FPT [Dreyfus and Wagner, *Networks* 1971], Strongly Connected Steiner Subgraph is W[1]-hard [Guo et al., *SIAM J. Discrete Math.* 2011], and Directed Steiner Network is solvable in polynomial-time for constant number of terminals [Feldman and Ruhl, *SIAM J. Comput.* 2006], and moreover reveals a large continent of tractable cases that were not known before.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Directed Steiner Tree, Directed Steiner Network, fixed-parameter tractability, dichotomy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.27

1 Introduction

Steiner Tree is a basic and well-studied problem of combinatorial optimization: given an edge-weighted undirected graph G and a set $R \subseteq V(G)$ of terminals, it asks for a minimum-

* Supported by ERC Starting Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 27; pp. 27:1–27:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

cost tree connecting the terminals. The problem is well known to be NP-hard, in fact, it was one of the 21 NP-hard problems identified by Karp's seminal paper [22]. There is a large literature on approximation algorithms for Steiner Tree and its variants, resulting for example in constant-factor approximation algorithms for general graphs and approximation schemes for planar graphs (see [8, 15, 9, 4, 3, 2, 7, 26, 24, 23, 1, 17]). From the viewpoint of parameterized algorithms, the first result is the classic dynamic-programming algorithm of Dreyfus and Wagner [17] from 1971, which solves the problem with $k = |R|$ terminals in time $3^k \cdot n^{O(1)}$, showing that the problem is fixed-parameter tractable (FPT) parameterized by the number of terminals. More recently, the running time was improved to $2^k \cdot n^{O(1)}$ by Björklund et al. [5] using the technique of fast subset convolution. Steiner Forest is the generalization where the input contains an edge-weighted graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of pairs of terminals and the task is to find a minimum-cost subgraph containing an s_i–t_i path for every $1 \leq i \leq k$. The fixed-parameter tractability of Steiner Forest follows from the observation that the connected components of the solution induces a partition on the set $\{s_1, \ldots, s_k, t_1, \ldots, t_k\}$ of terminals, and hence we can solve the problem by for example trying every partition and invoking a Steiner Tree algorithm for each class of the partition.

On directed graphs, Steiner problems can become significantly harder, and while there is a richer landscape of variants, very few results are known [21, 11, 18, 10, 27, 14, 13]. A natural and well-studied generalization of Steiner Tree to directed graphs is Directed Steiner Tree (DST), where an arc-weighted directed graph G and terminals $r, t_1, \ldots, t_k$ are given and the tasks is to find a minimum-cost subgraph containing an $r \to t_i$ path for every $1 \leq i \leq k$. Using essentially the same techniques as in the undirected case [5, 17], one can show that this problem is also FPT parameterized by the number of terminals. An equally natural generalization of Steiner Tree to directed graphs is the Strongly Connected Steiner Subgraph (SCSS) problem, where an arc-weighted directed graph G with terminals $t_1, \ldots, t_k$ is given, and the task is to find a minimum-cost subgraph containing a $t_i \to t_j$ path for any $1 \leq i, j \leq k$ with $i \neq j$. Guo et al. [21] showed that, unlike DST, the SCSS problem is W[1]-hard parameterized by the number k of terminals (see also [14]). A common generalization of DST and SCSS is the Directed Steiner Network (DSN) problem (also called Directed Steiner Forest or Point-to-Point Connection), where an arc-weighted directed graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of terminal pairs are given and the task is to find a minimum-cost subgraph containing an $s_i \to t_i$ path for every $1 \leq i \leq k$. Being a generalization of SCSS, the Directed Steiner Network problem is also W[1]-hard, but Feldman[1] and Ruhl [18] showed that the problem is solvable in time $n^{O(k)}$, that is, in polynomial time for every constant k.

Besides Directed Steiner Tree, what other special cases of Directed Steiner Network are fixed-parameter tractable? Our main result gives a complete map of the complexity landscape of directed Steiner problems, precisely describing all the FPT/W[1]-hard variants and revealing highly non-trivial generalizations of Directed Steiner Tree that are still tractable. Our results are expressed in the following formal framework. The pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in the input of DSN can be interpreted as a directed (unweighted) *pattern graph* on a set R of terminals. If this pattern graph is an out-star, then the problem is precisely DST; if it is a bidirected clique, then the problem is precisely SCSS. More generally, if $\mathcal{H}$ is any class of graphs, then we define the Directed Steiner $\mathcal{H}$-Network ($\mathcal{H}$-DSN) problem as the restriction of DSN where the pattern graph is a member of $\mathcal{H}$. That is, the

[1] We note that Jon Feldman (co-author of [18]) is not the same person as Andreas Emil Feldmann (co-author of this paper).

Figure 1 Two 4-caterpillars: an out- (left) and an in-caterpillar (right).

input of $\mathcal{H}$-DSN is an arc-weighted directed graph G, a set $R \subseteq V(G)$ of terminals, and an unweighted directed graph $H \in \mathcal{H}$ on R; the task is to find a minimum-cost network $N \subseteq G$ such that N contains an $s \to t$ path for every $st \in E(H)$.

We give a complete characterization of the classes $\mathcal{H}$ for which $\mathcal{H}$-DSN is FPT or W[1]-hard. We need the following definition of "almost-caterpillar graphs" to describe the borderline between the easy and hard cases (see Figure 1).

▶ **Definition 1.** A λ_0-*caterpillar* graph is constructed as follows. Take a directed path $(v_1, \ldots, v_{\lambda_0})$ from v_1 to v_{λ_0}, and let $W_1, \ldots, W_{\lambda_0}$ be pairwise disjoint vertex sets such that $v_i \in W_i$ for each $i \in \{1, \ldots, \lambda_0\}$. Now add edges such that either every W_i forms an out-star with root v_i, or every W_i forms an in-star with root v_i. In the former case we also refer to the resulting λ_0-caterpillar as an *out-caterpillar*, and in the latter as an *in-caterpillar*. A 0-caterpillar is the empty graph. The class $\mathcal{C}_{\lambda,\delta}$ contains all directed graphs H such that there is a set of edges $F \subseteq E(H)$ of size at most δ for which the remaining edges $E(H) \setminus F$ span a λ_0-caterpillar for some $\lambda_0 \leq \lambda$.

If there is an $s \to t$ path in the pattern graph H for two terminals $s, t \in R$, then adding the edge st to H does not change the problem: connectivity from s to t is already implied by H, hence adding this edge does not change the feasible solutions. That is, adding a transitive edge does not change the solution space and hence it is really only the transitive closure of the pattern H that matters. We say that two pattern graphs are *transitively equivalent* if their transitive closures are isomorphic. We denote the class of patterns that are transitively equivalent to some pattern of $\mathcal{C}_{\lambda,\delta}$ by $\mathcal{C}^*_{\lambda,\delta}$. Our main result is a sharp dichotomy saying that $\mathcal{H}$-DSN is FPT if every pattern of $\mathcal{H}$ is transitively equivalent to an almost-caterpillar graph and it is W[1]-hard otherwise. We measure the running time in λ, δ, and the *vertex cover number* τ of the input pattern H, i.e. τ is the size of the smallest vertex subset W of H such that every edge of H is incident to a vertex of W.

▶ **Theorem 2.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns.*

1. *If there are constants λ and δ such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$, then $\mathcal{H}$-DSN with parameter $k = |R|$ is FPT and can be solved in $2^{O(k+\max\{\omega^2,\, \tau\omega \log \omega)\})} n^{O(\omega)}$ time, where $\omega = (1+\lambda)(\lambda+\delta)$ and τ is the vertex cover number of the given input pattern $H \in \mathcal{H}$.*
2. *Otherwise, if there are no such constants λ and δ, then the problem is* W[1]*-hard for parameter k.*

Invoking Theorem 2 with specific classes $\mathcal{H}$, we can obtain algorithmic or hardness results for specific problems. For example, we may easily recover the following facts:

- If $\mathcal{H}_{\text{DST}}$ is the class of all out-stars, then $\mathcal{H}_{\text{DST}}$-DSN is precisely the DST problem. As $\mathcal{H}_{\text{DST}} \subseteq \mathcal{C}^*_{1,0}$ holds, Theorem 2(1) recovers the fact that DST can be solved in time $2^{O(k)} n^{O(1)}$ and is hence FPT parameterized by the number $k = |R|$ of terminals [17, 5].
- If $\mathcal{H}_{\text{SCSS}}$ is the class of all bidirected cliques, then $\mathcal{H}_{\text{SCSS}}$-DSN is precisely the SCSS problem. One can observe that $\mathcal{H}_{\text{SCSS}}$ is not contained in $\mathcal{C}^*_{\lambda,\delta}$ for any constants λ, δ (for example, because every graph in $\mathcal{C}_{\lambda,\delta}$ has at most $\lambda + 2\delta$ vertices with both positive

in-degree and positive out-degree, and this remains true also for the graphs in $\mathcal{C}^*_{\lambda,\delta}$). Hence Theorem 2(2) recovers the fact that SCSS is W[1]-hard [21].

- Let $\mathcal{H}_d$ be the class of directed graphs with at most d edges. As $\mathcal{H}_d \subseteq \mathcal{C}^*_{0,d}$ holds, Theorem 2(1) recovers the fact that Directed Steiner Network with at most d requests is polynomial-time solvable for every constant d [18]. Note that any pattern of $\mathcal{H}_{\text{SCSS}}$ is transitively equivalent to a bidirected star, which has vertex cover number $\tau = 1$. Hence for the important spacial case of SCSS, our algorithm recovers the running time of $2^{O(d \log d)} n^{O(d)} = n^{O(d)}$ given in [18].
- Very recently, Suchý [25] studied the following generalization of DST and SCSS: in the q-Root Steiner Tree (q-RST) problem, a set of q roots and a set of k leaves are given, and the task is to find a minimum-cost network where the roots are in the same strongly connected component and every leaf can be reached from every root. Building on the work of [18], Suchý [25] presented an algorithm with running time $2^{O(k)} \cdot n^{O(q)}$ for this problem, which shows that it is FPT for every constant q. Let $\mathcal{H}_{q\text{-RST}}$ be the class of directed graphs that are obtained from an out-star by making $q-1$ of the edges bidirected. Observe that $\mathcal{H}_{q\text{-RST}}$ is a subset of $\mathcal{C}_{1,q-1}$, that q-RST can be expressed by an instance of $\mathcal{H}_{q\text{-RST}}$-DSN, and that any pattern of $\mathcal{H}_{q\text{-RST}}$ has vertex cover number $\tau = 1$. Thus Theorem 2(1) implies that q-RST can be solved in time $2^{O(k+q \log q)} \cdot n^{O(q)} = 2^{O(k)} \cdot n^{O(q)}$, recovering the fact that it is FPT for every constant q.

Thus the algorithmic side of Theorem 2 unifies and generalizes three algorithmic results: the fixed-parameter tractability of DST (which is based on dynamic programming on the tree structure of the solution) and q-RST (which is based on simulating a "pebble game"), and also the polynomial-time solvability of DSN with constant number of requests (which also is based on simulating a "pebble game"). Let us point out that our algorithmic results are significantly more general than just the unification of these three results: the generalization from stars to bounded-length caterpillars is already a significant extension and very different from earlier results. We consider it a major success of the systematic investigation that, besides finding the unifying algorithmic ideas generalizing all previous results, we were able to find tractable special cases in an unexpected new direction.

There is a surprising non-monotonicity in the classification result of Theorem 2. As DST is FPT and SCSS is W[1]-hard, one could perhaps expect that $\mathcal{H}$-DSN becomes harder as the pattern become denser. However, it is possible that the addition of further requests makes the problem easier. For example, if $\mathcal{H}$ contains every graph that is the vertex-disjoint union of two out-stars, then $\mathcal{H}$-DSN is classified to be W[1]-hard by Theorem 2(2). However, if we consider those graphs where there is also a directed edge from the center of one star to the other star, then these graphs are 2-caterpillars (i.e., contained in $\mathcal{C}_{2,0}$) and hence $\mathcal{H}$-DSN becomes FPT by Theorem 2(1). This unexpected non-monotonicity further underlines the importance of completely mapping the complexity landscape of the problem area: without complete classification, it would be very hard to predict what other tractable/intractable special cases exist.

We mention that one can also study the vertex-weighted version of the problem, where the input graph has weights on the vertices and the goal is to minimize the total vertex-weight of the solution. In general, vertex-weighted problems can be more challenging than edge-weighted variants [15, 4, 23, 12]. However, for general directed graphs, there are easy transformations between the two variants. Thus the results of this paper can be interpreted for the vertex-weighted version as well.

1.1 Our techniques

We prove Theorem 2 the following way. In Section 2, we first establish the combinatorial bound that there is a solution whose cutwidth, and hence also (undirected) treewidth, is bounded by the number of requests.

▶ **Theorem 3.** *A minimal solution M to a pattern H has cutwidth at most $7m$ if $m = |E(H)|$.*

Then in Section 3 we go on to generalize this to almost-caterpillars, showing that if the pattern is in $\mathcal{C}^*_{\lambda,\delta}$, then the (undirected) treewidth can be bounded in λ and δ.

▶ **Theorem 4.** *The treewidth of a minimal solution to any pattern graph in $\mathcal{C}^*_{\lambda,\delta}$ is at most $7(1+\lambda)(\lambda+\delta)$.*

This combinatorial bound can be exploited in an algorithm that restricts the search for a bounded-treewidth solution.

▶ **Theorem 5.** *Let an instance of $\mathcal{H}$-DSN be given by a graph G with n vertices, and a pattern H on k terminals with vertex cover number τ. If the optimum solution to H in G has treewidth ω then the optimum can be computed in time $2^{O(k+\max\{\omega^2,\, \tau\omega\log\omega\})}n^{O(\omega)}$.*

Combining Theorem 4 and Theorem 5 proves the algorithmic side of Theorem 2. We remark that the proof is completely self-contained (with the exception of some basic facts on treewidth) and in particular we do not build on the algorithms of Feldman and Ruhl [18]. As combining Theorem 3 and Theorem 5 already proves that DSN with a constant number of requests can be solved in polynomial time, as a by-product this gives an independent proof for the result of Feldman and Ruhl [18]. One can argue which algorithm is simpler, but perhaps our proof (with a clean split of a combinatorial and an algorithmic statement) is more methodological and better reveals the underlying reason why the problem is tractable.

Finally, in Section 4 we show that whenever the patterns in $\mathcal{H}$ are not transitively equivalent to almost-caterpillars, the problem is W[1]-hard. We first show that there is only a small number of obstacles for not being transitively equivalent to almost-caterpillars: the graph class contains (possibly after identification of vertices) arbitrarily large strongly connected graphs, pure diamonds, or flawed diamonds (see Lemma 22 for the precise statement). We provide a separate W[1]-hardness proof for each of these cases, completing the proof of the hardness side of Theorem 2.

Due to space limitations we defer all missing proofs to the full version of this extended abstract, including the algorithm that implies Theorem 5.

2 The cutwidth of minimal solutions for bounded-size patterns

Consider a *minimal* solution M to an instance of $\mathcal{H}$-DSN, in which no edge can be removed without making the solution infeasible. The goal of this section is to prove Theorem 3: we bound the *cutwidth* of a minimal solution M to a pattern H in terms of $m = |E(H)|$. A *layout* of a graph G is an injective function $\psi : V(G) \to \mathbb{N}$ inducing a total order on the vertices of G. Given a layout, we define the set $V_i = \{v \in V(G) \mid \psi(v) \leq i\}$ and say that an edge *crosses the cut* $(V_i, \overline{V}_i)$ if it has one endpoint in V_i and one endpoint in $\overline{V}_i := V(G) \setminus V_i$. The *cutwidth of the layout* is the maximum number of edges crossing any cut $(V_i, \overline{V}_i)$ for any $i \in \mathbb{N}$. The cutwidth of a graph is the minimum cutwidth over all its layouts.

Like Feldman and Ruhl [18], we consider the two extreme cases of *directed acyclic graphs (DAGs)* and *strongly connected components (SCCs)* in our proof. Contracting all SCCs of M without removing parallel edges sharing the same head and tail, but removing

the resulting self-loops, results in a directed acyclic multi-graph D, the so-called *condensation graph* of M. We bound the cutwidth of D and the SCCs of M separately, and then put together these two bounds to obtain a bound for the cutwidth of M. As we will see, bounding the cutwidth of the acyclic multi-graph D and putting together the bounds are fairly simple. The main technical part is bounding the cutwidth of the SCCs.

We will need two simple facts about cutwidth. First, the cutwidth of an acyclic multi-graph can be bounded using the existence of a *topological ordering* of the vertices. That is, for any acyclic graph G there is an injective function $\varphi : V(G) \to \mathbb{N}$ such that $\varphi(u) < \varphi(v)$ if $uv \in E(G)$. Note that such a function in particular is a layout.

▶ **Lemma 6.** *The layout given by a topological ordering φ_D of an acyclic directed multi-graph D that is the union of m paths, has cutwidth at most m.*

▶ **Lemma 7.** *Let G be a directed graph and D be its condensation multi-graph. If the cutwidth of D is x and the cutwidth of every SCC of G is at most y, then the cutwidth of G is at most $x + y$.*

▶ **Lemma 8.** *Any SCC U of a minimal solution M to a pattern H with at most m edges has cutwidth at most $6m$.*

Proof. First we establish that U is a minimal solution to a certain pattern.

▶ **Claim 9.** *U is a minimal solution to a pattern H_U with at most m edges.*

Let R_U be the terminals in the pattern H_U given by Claim 9 and let us select an arbitrary root $t \in R_U$. Note that H_U has at most m edges, hence $|R_U| \leq 2m$. Let S_{in} (resp., S_{out}) be an in-star (resp., out-star) connecting t with every other vertex of R_U. As U is a strongly connected graph containing every vertex of R_U, it is also a solution to the pattern S_{in} on R_U. Let us select an $A_{in} \subseteq U$ that is a minimal solution to S_{in}; it is not hard to see that A_{in} is an in-arborescence with at most $2m$ leaves. Similarly, let $A_{out} \subseteq U$ be an out-arborescence that is a minimal solution to S_{out}. Observe that U has to be exactly $A_{in} \cup A_{out}$: if there is an edge $e \in E(U)$ that is not in $A_{in} \cup A_{out}$, then $U \setminus e$ still contains a path from every vertex of R_U to every other vertex of R_U though t, contradicting the fact that U is a minimal solution to pattern H_U.

Let Z be the set of edges obtained by reversing the edges in $E(A_{in}) \setminus E(A_{out})$. As reversing edges does not change the cutwidth, bounding the cutwidth of $A_{out} \cup Z$ will also imply a bound on the cutwidth of $U = A_{in} \cup A_{out}$.

▶ **Claim 10.** *The union $A_{out} \cup Z$ is a directed acyclic graph.*

Claim 10 implies a topological ordering on the vertices of $A_{out} \cup Z$. This order can be used as a layout for U. Using some more structural insights, the number of edges crossing a given cut can be bounded in the number of edges of the pattern graph, as the following claim shows.

▶ **Claim 11.** *Any topological ordering φ of the graph $A_{out} \cup Z$ has cutwidth at most $6m$.*

As the underlying undirected graph of U and $A_{out} \cup Z$ are the same, Claim 11 implies that the cutwidth of U is at most $6m$. This completes the proof of Lemma 8. ◀

The proof of Theorem 3 follows easily from putting together the ingredients. We remark that the bound on the cutwidth in Claim 11 is asymptotically tight: Take a constant degree expander on m vertices. It has treewidth $\Omega(m)$ [20], and so its cutwidth is at least as large. Now bi-direct each (undirected) edge $\{u, v\}$ by replacing it with the directed edges uv and vu.

Next subdivide every edge uv to obtain edges ut and tv for a new vertex t, and make t a terminal of R. This yields a strongly connected instance G. The pattern graph H for this instance is a cycle on R, which has $O(m)$ edges, since the terminals are subdivision points of bi-directed edges of a constant degree graph with m vertices. As H is strongly connected, every minimal solution to H contains the edges ut and tv incident to each terminal t. Thus a minimal solution contains all of G and has cutwidth $\Omega(m)$. Since G is strongly connected, it also contains the required arborescences A_{in} and A_{out}.

3 The treewidth of minimal solutions to almost-caterpillar patterns

In this section, we prove that any minimal solution M to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$ has the following structure.

▶ **Theorem 12.** *A minimal solution M to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$ consists of a subgraph M^c that is a minimal solution to a sub-pattern H^c of H with at most $(1+\lambda)(\lambda+\delta)$ edges, and a forest $M \setminus M^c$ of out-arborescences, each of which intersects M^c only at the root.*

According to Theorem 3, the cutwidth of the core M^c is therefore at most $7(1+\lambda)(\lambda+\delta)$. It is well known [6] that the cutwidth is an upper bound on the treewidth of a graph, and so also the treewidth of M^c is at most $7(1+\lambda)(\lambda+\delta)$. It is easy to see that attaching any number of arborescences to M^c does not increase the treewidth. Thus we obtain Theorem 4, which is the basis for our algorithm to solve $\mathcal{H}$-DSN in case every pattern of $\mathcal{H}$ is transitively equivalent to an almost-caterpillar.

In particular, when adding δ edges to the pattern of the DST problem, which is a single out-star, i.e., a 1-caterpillar, then the pattern becomes a member of $\mathcal{C}_{1,\delta}$ and hence our result implies a linear treewidth bound of $O(\delta)$. The example given at the end of Section 2 also shows that there are patterns $H \in \mathcal{C}_{\lambda,\delta}$ for which every minimal solution has treewidth $\Omega(\lambda+\delta)$: just consider the case when H is a cycle of length $\lambda+\delta$ (i.e., it contains a trivial caterpillar graph). One interesting question is whether the treewidth bound of $7(1+\lambda)(\lambda+\delta)$ in Theorem 4 is tight. We conjecture that the treewidth of any minimal solution to a pattern graph $H \in \mathcal{C}^*_{\lambda,\delta}$ actually is $O(\lambda+\delta)$.

Proof (of Theorem 12). Let M be a minimal solution to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$. Since every pattern in $\mathcal{C}^*_{\lambda,\delta}$ has a transitively equivalent pattern in $\mathcal{C}_{\lambda,\delta}$ and replacing a pattern with a transitively equivalent pattern does not change the space of feasible solutions, we may assume that H is actually in $\mathcal{C}_{\lambda,\delta}$, i.e., H consists of a caterpillar of length at most λ and δ additional edges.

The statement is trivial if $|E(H)| \leq \delta$. Otherwise, according to Definition 1, H contains a λ_0-caterpillar for some $1 \leq \lambda_0 \leq \lambda$ and at most δ additional edges. Hence let us fix a set F of at most δ edges of H, such that the remaining edges of H form a λ_0-caterpillar C, for some $1 \leq \lambda_0 \leq \lambda$, with a path $(v_1, \ldots, v_{\lambda_0})$ on the roots of the stars S_i. We only consider the case when C is an out-caterpillar as the other case is symmetric, i.e., every S_i is an out-star. Define $I = H \setminus \bigcup_{i=1}^{\lambda_0} S_i$ to be all of H except the stars. Note that $|E(I)| \leq \lambda+\delta$. We fix a subgraph M_I of M that is a minimal solution to the sub-pattern I, and for every $st \in E(I)$ we fix a path P_{st} in M_I. Note that M_I is the union of these at most $\lambda+\delta$ paths, since M_I is a minimal solution. For each star S_i, let us consider a minimal solution $M_{S_i} \subseteq M$ to S_i; note that M_{S_i} has to be an out-arborescence.

For $i \in \{1, \ldots, \lambda_0\}$, let ℓ be a leaf of S_i, and let e be an edge of M. If $M \setminus e$ has no path from v_i to ℓ, then we say that e is *ℓ-necessary*. More generally, we say that e is *i-necessary* if e is ℓ-necessary for some leaf ℓ of S_i.

▶ **Claim 13.** *Let P be a path in M, and for some $i \in \{1, \ldots, \lambda_0\}$, let $W_i \subseteq E(M)$ contain all i-necessary edges f for which $f \notin E(P)$, but the head of f is a vertex of P. Then there exists one leaf ℓ of S_i such that every $f \in W_i$ is ℓ-necessary.*

Using this observation, we identify the core M^c of M using the at most $\lambda + \delta$ paths P_{st} that make up M_I, and then selecting an additional at most λ_0 paths for each P_{st}. To construct M^c together with its pattern graph H^c, we initially let $M^c = M_I$ and $H^c = I$ and repeat the following step for every $st \in E(I)$ and $1 \leq i \leq \lambda_0$. For a given st and i, let us check if there are i-necessary edges $f \notin E(P_{st})$ that have their heads on the path $P_{st} \subseteq M_I$. If so, then by Claim 13 all these edges are ℓ-necessary for some leaf ℓ of S_i. We add an arbitrary path of M from v_i to ℓ (which contains all these edges) to M^c and add the edge $v_i\ell$ to H^c. After repeating this step for every $st \in E(H)$ and i, we remove superfluous edges from M^c: as long as there is an edge $e \in E(M^c)$, which can be removed while maintaining feasibility for the pattern H^c, i.e., for every $vw \in E(H^c)$ there is a $v \to w$ path in M^c not containing e, we remove e. Finally, we remove any isolated vertices from M^c.

Note that the resulting network M^c is a minimal solution to H^c by construction. Also note that H^c contains at most $\lambda + \delta$ edges from I and at most $\lambda_0 \leq \lambda$ additional edges for each edge of I, so that $|E(H^c)| \leq (1 + \lambda)(\lambda + \delta)$. We prove that the remaining graph $M^c \setminus E(M)$ consists of arborescences, each of which intersects M^c only at the root. For this, we rely on the following key observation.

▶ **Claim 14.** *If a vertex u has at least two incoming edges in M, then every such edge is in the core M^c.*

Proof. First we show that there is an $st \in E(I)$ such that every $s \to t$ path in M goes through u. Suppose for contradiction that for every $st \in E(I)$ there is a path from s to t in M avoiding u. Since M is a minimal solution, the edges entering u must then be needed for some stars S_i of the pattern H instead. Let e and f be two edges entering u. As e and f have the same head, they cannot be part of the same out-arborescence M_{S_i}. Therefore, there are indices $i < j$ such that (w.l.o.g.) e is i-necessary and f is j-necessary.

There is a path in M from the root v_i of S_i to the root v_j of S_j, due to the path $(v_1, \ldots, v_{\lambda_0})$ in the caterpillar $C \subseteq H$. Since path $(v_1, \ldots, v_{\lambda_0})$ is part of I, our assumption on e and f implies that there is a path P in M from v_i to v_j that avoids both e and f. As $f \in E(M_{S_j})$, there is a path Q in M starting in v_j and passing through f. This path cannot contain e, as e and f have the same head u. The existence of P and Q implies that u can be reached from v_i by a path through v_j and f, avoiding the edge e. Thus for any edge $v_i\ell \in E(S_i)$, if there is a $v_i \to \ell$ path going through e (and hence vertex u), then it can be rerouted to avoid e and use edge f instead. This however contradicts the fact that e is i-necessary.

We have proved that there is an $st \in E(I)$ such that every $s \to t$ path in M goes through u. Suppose that there is an edge $e \notin E(M^c)$ entering u. If e is needed for some $s't' \in E(I)$ in M, then e is also present in M^c, and we are done. Otherwise, as M is a minimal solution, edge e is i-necessary for some $i \in \{1, \ldots, \lambda_0\}$. Consider now the step in the construction of M^c when we considered $st \in E(I)$ and integer i. As we have shown, the $s \to t$ path P_{st} goes through u. Thus e is an i-necessary edge not in $E(P_{st})$ such that its head is on P_{st}. This means that we identified a leaf ℓ of S_i such that e is ℓ-necessary, introduced $v_i\ell$ into H^c, and added a $v_i \to \ell$ path to H^c, which had to contain e. Moreover, since all paths from v_i to ℓ in M pass through e, edge e then remains in M^c when removing superfluous edges. ◀

We are now ready to show that every component of the remaining part is an out-arborescence and intersects the core only in a single vertex.

▶ **Claim 15.** *The remaining graph $M^+ := M \setminus E(M^c)$ is a forest of out-arborescences, each of which intersects M^c only at the root.*

Since we have already established that M^c is a minimal solution to H^c with $|E(H^c)| \leq (1+\lambda)(\lambda+\delta)$, Claim 15 completes the proof of Theorem 12. ◀

4 Characterizing the hard cases

We now turn to proving the second part of Theorem 2, i.e., that $\mathcal{H}$-DSN is W[1]-hard for every class $\mathcal{H}$ where the patterns are not transitively equivalent to almost-caterpillars.

▶ **Theorem 16.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns for which there are no constants λ and δ such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$. Then the problem $\mathcal{H}$-DSN is* W[1]*-hard for parameter k.*

A major technical simplification is to assume that the class $\mathcal{H}$ is closed under identifying terminals and transitive equivalence. As we show in Section 4.1, this assumption is not really restrictive: it is sufficient to prove hardness for the closure of $\mathcal{H}$ under identification and transitive equivalence, since any W[1]-hardness result for the closure can be transferred to $\mathcal{H}$. For classes closed under these operations, it is possible to give an elegant characterization of the classes that are not almost-caterpillars. There are only a few very specific reasons why a class $\mathcal{H}$ is not in $\mathcal{C}^*_{\lambda,\delta}$ for any λ and δ: either $\mathcal{H}$ contains every directed cycle, or $\mathcal{H}$ contains every "pure diamond," or $\mathcal{H}$ contains every "flawed diamond" (see Section 4.2 for the precise definitions). Then in Section 4.3, we provide a W[1]-hardness proof for each of these cases, completing the hardness part of Theorem 2.

4.1 Closed classes

We define the operation of *identifying terminals* in the following way: given a partition $\mathcal{V}$ of the vertex set $V(H)$ of a pattern graph H, each set $W \in \mathcal{V}$ is identified with a single vertex of W, after which any resulting isolated vertices and self-loops are removed, while parallel edges having the same head and tail are replaced by only one copy of that edge. A class of patterns is *closed* under this operation if for any pattern H in the class, all patterns that can be obtained by identifying terminals are also in the class. Similarly, we say that a class $\mathcal{H}$ is closed under transitive equivalence if whenever H and H' are two transitively equivalent patterns such that $H \in \mathcal{H}$, then H' is also in $\mathcal{H}$. The closure of the class $\mathcal{H}$ under identifying terminals and transitive equivalence is the smallest closed class $\mathcal{H}' \supseteq \mathcal{H}$. It is not difficult to see that any member of the closure can be obtained by a replacement with a transitively equivalent pattern and a single application of identifying terminals.

The following lemma shows that if we want to prove W[1]-hardness for a class, then it is sufficient to prove hardness for its closure. More precisely, due to an slight technicality, the actual statement we prove is that it is sufficient to prove W[1]-hardness for a decidable subclass of the closure.

▶ **Lemma 17.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns, let $\mathcal{H}'$ be the closure of $\mathcal{H}$ under identifying terminals and transitive equivalence, and let $\mathcal{H}''$ be a decidable subclass of $\mathcal{H}'$. There is a parameterized reduction from $\mathcal{H}''$-DSN to $\mathcal{H}$-DSN with parameter k.*

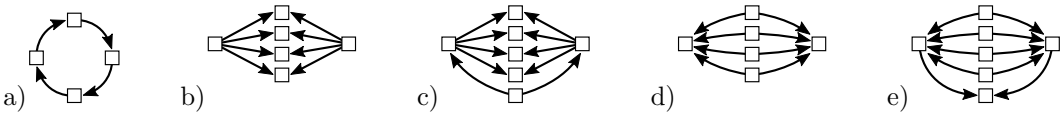

Figure 2 The obstruction appearing in Lemma 19: a) a directed cycle of length 4, b) a pure 4-out-diamond, c) a flawed 4-out-diamond, d) a pure 4-in-diamond, e) a flawed 4-in-diamond.

4.2 Obstructions: SCCs and diamonds

To show the hardness for a closed class that is not the subset of $\mathcal{C}^*_{\lambda,\delta}$ for any λ and δ, we will characterize such a class in terms of the occurrence of arbitrarily large cycles, and another class of patterns called "diamonds" (cf. Figure 2).

▶ **Definition 18.** A *pure α-diamond* graph is constructed as follows. Take a vertex set L of size $\alpha \geq 1$, and two additional vertices r_1 and r_2. Now add edges such that L is the leaf set of either two in-stars or two out-stars S_1 and S_2 with roots r_1 and r_2, respectively. If we add an additional vertex x with edges r_1x and r_2x if S_1 and S_2 are in-stars, and edges xr_1 and xr_2 otherwise, the resulting graph is a *flawed α-diamond*. We refer to both pure α-diamonds and flawed α-diamonds as *α-diamonds*. If S_1 and S_2 are in-stars we also refer to the resulting α-diamonds as *in-diamonds*, and otherwise as *out-diamonds*.

The goal of this section is to prove the following useful characterization precisely describing classes that are not almost-caterpillars.

▶ **Lemma 19.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. Exactly one of the following statements is true:*

- *$\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ for some constants λ and δ.*
- *$\mathcal{H}$ contains every directed cycle, or every pure in-diamond, or every pure out-diamond, or every flawed in-diamond, or every flawed out-diamond.*

For the proof of Theorem 16, we only need the fact that at least one of these two statements hold: if the class $\mathcal{H}$ is not in $\mathcal{C}^*_{\lambda,\delta}$, then we can prove hardness by observing that $\mathcal{H}$ contains one of the hard classes. For the sake of completeness, we give a simple proof that the two statements cannot hold simultaneously in the full version of this extended abstract.

Showing that at least one of the two statements of Lemma 19 hold is not as easy to prove. First, the following two lemmas show how a large cycle or a large diamond can be identified if certain structures appear in a pattern. The main part of the proof is to show that if $\mathcal{H}$ contains patterns that are arbitrarily far from being a caterpillar, then one of these two lemmas can be invoked (see Lemma 22).

▶ **Lemma 20.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. If some $H \in \mathcal{H}$ contains a matching of size α, then $\mathcal{H}$ contains a directed cycle of length α.*

Proof. A *matching* of a graph is a subset M of its edges such that no two edges of M share a vertex. A matching $e_1, \ldots, e_\alpha$ of α edges can be transformed into a cycle of length α by identifying the head of e_i and tail of e_{i+1} (and the head of e_α with the tail of e_1). All remaining vertices that do not belong to the cycle can then be identified with any vertex of the cycle, so that the resulting graph consists of the cycle and some additional edges. Since $\mathcal{H}$ is closed under identifying terminals, this graph would be contained in $\mathcal{H}$. As this graph is strongly connected and $\mathcal{H}$ is closed also under transitive equivalence, we can conclude that $\mathcal{H}$ contains a cycle of length α. ◀

Next we give a sufficient condition for the existence of large diamonds. We say that an edge uv of a graph H is *transitively non-redundant* if there is no $u \to v$ path in $H \setminus uv$.

▶ **Lemma 21.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. Let $H \in \mathcal{H}$ be a pattern graph that contains two out-stars (or two in-stars) S_1 and S_2 as induced subgraphs, with at least α edges each and roots r_1 and r_2, respectively. The class $\mathcal{H}$ contains an α-diamond if*

1. *H contains neither a path from r_1 to r_2, nor from r_2 to r_1,*
2. *the leaves of S_1 and S_2 have out-degree 0 (if S_1 and S_2 are out-stars) or in-degree 0 (if S_1 and S_2 are in-stars), and*
3. *the edges of the stars are transitively non-redundant.*

To show that at least one of the two statements of Lemma 19 hold, we prove that if the second statement is false, then the first statement is true. That is, if $\mathcal{H}$ does not contain all cycles (i.e., there is an α_1 such that $\mathcal{H}$ contains no cycle larger than α_1), $\mathcal{H}$ does not contain all pure out-diamonds (i.e., there is an α_2 such that $\mathcal{H}$ contains no pure out-diamond larger than α_2), etc., then $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ for some constants λ and δ. In other words, if we let α to be the maximum of α_1, α_2, etc., then we may assume that $\mathcal{H}$ contains no pure of flawed α-diamond or cycle of length α, and we need to prove $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ under this assumption. Thus the following lemma completes the proof of Lemma 19.

▶ **Lemma 22.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. If for some integer α the class $\mathcal{H}$ contains neither a pure α-diamond, flawed α-diamond, nor a cycle of length α, then there exist constants λ and δ (depending on α) such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$.*

Proof. Suppose that there is such an integer α. Let $\lambda := 2\alpha$ and $\delta := 4\alpha^3 + 6\alpha^2$. Given any $H' \in \mathcal{H}$, we show how a transitively equivalent pattern $H \in \mathcal{C}_{\lambda,\delta}$ can be constructed, implying that H' belongs to $\mathcal{C}^*_{\lambda,\delta}$. A *vertex cover* of a graph is a subset X of its vertices such that every edge is incident to a vertex of X. By Lemma 20, H' cannot contain a matching of size α. It is well-known that if a graph has no matching of size α, then it has a vertex cover of size at most 2α (take the endpoints of any maximal matching). Let us fix a vertex cover X of H' having size at most 2α.

To obtain H from H', we start with a graph H on $V(H')$ having no edges and perform the following three steps.

1. Let us take the transitive closure on the vertex set X in H', i.e., let us introduce into H every edge uv with $u, v \in X$ such that there is a $u \to v$ path in H'.
2. Let us add all edges uv of H' to H for which $u \notin X$ or $v \notin X$.
3. Fixing an ordering of the edges introduced in step 2, we remove transitively redundant edges: following this order, we subsequently remove those edges uv for which there is a path from u to v in the remaining graph H that is not the edge uv itself.

It is clear that H is transitively equivalent to H'. Note that X is a vertex cover of H as well, and hence its complement $I = V(H) \setminus X$ is an *independent set*, i.e. no two vertices of I are adjacent. Let $E_I \subseteq E(H)$ be the set of edges between X and I. In the rest of the proof, we argue that the resulting pattern H belongs to $\mathcal{C}_{\lambda,\delta}$. We show that H can be decomposed into a path $P = (v_1, \ldots, v_{\lambda_0})$ in X, a star S_{v_i} centered at each v_i using the edges in E_I, and a small set of additional edges. This small set of additional edges is constructed in three steps, by considering a sequence of larger and larger sets $F_1 \subseteq F_2 \subseteq F_3$.

As E_I consists of edges between X and I, it can be partitioned into a set of stars with roots in X. The following claim shows that almost all of these edges are directed towards X or almost all of them are directed away from X.

▶ **Claim 23.** *Either there are less than $2\alpha^2$ edges uv in E_I with head in X, or less than $2\alpha^2$ edges uv in E_I with tail in X.*

Assume that the former case of Claim 23 is true, so that the number of edges in E_I with heads in X is bounded by $2\alpha^2$; the other case can be handled symmetrically. We will use the out-stars spanned by E_I for the caterpillar, which means that we obtain an out-caterpillar. We use the set F_1 to account for the edges in E_I with heads in X. Additionally, we will also introduce into F_1 those edges in E_I with tails in X that are adjacent to an edge of the former type. Formally, for any edge $uv \in E_i$ with $v \in X$, we introduce into F_1 every edge of E_I incident to u. After this step, F_1 contains less than $4\alpha^3$ edges, since there are less than $2\alpha^2$ edges $uv \in E_I$ with $v \in X$ and u can only be adjacent to vertices in X, which has size less than 2α.

For any vertex $v \in X$, let S_v denote the out-star formed by the edges of $E_I \setminus F$ incident to v. Let $X' \subseteq X$ contain those vertices $v \in X$ for which S_v has at least α leaves.

▶ **Claim 24.** *For any two distinct $u, v \in X'$, at least one of uv and vu is in H, and the stars S_u and S_v are vertex disjoint.*

We extend F_1 to F_2 by adding all edges of stars S_v with $v \in X \setminus X'$ to F_2. Since X contains less than 2α vertices and we extend F_1 only by stars with less than α edges, this step adds less than $2\alpha^2$ edges, i.e., $|F_2| \leq |F_1| + 2\alpha^2 = 4\alpha^3 + 2\alpha^2$.

By Claim 24, X' induces a *semi-complete* directed graph in H, i.e., at least one of the edges uv and vu exists for every pair $u, v \in X'$. It is well-known that every semi-complete directed graph contains a Hamiltonian path (e.g., [16, Chapter 10, Exercise 1]), and so there is a path $P = (v_1, \ldots, v_{\lambda_0})$ with $\lambda_0 = |X'| \leq 2\alpha = \lambda$ in H on the vertices of X'. We extend F_2 to F_3 by including any edge induced by vertices of X' that is not part of P. There are less than $4\alpha^2$ such edges, and hence we have $|F_3| \leq |F_2| + 4\alpha^2 \leq 4\alpha^3 + 6\alpha^2 = \delta$. The edges of H not in F_3 span the path P and disjoint out-stars S_{v_i} with $i \in \{1, \ldots, \lambda_0\}$, i.e., they form a λ_0-caterpillar. This proves that $H \in \mathcal{C}_{\lambda,\delta}$ and hence $H' \in \mathcal{C}^*_{\lambda,\delta}$, what we had to show. ◀

4.3 Reductions

Lemma 19 implies that in order to prove Theorem 16, we need W[1]-hardness proofs for the class of all directed cycles, the class of all pure in-diamonds, the class of all pure out-diamonds, etc. We provide these hardness proofs and then formally show that they imply Theorem 16.

Let us first consider the case when $\mathcal{H}$ is the class of all directed cycles. Recall that, given an arc-weighted directed graph G and a set $R \subseteq V(G)$ of terminals, the Strongly Connected Steiner Subgraph (SCSS) problem asks for a minimum-cost subgraph that is strongly connected and contains every terminal in R. This problem is known to be W[1]-hard parameterized by the number $k := |R|$ of terminals [21]. We can reduce SCSS to an instance of DSN where the pattern H is a directed cycle on R, which expresses the requirement that all the terminals are in the same strongly connected component of the solution. Thus the W[1]-hardness of SCSS immediately implies the W[1]-hardness of $\mathcal{H}$-DSN if $\mathcal{H}$ contains all directed cycles.

▶ **Lemma 25** (follows from [21]). *If $\mathcal{H}$ is the class of directed cycles, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals.*

Next we turn our attention to classes containing all diamonds. The following reductions are from the W[1]-hard Multicoloured Clique problem [19], in which an undirected graph together with a partition $\{V_1, \ldots, V_k\}$ of its vertices into k sets is given, such that for

any $i \in \{1, \ldots, k\}$ no two vertices of V_i are adjacent. The aim is to find a clique of size k, i.e. a set of pairwise adjacent vertices $\{w_1, \ldots, w_k\}$ with $w_i \in V_i$ for each $i \in \{1, \ldots, k\}$.

▶ **Lemma 26.** *If $\mathcal{H}$ is the class of all pure out-diamonds, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals. The same holds if $\mathcal{H}$ is the class of all pure in-diamonds.*

The reduction for the case when the pattern is a flawed α-diamond is essentially the same as the one for pure α-diamonds, as we show next.

▶ **Lemma 27.** *If $\mathcal{H}$ is the class of all flawed out-diamonds, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals. The same holds if $\mathcal{H}$ is the class of all flawed in-diamonds.*

Given the three reductions above, we can now prove Theorem 16, based on the additional reduction given in Lemma 17. We defer the final proof to the full version of this extended abstract.

References

1. Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.
2. MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21, 2011. doi:10.1145/2027216.2027219.
3. MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Euclidean prize-collecting Steiner forest. *Algorithmica*, 62(3-4):906–929, 2012. doi:10.1007/s00453-011-9491-8.
4. MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Improved approximation algorithms for (budgeted) node-weighted Steiner problems. In *40th International Colloquium on Automata, Languages, and Programming*, pages 81–92, 2013. doi:10.1007/978-3-642-39206-1_8.
5. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 67–74, 2007.
6. Hans L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–125, 1988.
7. Glencora Borradaile, Philip N. Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5(3), 2009. doi:10.1145/1541885.1541892.
8. Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for Euclidean Steiner forest. *ACM Transactions on Algorithms*, 11(3):19:1–19:20, 2015. doi:10.1145/2629654.
9. Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6, 2013. doi:10.1145/2432622.2432628.
10. Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33(1):73–91, 1999. doi:10.1006/jagm.1999.1042.
11. Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Transactions on Algorithms*, 7(2):18, 2011. doi:10.1145/1921659.1921664.

12 Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1265–1274, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283519.

13 Rajesh Hemant Chitnis, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Saeed Seddighin. A tight algorithm for strongly connected Steiner subgraph on two terminals with demands (extended abstract). In *9th International Symposium on Parameterized and Exact Computation*, pages 159–171, 2014. doi:10.1007/978-3-319-13524-3_14.

14 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight bounds for planar strongly connected Steiner subgraph with fixed number of terminals (and extensions). In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.

15 Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Philip N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 10(3):13:1–13:20, 2014. doi:10.1145/2601070.

16 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.

17 S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

18 Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM J. Comput.*, 36(2):543–561, 2006. doi:10.1137/S0097539704441241.

19 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

20 Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.

21 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.

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

23 Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995. doi:10.1006/jagm.1995.1029.

24 Sridhar Rajagopalan and Vijay V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.314909.

25 Ondřej Suchý. On directed steiner trees with multiple roots. To appear in WG 2016. arXiv:1604.05103.

26 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005. doi:10.1137/S0895480101393155.

27 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18(1):99–110, 1997. doi:10.1007/BF02523690.