


An Exact Algorithm for the Steiner Forest Problem

Daniel R. Schmidt¹

Institut für Informatik, Universität zu Köln, Germany

schmidt@informatik.uni-koeln.de

 <https://orcid.org/0000-0001-7381-912X>

Bernd Zey

Fakultät für Informatik, TU Dortmund, Germany

bernd.zey@tu-dortmund.de

François Margot

Carnegie-Mellon-University, Pittsburgh PA, USA

Abstract

The Steiner forest problem asks for a minimum weight forest that spans a given number of terminal sets. The problem has famous linear programming based 2-approximations [1, 15, 20] whose bottleneck is the fact that the most natural formulation of the problem as an integer linear program (ILP) has an integrality gap of 2. We propose new cut-based ILP formulations for the problem along with exact branch-and-bound based algorithms. While our new formulations cannot improve the integrality gap, we can prove that one of them yields stronger linear programming bounds than the two previous strongest formulations: The directed cut formulation [2, 7] and the advanced flow-based formulation by Magnanti and Raghavan [25]. In an experimental evaluation, we show that the linear programming bounds of the new formulations are indeed strong on practical instances and that our new branch-and-bound algorithms outperform branch-and-bound algorithms based on the previous formulations. Our formulations can be seen as a cut-based analogon to [25], whose existence was an open problem.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

Keywords and phrases branch-and-bound algorithms, Steiner network problems

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.70



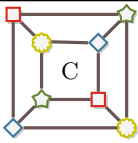
Related Version A preliminary version is available at <https://arxiv.org/abs/1709.01124>.

1 Introduction

The Steiner forest problem (SFP) is one of the fundamental network design problems. Given an edge-weighted undirected graph $G = (V, E)$ and terminal sets $T^1, \dots, T^K \subseteq V$, it asks for a minimum weight forest in G such that the nodes inside each terminal set are connected. Steiner forest is a particularly important problem in the design of real-world communication networks where unwieldy additional constraints make it hard to obtain hard guarantees and clean approximation algorithms. Instead, linear programming based branch-and-bound (B&B) algorithms are a popular choice here: They find an *optimum* solution in (worst-case) exponential time, but can also be run in a heuristic mode where the algorithm stops with a sub-optimum solution after a given time limit. In the latter case, B&B algorithms still

¹ Supported by a fellowship in the Postdoc-Program of the German Academic Exchange Service (DAAD).



formulation			
undirected flow/cut (\mathcal{L}^{uc}), lifted cut (\mathcal{L}^{klsvz}) [23]	2	2	4
layered directed, \mathcal{L}^{dc} [2, 7]	3	2	4
Magnanti-Raghavan [25], \mathcal{L}^{mr}	3	3	6
our extended cut-based, \mathcal{L}^{edc}	3	2.5	5.14
our strengthened extended cut-based, \mathcal{L}^{sedc}	3	3	6
<i>integer optimum</i>	3	3	7

■ **Figure 1** A comparison of lower bounds from LP relaxations. The terminal sets of the three Steiner forest instances are depicted in different shapes (\square , \diamond , \star , and \odot). All edges have unit cost.

provide a *per-instance* quality guarantee by linear programming. This per-instance guarantee is appealing to practitioners: While often only a few selected instances need to be solved, it is common that they defy theoretical analysis. Hence, B&B algorithms complement the fixed parameter tractable (FPT) algorithms which exploit special structures of practical instances. In contrast, however, B&B algorithms allow us to include the real-world constraints without additional analyses and make no structural assumptions. In this way, they provide another important tool for problem solving in practice.

A linear programming based B&B algorithm systematically finds an optimum solution to an integer linear program (ILP). Assume that we are minimizing. The algorithm first removes the integrality requirement, turning the NP-hard ILP into a polynomial time solvable linear programming (LP) relaxation. Any solution to the ILP is a solution to the LP relaxation and thus, the value of any LP solution x^* is a lower bound on the optimum value of the ILP. If x^* is integral, we found an optimum solution to the ILP. Otherwise, there is at least one fractional variable, say $x_i^* \notin \mathbb{Z}$. In any optimum integral solution, we have either $x_i \leq \lfloor x_i^* \rfloor$ or $x_i \geq \lceil x_i^* \rceil$. We create a subproblem for each of the two cases and recurse the algorithm. If at any point in the recursion (the *branch-and-bound tree*) we obtain an integral solution or if a subproblem turns out to be infeasible, we solved the subproblem and we can prune the corresponding branch from the tree. We keep track of the best integral solution we find (the *incumbent*), as it provides an upper bound on the value of an optimum ILP solution. Since the LP value of each subproblem provides a lower bound for its optimum integral value, we can equally prune the tree once the LP value rises above the value of the incumbent. This highlights why strong LP relaxations are paramount for B&B algorithms: The better the LP bounds, and the sooner the LP relaxation becomes integral, the sooner the recursion can be pruned. As different ILP formulations for the same problem yield different LP bounds and finding a strong ILP formulation is an interesting challenge.

While B&B algorithms for the Steiner forest problem all follow the same basic algorithm, they differ on the LP relaxation they employ to generate lower bounds. From a theoretical perspective, almost all LP relaxations for the Steiner forest problem have a worst-case integrality gap of 2, with the only known exception being the *lifted cut relaxation* by Könemann, Leonardi, Schäfer, and van Zwam [23] that achieves a gap of $2 - \epsilon$. Still, the different LP relaxations do not all yield equally good bounds: The bound from the *directed cut* relaxation will never be worse (and often much better) than the bound from the *undirected cut* relaxation, since the former is a specialization of the latter. Likewise, Magnanti and Raghavan [25] show that their *improved flow* relaxation is always as least as good as the

undirected cut relaxation. In that sense, some relaxations are stronger than others, while others are incomparable (see Figure 1): The lifted cut relaxation is at least as strong as the undirected cut formulation, but it may yield stronger *or* weaker bounds than the directed variant. Experiments support this notion of relaxation strength. For instance, it has been observed that the *directed cut formulation* is better suited for B&B algorithms than the *undirected cut formulation* [6], at least for the Steiner tree problem (the special case where $K = 1$). Magnanti and Raghavan obtain particularly strong bounds from the improved flow formulations in their experiments where their B&B algorithm can solve many instances without having to branch. The bounds from the lifted cut relaxation are identical to the ones from the undirected cut relaxation in our experiments.

Our contribution. The above observations seem to turn the improved flow formulation and the directed cut formulation into the canonical choices for a B&B algorithm. Unfortunately, the improved flow relaxation is exponentially large and it is unknown if it can be solved efficiently. The directed cut relaxation is easy to solve, but its bounds are considerably weaker if used for the Steiner forest problem (an analysis is given in Section 2).

We propose a branch-and-bound algorithm that is based on a new, cut-based ILP formulation for the Steiner forest problem. Its LP relaxation is stronger than the improved flow relaxation and as the directed cut relaxation, and therefore, as the undirected cut relaxation as well. In contrast to the improved flow formulation it can be solved in polynomial time. This answers an open problem in [25] which asks for a cut-based ILP formulation that is at least as strong as the improved flow formulation. In our experiments the LP bounds are stronger than what can be achieved from any of the previous relaxations. They can also be computed quickly and reliably. Using known techniques and a computational analysis, we engineer our branch-and-bound algorithm to solve all medium sized and almost all large instances from the benchmark set. The algorithm outperforms B&B algorithms based on the previous formulations. Figure 1 shows a comparison of the formulations on widely-used small example instances.

While we focus on B&B algorithms here, new integer linear programming formulations are interesting beyond exact algorithms: Current approximation algorithms for the Steiner forest problem with a guarantee of 2 are based on iterative rounding [20] and the primal-dual analysis technique [1, 15]; two techniques that rely on strong LP relaxations. Even though our new formulation has an integrality gap of 2 as well and thus cannot directly improve the known LP-based approximation algorithms, we believe that it can inspire new research in this direction.

Related work. The directed cut relaxation for the Steiner tree problem [2, 6, 22] can be trivially extended to the Steiner forest case. It cuts off fractional solutions by imposing a direction on each edge, looking for a rooted directed tree that connects all terminals. In the Steiner *tree* case where only one terminal set exists, this process is straight-forward. When multiple sets are present, however, one directed tree per set is needed and these, in general, can impose conflicting orientations to the edges. This is a major additional difficulty in solving the Steiner forest problem. Magnanti and Raghavan [25] show how to consolidate the conflicts with the improved flow formulation.

The issues with conflicting orientations can be avoided altogether by using strong undirected formulations. Goemans [13], Lucena [24], as well as Margot, Prodon and Liebling [26] independently propose an ILP formulation for the Steiner *tree* problem that builds on Edmond's complete description of the tree polytope [11]. This tree-based formulation has a straight-forward extension to the Steiner *forest* problem, but its LP bounds are identical to the ones from the directed formulations.

The literature for the Steiner *tree* problem is more extensive: Several surveys compare ILP formulations and their polyhedral properties [7, 8, 14, 27, 28]. They are the basis for B&B algorithms [6, 22]. Exact FPT algorithms identify parameters that make the problem difficult to solve [4, 9, 12, 19]. Similarly, preprocessing techniques reduce the size of Steiner tree instances by removing trivial parts [10, 27, 28]. While the Steiner tree B&B algorithms imply B&B algorithms for the general Steiner forest case, the Steiner *forest* problem was mostly studied in the context of approximation algorithms [1, 15, 17, 18, 20]. A PTAS on planar and bounded treewidth graphs exists [3].

Notation. Throughout, let $G = (V, E)$ be an undirected, simple graph and let $A = \{(i, j), (j, i) \mid \{i, j\} \in E\}$ be the arcs of the bidirection of G . A *cut-set* in G is a subset $S \subseteq V$. Any cut-set $S \subseteq V$ induces a *cut* $\delta(S) := \{\{i, j\} \in E \mid |\{i, j\} \cap S| = 1\}$. We abbreviate $\delta(i) := \delta(\{i\})$ if $S = \{i\}$. If $D = (V, A)$ is a directed graph, we distinguish the *outgoing cut* $\delta^+(S) = \{(i, j) \in A \mid i \in S \text{ and } j \notin S\}$ and the *incoming cut* $\delta^-(S) = \{(i, j) \in A \mid i \notin S \text{ and } j \in S\}$. Given a vector $x \in X^d$, $d \in \mathbb{Z}_{\geq 0}$, and an index set $I \subseteq \{1, \dots, d\}$ we write $x(I)$ to abbreviate $\sum_{i \in I} x_i$. Finally, for $k \in \mathbb{Z}_{\geq 1}$, let $[k] := \{1, \dots, k\}$.

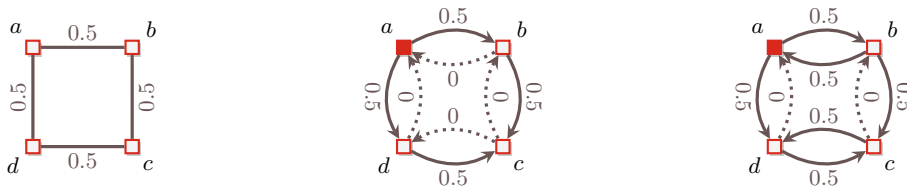
The Steiner forest problem. Consider an undirected graph $G = (V, E)$ together with $K \in \mathbb{N}$ terminal sets $T^1, \dots, T^K \subseteq V$. A *feasible* Steiner forest is a forest $(V_F \subseteq V, E_F \subseteq E)$ in G that, for all $k \in [K]$, contains an s - t -path for all $s, t \in T^k$. A feasible forest (V_F, E_F) is optimum with respect to edge weights $c \in \mathbb{R}_{\geq 0}^{|E|}$ if it minimizes the total cost $\sum_{e \in E_F} c_e$. Assume without loss of generality that the terminal sets are pairwise disjoint: If T^k and T^ℓ share at least one node, then any forest is feasible for T^1, \dots, T^K if and only if it is feasible for the instance where T^k and T^ℓ are replaced by $T^k \cup T^\ell$. We denote the set of all terminal nodes by $\mathfrak{T} := T^1 \cup \dots \cup T^K$ and write $\tau(t) := k$ if $t \in T^k$. For each terminal set T^k , $k \in [K]$, we select an arbitrary node $r^k \in T^k$ as a fixed root node and define $\mathfrak{R} := \{r^1, \dots, r^K\}$. A cut-set $S \subseteq V$ is relevant for the terminal set T^k if it separates r^k from some terminal $t \in T^k$, i.e., if $r^k \in S$ but $t \notin S$ for some $t \in T^k$. We write \mathfrak{S}^k for the set of all cut-sets that are relevant for T^k and $\mathfrak{S} := \mathfrak{S}^1 \cup \dots \cup \mathfrak{S}^K$ for the set of all relevant cut-sets. If $P := \{(x, y) \in \mathbb{R}^{n_1+n_2} \mid Ax + By = d\}$ is a polyhedron let $\text{Proj}_x(P) := \{x \in \mathbb{R}^{n_1} \mid \exists y \in \mathbb{R}^{n_2} : (x, y) \in P\}$ be the projection of P onto the x variables.

2 Eliminating cycles from the linear programming relaxation

Let us briefly review the existing branch-and-bound algorithms and how they model the Steiner forest problem as an ILP. A forest F in $G = (V, E)$ is feasible if and only if any relevant cut-set $S \subset V$ contains at least one edge of F , i.e. if $|\delta_F(S)| \geq 1$ for all $S \in \mathfrak{S}$. Thus, since $c \geq 0$, the undirected cut formulation

$$\begin{aligned} \min \{ & c^T x \mid x \in \mathfrak{L}^{uc} \text{ and integer} \} \text{ where} & \text{(IPuc)} \\ \mathfrak{L}^{uc} := & \{x \in [0, 1]^E \mid x(\delta(S)) \geq 1 \quad \forall S \in \mathfrak{S}\} & (1) \end{aligned}$$

is a valid ILP formulation. While it can be solved efficiently, it yields weak bounds even on trivial instances (see Figure 1). The reason for the weak bounds becomes apparent when we see formulation (IPuc) as a set cover problem: We look for a choice of edges such that each cut $\delta(S)$ in G is covered by at least one edge. Consider any cycle C of length s in G . Any set cover needs $s - 1$ edges to cover C . On the other hand, we obtain a fractional solution of value $\frac{s}{2}$ by setting $x_e = 0.5$ for all edges $e \in C$. Figure 2a shows an example.



(a) A feasible solution for (1). The edges $\{c, d\}$ and $\{b, c\}$ cover the cuts $\delta(\{a, b, c\})$ and $\delta(\{a, b, d\})$. (b) An *infeasible* solution for (2a)–(2d). The arcs (d, c) and (b, c) cover the cut $\delta(\{a, b, d\})$, but not the cut $\delta(\{a, b, c\})$. (c) A feasible solution for (2a)–(2d). Additional capacity is needed on (c, d) and (b, a) to cover all relevant cuts.

■ **Figure 2** An unit cost example where \mathcal{L}^{dc} yields a stronger LP bound than \mathcal{L}^{uc} . The instance has a single terminal set that contains all four nodes of the graph. Node a has been chosen as the root.

The formulation can be improved with a standard construction [7, 2]. Recall that we choose $r^k \in T^k$ as an arbitrary root node of set T^k and consider the bi-directed graph underlying G . For all $k \in [K]$, we now look for an arborescence (a directed tree) rooted at r^k . If any cut-set S is relevant for T^k , then at least one arc must leave S :

$$\min\{c^T x \mid (x, y) \in \mathcal{L}^{dc} \text{ and integer}\} \text{ where} \tag{IPdc}$$

$$\mathcal{L}^{dc} := \left\{ (x, y) \mid y^k(\delta^+(S)) \geq 1 \quad \forall k \in [K], \forall S \in \mathfrak{S}^k \right. \tag{2a}$$

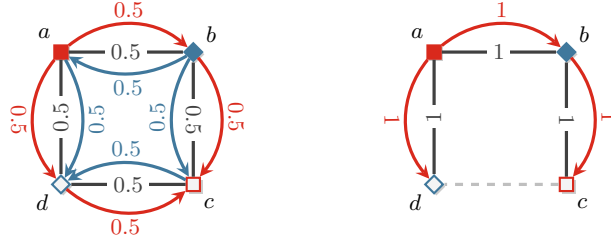
$$\left. y_{ij}^k + y_{ji}^k \leq x_{ij} \quad \forall \{i, j\} \in E, \forall k \in [K] \right. \tag{2b}$$

$$\left. y_{ij}^k, y_{ji}^k \in [0, 1] \quad \forall \{i, j\} \in E, \forall k \in [K] \right. \tag{2c}$$

$$\left. x_{ij} \in [0, 1] \quad \forall \{i, j\} \in E \right\}. \tag{2d}$$

Since any solution (x, y) of (IPdc) can be turned into a feasible Steiner forest $F := \{\{i, j\} \in E \mid \exists k : y_{ij}^k + y_{ji}^k \geq 1\}$ and any feasible Steiner forest can be turned into a solution to (IPdc), this strengthened formulation indeed captures the Steiner forest problem. The formulation eliminates directed cycles from the basic optima of its LP relaxation and indeed the bound of the relaxation coincides with the integer optimum on instance A from Figure 1. However, a slightly modified instance makes the problem reappear, see instance B in Figure 1 or Figure 3: While the support of any y^k is free of directed cycles, the *union* of the supports is not. This is the reason why the formulation works exceptionally well for the Steiner *tree* problem where $K = 1$. If $K > 1$, however, the LP relaxation of (IPdc) is again weak. Still, for practical purposes no better formulation was known prior to this work. The offending cycles potentially appear whenever two terminal sets T^k and T^ℓ – and thus their roots r^k and r^ℓ – end up in the same connected component of the solution, i.e. of the support of x . If we knew beforehand that T^k and T^ℓ lie in the same connected component of an optimum solution, we could simplify the instance, replacing T^k and T^ℓ by their union $T^k \cup T^\ell$. Iterating this idea would yield a solution where all the arborescences are disjoint and the offending cycles are eliminated.

Unfortunately, we cannot know the connected components of a Steiner forest a priori. Instead, Magnanti and Raghavan [25] – we denote their model by (IPmr) and the LP relaxation by \mathcal{L}^{mr} – propose to compute the connected components of a solution on-the-fly in the ILP formulation. Then, whenever T^k and T^ℓ , $k \leq \ell$, lie in the same connected component, they look for a common arborescence that is rooted at r^k and connects all terminals in $T^k \cup T^\ell$. Unfortunately, their formulation has a size of $\Omega(\prod_{k=1}^K \sum_{\ell=k}^K |T^\ell|)$, i.e. it is exponential in the number of terminal sets K . We shall see in the next section how we achieve the same effect with a much smaller ILP formulation.



■ **Figure 3** Detailed picture of instance (B) from Figure 1. *On the left:* The red and blue arcs form a solution for relaxation (2a)–(2d) for the red (□) and blue (◇) terminal set. The gray edges show the values of the x variables. Looking for a Steiner arborescence for each terminal set does not cut off a fractional optimum of cost 2. *On the right:* A solution that roots different terminal sets at the root node of the red (□) terminal set. The fractional optimum is cut off.

3 An new ILP formulation for the Steiner forest problem

Our extended formulation makes use of three kinds of variables. As before, we use a variable x_{ij} for all edges $\{i, j\} \in E$ to determine if $\{i, j\}$ is included in the forest F and two corresponding directed variables y_{ij}, y_{ji} . Likewise, the variables y_{ij}^k and y_{ji}^k for each $k \in [K]$ and each $\{i, j\} \in E$ determine if the arcs (i, j) and (j, i) , respectively, are included in the arborescence rooted at r^k . Finally, we introduce an additional variable $z_{k\ell}$ for each $k \in [K]$ and each $\ell \geq k$, with the interpretation that $z_{k\ell} = 1$ iff T^k and T^ℓ both lie in the arborescence spanned by y^k . In the latter case, we say that r^k is responsible for the terminals in T^ℓ . To make it easier to state the formulation, we define $\mathfrak{T}^{i \dots j}$ as $T^i \cup \dots \cup T^j$ and let $\mathfrak{T}_r^{i \dots j} := \mathfrak{T}^{i \dots j} \setminus \{r^i\}$ be the same set without the i th root node (all other root nodes are still included). In particular, the set $\mathfrak{T}_r^{\ell \dots K}$ contains all the terminal nodes that can potentially be connected to r^ℓ . We extend our previous notion and say that a cut-set $S \subseteq V$ is relevant for r^k and T^ℓ if $r^k \in S$ and some terminal $t \in T^\ell$ is not in S . The set of all cut-sets that are relevant for r^k and T^ℓ is written by \mathfrak{S}_ℓ^k in the sequel. Then, our formulation reads:

$$\min \left\{ c^T x \mid (x, y, z) \in \mathfrak{L}^{sedc} \text{ and integer} \right\} \text{ where} \quad (\text{IPsedc})$$

$$\mathfrak{L}^{sedc} := \left\{ (x, y, z) \mid y^k(\delta^+(S)) \geq z_{k\ell} \quad \forall k \in [K], \ell \geq k, \forall S \in \mathfrak{S}_\ell^k \right\} \quad (3a)$$

$$\sum_{\ell=1}^k z_{\ell k} = 1 \quad \forall k \in [K] \quad (3b)$$

$$y_{ij} \geq \sum_{k \in [K]} y_{ij}^k, \quad y_{ji} \geq \sum_{k \in [K]} y_{ji}^k \quad \forall \{i, j\} \in E \quad (3c)$$

$$z_{kk} \geq z_{k\ell} \quad \forall k \in [K] \setminus \{1, K\}, \forall \ell \geq k+1 \quad (3d)$$

$$y_{ij} + y_{ji} \leq x_{ij} \quad \forall \{i, j\} \in E \quad (3e)$$

$$y(\delta^-(v)) \leq 1 \quad \forall v \in V \quad (3f)$$

$$y^k(\delta^-(t)) = 0 \quad \forall k \in [K] \setminus \{1\}, \forall t \in \mathfrak{T}^{1 \dots k-1} \quad (3g)$$

$$y_{ij}^k, y_{ji}^k \in [0, 1] \quad \forall \{i, j\} \in E, \forall k \in [K] \quad (3h)$$

$$x_{ij}, y_{ij}, y_{ji} \in [0, 1] \quad \forall \{i, j\} \in E \quad (3i)$$

$$z_{k\ell} \in [0, 1] \quad \forall k \in [K], \forall \ell \geq k \}. \quad (3j)$$

For any k, ℓ , the left hand side of the directed cut-set constraint (3a) is non-negative and the constraint is trivially satisfied if $z_{k\ell} = 0$. If otherwise $z_{k\ell} = 1$, we need to connect all terminals from T^ℓ to the k -th root r^k . Then, any cut-set S separating r^k from some terminal in T^ℓ must have at least one outgoing edge. This is exactly the condition modeled by (3a). For each $k \in [K]$, the constraints (3b) ensure that exactly one root r^ℓ is responsible for T^k (and r^1 is always responsible for T^1 , i.e., $z_{11} = 1$). We use constraints (3c) to enforce that each edge $\{i, j\}$ is part of at most one arborescence. We also want to make sure that no “transitive” responsibilities exist: If r^k is responsible for T^ℓ , then r^ℓ cannot be responsible for some T^m , $m \neq \ell$. This is modeled by the symmetry breaking constraints (3d). They make sure that if root r^k is responsible for some terminal set T^ℓ , then r^k must be responsible for T^k as well. The capacity constraints (3e) say that if an edge $\{i, j\}$ is used in any arborescence, then it must be included in the tree. Moreover, no node in any arborescence should have more than one incoming arc, as modeled by the indegree constraints (3f). Finally, the terminals in $T^{1\dots k-1}$ cannot be attached to root r^k and thus, no arc of the corresponding arborescence should enter such a terminal, see constraint (3g).

To solve \mathfrak{L}^{sedc} efficiently, we only add a subset of the cut-set constraints (3a) at the beginning. Then, given a solution (x^*, y^*, z^*) to a partially generated \mathfrak{L}^{sedc} , we can find a relevant S , a k and an ℓ such that $y^k(\delta^+(S)) < z_{k\ell}$ (or decide that none exist) efficiently: For each $k \in [K]$, each $\ell \geq k$ and each $t \in T^\ell$, we compute a minimum r^k - t -cut. If for any k, ℓ such a cut $\delta(S)$ has a value of strictly less than $z_{k\ell}$, then (x^*, y^*, z^*) does not satisfy the cut-set constraint corresponding to S and we add it to our LP and iterate. Otherwise, all cuts $S \in \mathfrak{S}_\ell^k$ must have a value of at least $z_{k\ell}$ and $(x^*, y^*, z^*) \in \mathfrak{L}^{sedc}$.

► **Lemma 1.** *Formulation (IPsedc) models the Steiner forest problem correctly. Its LP relaxation \mathfrak{L}^{sedc} can be solved in time polynomial in the size of G and K .*

Strength of the new formulation. How can we compare two LP relaxations \mathfrak{L}^A and \mathfrak{L}^B ? We cannot expect that the bound from \mathfrak{L}^A is stronger than the bound from \mathfrak{L}^B on all instances: Generally, the optima of both LPs will be integral on *some* instances and must coincide then. We can, however, ask that the bound obtained from \mathfrak{L}^A is never worse than the bound obtained from \mathfrak{L}^B . This is the case if any solution to \mathfrak{L}^A is feasible for \mathfrak{L}^B as well, i.e. if $\mathfrak{L}^A \subseteq \mathfrak{L}^B$. We say that \mathfrak{L}^A is *strictly stronger* than \mathfrak{L}^B if additionally at least one solution of \mathfrak{L}^B is infeasible for \mathfrak{L}^A , i.e. if $\mathfrak{L}^A \subsetneq \mathfrak{L}^B$. In general, some truncation or extension of the solution might be necessary if \mathfrak{L}^A and \mathfrak{L}^B live in different variable spaces, but we can project the solutions suitably.

Instead of comparing the models directly, we compare their equivalent flow-based models; replacing the cut-condition by a flow-balance constraint. We also introduce additional flow variables f . Any feasible solution to \mathfrak{L}^{sedf} defines a flow $f^{k,t}$ from r^k to any terminal $t \in \mathfrak{T}^{k\dots K}$ and ensures that the flow value of $f^{k,t}$ is exactly $z_{k\ell}$.

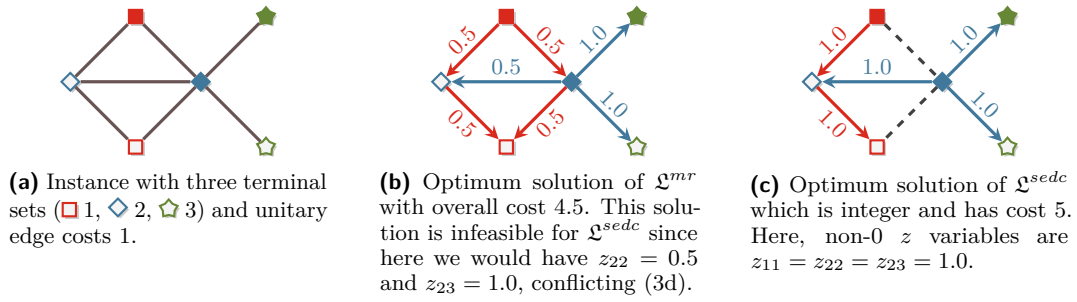
$$\mathfrak{L}^{sedf} := \left\{ (x, y, f, z) \mid \begin{array}{l} f_{ij}^{kt} \leq y_{ij}^k, f_{ji}^{kt} \leq y_{ji}^k \\ \forall k \in [K], \forall \{i, j\} \in E \\ \forall t \in \mathfrak{T}_r^{k\dots K} \end{array} \right. \quad (4a)$$

$$\begin{array}{l} f^{kt}(\delta^+(i)) - f^{kt}(\delta^-(i)) = \sigma_{kt}(i)z_{k\tau(t)} \\ \forall i \in V, \forall k \in [K] \\ \forall t \in \mathfrak{T}_r^{k\dots K} \end{array} \quad (4b)$$

$$f^{kt}(\delta^+(t)) = 0 \quad \forall k \in [K], \forall t \in \mathfrak{T}_r^{k\dots K} \quad (4c)$$

$$(3b)–(3j) \quad (4d)$$

$$\left. \begin{array}{l} f_{ij}^{kt}, f_{ji}^{kt} \in [0, 1] \\ \forall k \in [K], \forall t \in \mathfrak{T}_r^{k\dots K} \\ \forall \{i, j\} \in E \end{array} \right\} \quad (4e)$$



■ **Figure 4** Example instance where \mathfrak{L}^{sedc} gives a stronger bound than \mathfrak{L}^{mr} .

The constant $\sigma_{kt}(i)$ is set to 1 if $i = r^k$, to -1 if $i = t$, and to 0 otherwise. The constraints (4c) prohibit f^{kt} from leaving t and facilitate the comparison to \mathfrak{L}^{mr} . Analogously, the relaxation \mathfrak{L}^{dc} has an arc-flow-based equivalent \mathfrak{L}^{df} that forces a choice of arcs such that each root r^k is able to send one unit of flow to each terminal in $T^k \setminus \{r^k\}$.

► **Theorem 2.** $\text{Proj}_x(\mathfrak{L}^{sedc}) \subsetneq \text{Proj}_x(\mathfrak{L}^{dc})$

Proof sketch. We prove the claim by showing that \mathfrak{L}^{sedf} is strictly stronger than \mathfrak{L}^{df} . Let thus $(x, y, f, z) \in \mathfrak{L}^{sedf}$. We want to show that there exists some y' and some flow f' such that $(x, y', f') \in \mathfrak{L}^{df}$. Here, the challenge is that there might be a non-zero flow $f^{k,t}$ from r^k to $t \in T^\ell$ whereas in \mathfrak{L}^{df} , all the flow to t must originate from r^ℓ . Still, we can morally obtain a feasible flow f' in the following way: Since $r^\ell \in T^\ell$, there must be a flow of value exactly $z_{k\ell}$ from r^k to r^ℓ . But r^k also sends a flow with value $z_{k\ell}$ to t . Thus, if we reverse f^{k,r^ℓ} we can concatenate it with $f^{k,t}$ and maintain flow conservation. We remove all cycles and we obtain the desired flow f' . However, this construction might force us to change the orientation of some of the arcs, which poses an additional technical difficulty. Finally, we can iterate this argument and combine all flows $f^{m,t}$ from any root r^m to t . By constraint (3b), these flows must add up to one. Strictness follows from instance (B) in Figure 1. ◀

Our second theoretical result is that the new relaxation \mathfrak{L}^{sedc} is strictly stronger than the relaxation of [25]. Due to space restrictions we refer the reader to [25] for the description of \mathfrak{L}^{mr} with constraints (14b)–(14j).

► **Theorem 3.** $\text{Proj}_x(\mathfrak{L}^{sedc}) \subsetneq \text{Proj}_x(\mathfrak{L}^{mr})$

Proof sketch. As before, we compare \mathfrak{L}^{sedf} instead of \mathfrak{L}^{sedc} . The major difference between \mathfrak{L}^{sedf} and \mathfrak{L}^{mr} is this: While in \mathfrak{L}^{sedf} , any two flows f^{kt} and $f^{kt'}$ for $t, t' \in T^\ell$ must have the same flow value $z_{k\ell}$, the same flows can have different values in \mathfrak{L}^{mr} . In that sense, \mathfrak{L}^{sedf} is more restricted and it makes sense that any flow that is feasible in \mathfrak{L}^{sedf} is feasible in \mathfrak{L}^{mr} , too, whereas the converse is not necessarily true (see Figure 4). More formally, let $(\bar{x}, \bar{y}, \bar{z}, \bar{f}) \in \mathfrak{L}^{sedf}$. We argue that $(\bar{x}, \bar{y}, \bar{f}) \in \mathfrak{L}^{mr}$. It follows from (4b) that $(\bar{x}, \bar{y}, \bar{f})$ satisfies (14b) from [25]. Constraint (14c) follows from (4b), (4c), and (3b). For (14d), apply (4b), (4b) with $t = \bar{\ell}$, and (4c). Constraint (14e) follows from (4a), (3c), and (3e). Likewise, constraint (14f) follows from (4a), (3c), and applying (3f). Finally, the constraint (14g) is implied by (3g). (14h) is equivalent to (4c). ◀

3.1 A smaller cut-based formulation

We remark that (IPsedc) can be written in the slightly different form below. While the reformulation is smaller and less involved, it turns out that its linear programming bounds are potentially weaker than the ones from (IPsedc). We need two variables y_{ij}, y_{ji} , and a variable x_{ij} for each edge $\{i, j\} \in E$. As before, for all $k \in [K]$ and all $\ell \geq k$, we have a decision variable $z_{k\ell}$ that tells us whether the terminals in T^ℓ should be connected to the root r^k .

$$\min \left\{ c^T x \mid (x, y, z) \in \mathfrak{L}^{edc} \text{ and integer} \right\} \text{ where} \quad (\text{IPedc})$$

$$\mathfrak{L}^{edc} := \left\{ (x, y, z) \mid y(\delta^+(S)) \geq \sum_{\substack{k \leq \ell: \\ r^k \in S}} z_{k\ell} \quad \forall \ell \in [K], \forall S \subseteq V: T^\ell \cap S \neq T^\ell \right\} \quad (5a)$$

$$\sum_{\ell=1}^k z_{\ell k} = 1 \quad \forall k \in [K] \quad (5b)$$

$$z_{kk} \geq z_{k\ell} \quad \forall k \in [K] \setminus \{1, K\}, \forall \ell \geq k+1 \quad (5c)$$

$$y_{ij} + y_{ji} \leq x_{ij} \quad \forall \{i, j\} \in E \quad (5d)$$

$$y_{ij}, y_{ji}, x_{ij} \in [0, 1] \quad \forall \{i, j\} \in E \quad (5e)$$

$$z_{k\ell} \in [0, 1] \quad \forall k \in [K], \forall \ell \geq k \quad (5f)$$

To see why the formulation is correct, consider a cut-set $S \subseteq V$ with $t \notin S$ for some terminal $t \in T^\ell$. If S contains a root node r^k with $z_{k\ell} = 1$, then S must have at least one outgoing arc and the right-hand side of (5a) evaluates to 1 (observe that because of (5b) the right-hand side can never exceed 1). Otherwise, the right-hand side of (5a) evaluates to 0 and the constraint is trivially satisfied. The LP relaxation of (IPedc) can be solved in polynomial time using a similar algorithm as for \mathfrak{L}^{sedc} .

► **Lemma 4.** $\text{Proj}_x(\mathfrak{L}^{sedc}) \subsetneq \text{Proj}_x(\mathfrak{L}^{edc})$.

Proof. Let $(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathfrak{L}^{sedc}$. We argue that $(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathfrak{L}^{edc}$. The constraints (5b)–(5d) are trivially satisfied. Now, consider a directed cut $S \subseteq V: S \cap T^\ell \neq \emptyset$, for some set $\ell \in [K]$. Any cut S is relevant to the sum in the right-hand side of constraint (5a) if and only if it is a valid cut for constraint (3a), hence

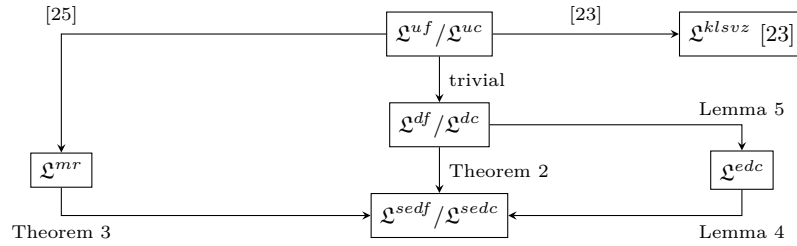
$$\tilde{y}(\delta^+(S)) \stackrel{(3c)}{\geq} \sum_{k=1}^K \tilde{y}^k(\delta^+(S)) \geq \sum_{k=1}^{\ell} \tilde{y}^k(\delta^+(S)) \stackrel{(3a)}{\geq} \sum_{k \leq \ell} \tilde{z}_{k\ell} \geq \sum_{\substack{k \leq \ell: \\ r^k \in S}} \tilde{z}_{k\ell}$$

and thus (5a) is satisfied. Again, strictness follows from instance (B) in Figure 1. ◀

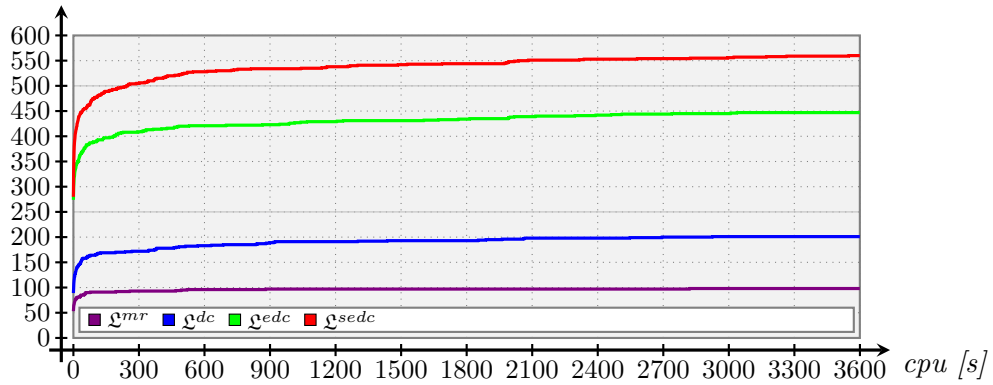
On the other hand, the model is stronger than the directed model without z variables.

► **Lemma 5.** $\text{Proj}_x(\mathfrak{L}^{edc}) \subsetneq \text{Proj}_x(\mathfrak{L}^{dc})$.

We summarize the results of the discussion in Figure 5 and remark that the relationship of \mathfrak{L}^{mr} to the models \mathfrak{L}^{dc} and \mathfrak{L}^{edc} is an open problem. Our conjecture is that it holds $\text{Proj}_x(\mathfrak{L}^{mr}) \subsetneq \text{Proj}_x(\mathfrak{L}^{edc}) \subsetneq \text{Proj}_x(\mathfrak{L}^{dc})$.



■ **Figure 5** Relationship of the LP relaxations. The arrows point to the stronger relaxation.



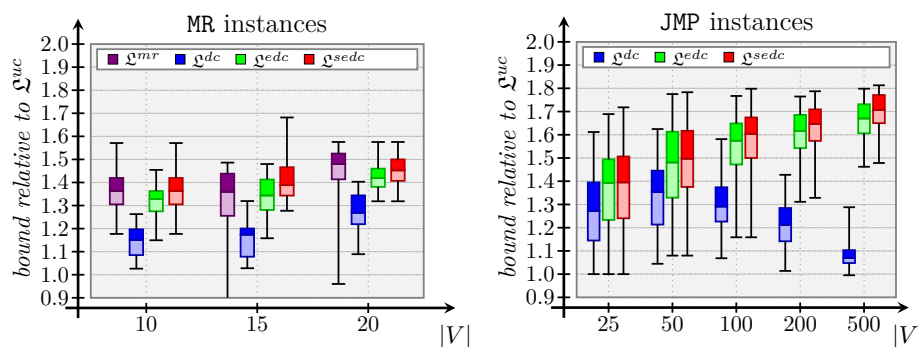
■ **Figure 6** Number of JMP instances (out of 580) solved by B&B after x seconds.

4 Experimental results

Settings. All experiments were performed on a Debian 9.4 machine with an Intel(R) Xeon(R) CPU E5-2643 running at 3.30GHz. Our code is written in C++ using the ILOG CPLEX 12.6.3 framework. We compiled with gcc-6.3 and -O2 flags. Automatic symmetry breaking and presolving was disabled in CPLEX, as well as all general integer cuts.

Instances. For the JMP instance set, we generated 580 random graphs with a frequently used method by Johnson, Minkoff, and Philipps [21]: First, distribute n nodes uniformly at random in a unit square. Then, insert an edge $\{i, j\}$ if the Euclidean distance between i and j is less than α/\sqrt{n} , where α is a parameter for the random generator. The cost of the edge $\{i, j\}$ is proportional to the Euclidean distance. Finally, connect all nodes with a minimum Euclidean spanning tree to ensure that the instance is connected.

To determine K random terminal sets, we first select $t \cdot |V|$ nodes uniformly at random (the number $K \in [n/2]$ of terminal sets and the terminal percentage $t \in [0, 1]$ are again parameters). We then bring the selected nodes into a random order and draw $K - 1$ distinct split points from $\{2, \dots, t \cdot |V| - 1\}$, thus splitting the random node order into K distinct terminal sets. For each $n \in \{25, 50, 150, 200, 500\}$, we choose a small, a medium, and a large number of terminal sets K . The percentage t of terminal nodes is picked from $\{0.25, 0.5, 0.75, 1.0\}$ unless a combination of n, K , and t results in a terminal set size of less than two. For each choice of n, K , and t , we generate five instances with $\alpha = 1.6$ and five instances with $\alpha = 2.0$; leading to 580 JMP instances. The MR instance set is generated based on [25] and contains 85 instances.



■ **Figure 7** Improvement of the linear programming bound over the bound obtained from \mathcal{L}^{uc} : The plot shows the ratio of the best bound after 1200 seconds over the optimum bound from \mathcal{L}^{uc} . The theoretical maximum improvement is at most 2. *On the right:* For $|V| \in \{25, 50\}$ all bounds are optimum; for $|V| \in \{100, 200\}$ only the \mathcal{L}^{edc} and \mathcal{L}^{sedc} bounds are optimum. For $|V| = 500$, about 50% of the \mathcal{L}^{edc} and \mathcal{L}^{sedc} and none of the \mathcal{L}^{dc} bounds are optimum.

The branch-and-bound algorithm. We solve formulation (IPsedc) by an B&B algorithm. As the algorithm requires solving the LP relaxation \mathcal{L}^{sedc} in each B&B node, we generate the LP relaxation dynamically with the separation procedure from Section 3. This allows us to efficiently solve each B&B node. We follow the same approach for the ILP from [25], for (IPedc), and for (IPdc) and compare the results. Similar separation procedures using minimum cuts exist for \mathcal{L}^{dc} and \mathcal{L}^{edc} , so that we can generate the relaxations efficiently as well. In more detail, we compute a minimum s - t -cut by computing a maximum s - t -flow f and then deriving a cut-set S , where a node $v \in V$ is included in S if and only if there is a directed path from v to t in the residual network of f . We can then derive a cut-set inequality based on S . Some algorithmic techniques have the potential to improve this on-the-fly generation [22]:

Back cuts. Additionally add the cut-set inequality corresponding to \bar{S} where $v \in V$ is included in \bar{S} if and only if there is a directed s - v -path in the residual network of f .

Nested cuts. Assign an infinite capacity to all saturated edges in the residual network of f and iterate. Nested cuts can be combined with back cuts: We first compute S and \bar{S} and then compute nested cuts on both sets.

Creep flows. Add a small $\varepsilon = 10^{-8}$ to all capacities. This lets us find a minimum weight cut that cuts few edges. The creep flow variant works together with both nested cuts and back cuts.

Cut purging. Finally, it can be beneficial to remove cut-set inequalities from the relaxation if they have not been binding for a number of iterations.

It is not clear a priori which combination of these variants leads to the best performance of the algorithm. In a preliminary experiment, we evaluated all 16 combinations for all the formulations under consideration. To avoid overfitting, we tested on a random subset of the instances only. Back cuts were beneficial in all cases. The \mathcal{L}^{sedc} relaxation benefited from additional creep flows, while \mathcal{L}^{dc} worked best with additional nested cuts and purging. In all cases, we compute the maximum s - t -flows with a custom implementation of the push-relabel algorithm with the *highest-label* strategy and the *gap heuristic* [16, 5]. Since the \mathcal{L}^{sedc} constraints (3f) and (3g) would be valid for \mathcal{L}^{dc} and \mathcal{L}^{edc} as well, we compare against \mathcal{L}^{sedc} without (3f) and (3g). This results in a fairer comparison.

Comparison of the algorithms. We compare B&B algorithms based on the previous best formulations with our new ones in Figure 6 using the JMP instance set. All algorithms are run in the tuned configuration from the preliminary experiment. The figure shows that our new \mathcal{L}^{sedc} based algorithm solves almost twice as many instances to optimality as the previous ones. A more detailed picture would show that all unsolved instances are large ones with $|V| = 500$. The \mathcal{L}^{edc} based algorithm solves significantly less instances, but still performs better than the algorithm based on \mathcal{L}^{dc} . The algorithm based on \mathcal{L}^{mr} mostly solved the small instances. The \mathcal{L}^{sedc} , the \mathcal{L}^{edc} , the \mathcal{L}^{dc} , and the \mathcal{L}^{mr} based algorithm solved 480, 385, 185, and 97 instances without branching, respectively. In the following, we analyze why our algorithms perform well.

The new bounds can be computed quickly. We compare our new approach against the two previous best: The relaxation \mathcal{L}^{dc} of the directed cut formulation and the relaxation \mathcal{L}^{mr} of [25]. Ideally, we would like to have relaxations that solve quickly and yield a strong bound. Indeed, the LP relaxations in our new algorithm can be solved to optimality quickly and reliably: We find the LP optimum of at least 500 out of 580 instances within 1200 seconds. Moreover, the bulk of the LP relaxations is solved within 100 seconds for \mathcal{L}^{edc} and within 400 seconds for \mathcal{L}^{sedc} . At the same time, the existing B&B algorithms struggle to solve their LP relaxations: The relaxation \mathcal{L}^{mr} could only be solved to optimality within 1200 seconds in 100 out of 580 times. In the same time frame, the relaxation \mathcal{L}^{dc} could be solved 280 times.

The new bounds are strong. The results from the previous section show that the optimum bound from the LP relaxations of the advanced formulations will never be worse than the bound from \mathcal{L}^{uc} . We would like to quantify the ratio of the bounds; however, the theoretical worst-case ratio of the bounds is 1. What ratio can we hope for on non-artificial instances? To answer this question, we solve the LP relaxations of the advanced formulations. Since any feasible solution to an LP relaxation yields a valid bound, we stop the computation after 1200 seconds and take the best bound obtained up to that point. We then compare this bound with the optimum of \mathcal{L}^{uc} in Figure 7. The bounds obtained from the relaxation of [23] were exactly the same as of \mathcal{L}^{uc} and are not shown here. \mathcal{L}^{mr} only solved a significant number of the small instances from the JMP set. To nonetheless obtain a fair comparison for \mathcal{L}^{mr} , we instead look at the MR instance set that is based on the original publication of [25]. The comparison can also be seen in Figure 7. We see that *if* \mathcal{L}^{mr} can be solved, it yields a bound that is comparable to the one from the new relaxations.

5 Conclusion

Overall, our new branch-and-bound algorithm works very well and its performance seems to be due to the strong bounds obtained from the new ILP formulation (IPsedc). While its relaxation \mathcal{L}^{sedc} is solved less quickly than the simplified relaxation \mathcal{L}^{edc} , its stronger bounds seem to pay off overall. At the same time, it answers Magnanti's and Raghavan's open problem: There is indeed an equally strong cut-based model to [25]. On the theoretical side, we would like to obtain an LP relaxation with an integrality gap of much less than 2. This problem is not solved by \mathcal{L}^{sedc} : We observe that it coincides with \mathcal{L}^{dc} if $K = 1$. On the other hand, Könemann et al. [23] propose an LP relaxation that has a better worst-case integrality gap. In our experiments, however, the relaxation always yields the same bounds as the weak undirected cut relaxation \mathcal{L}^{uc} , making it less suitable for practical purposes.

References

- 1 A. Agrawal, P. 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, 1995.
- 2 A. Balakrishnan, T. L. Magnanti, and R. T. Wong. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5):716–740, 1989.
- 3 M. Bateni, M. T. Hajiaghayi, and D. Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM*, 58(5):21:1–21:37, 2011.
- 4 A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: Fast subset convolution. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 67–74. ACM, 2007.
- 5 B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.
- 6 S. Chopra, E. R. Gorres, and M. R. Rao. Solving the Steiner tree problem on a graph using branch and cut. *ORSA Journal on Computing*, 4(3):320–335, 1992.
- 7 S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(1):209–229, 1994.
- 8 S. Chopra and M. R. Rao. The Steiner tree problem II: Properties and classes of facets. *Mathematical Programming*, 64(1-3):231–246, 1994.
- 9 S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.
- 10 C. W. Duin and A. Volgenant. Reduction tests for the steiner problem in graph. *Networks*, 19(5):549–567, 2006.
- 11 J. Edmonds. Submodular functions, matroids, and certain polyhedra. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization — Eureka, You Shrink!*, number 2570 in LNCS, pages 11–26. Springer Berlin Heidelberg, 2003.
- 12 R. E. Erickson, C. L. Monma, and A. F. Veinott. Send-and-split method for minimum-concave-cost network flows. *Mathematics of Operations Research*, 12(4):634–664, 1987.
- 13 M. X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63(1-3):157–182, 1994.
- 14 M. X. Goemans and Y.-S. Myung. A catalog of Steiner tree formulations. *Networks*, 23(1):19–28, 1993.
- 15 M. X. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
- 16 A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, 1988.
- 17 M. Groß, A. Gupta, A. Kumar, J. Matuschke, D. R. Schmidt, M. Schmidt, and J. Verschae. A local-search algorithm for Steiner forest. In A. R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
- 18 A. Gupta and A. Kumar. Greedy Algorithms for Steiner Forest. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 871–878. ACM, 2015.
- 19 S. Hougardy, J. Silvanus, and J. Vygen. Dijkstra meets Steiner: A fast exact goal-oriented Steiner tree algorithm. *Mathematical Programming Computation*, 9(2):135–202, 2017.
- 20 K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

- 21 D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: Theory and practice. In *Proceedings of the Symposium on Discrete Algorithms, SODA '00*, pages 760–769. SIAM, 2000.
- 22 T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.
- 23 J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam. A group-strategyproof cost sharing mechanism for the Steiner forest game. *SIAM Journal on Computing*, 37(5):1319–1341, 2008.
- 24 A. Lucena. Tight bounds for the Steiner problem in graphs. Technical report, RC for Process Systems Engineering, Imperial College, London, 1993.
- 25 T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45:61–79, 2005.
- 26 F. Margot, A. Prodon, and T. M. Liebling. Tree polytope on 2-trees. *Mathematical Programming*, 63(1–3):183–191, 1994.
- 27 T. Polzin. *Algorithms for the Steiner Problem in networks*. PhD thesis, Universität des Saarlandes, 2004. URL: <http://scidok.sulb.uni-saarland.de/volltexte/2004/218/index.html>.
- 28 T. Polzin and V. S. Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112(1):241–261, 2001.