I N F O R M A T I K

Improving Linear Programming Approaches for the Steiner Tree Problem

Ernst Althaus Tobias Polzin
Siavash Vahdati Daneshmand

MPI–I–2003–1–004 February 2003

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Authors' Addresses

Ernst Althaus, Tobias Polzin
Max-Planck-Institut für Informatik,
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
email: `{althaus,polzin}@mpi-sb.mpg.de`

Siavash Vahdati Daneshmand
Theoretische Informatik, Universität Mannheim,
68131 Mannheim, Germany
email: `vahdati@informatik.uni-mannheim.de`

Abstract

We present two theoretically interesting and empirically successful techniques for improving the linear programming approaches, namely graph transformation and local cuts, in the context of the Steiner problem. We show the impact of these techniques on the solution of the largest benchmark instances ever solved.

Keywords

Steiner problem; Lower Bounds; Linear Programming; Local Cuts

1 Introduction

In combinatorial optimization many algorithms are based (explicitly or implicitly) on linear programming approaches. A typical application of linear programming to optimization problems works as follows: First, the combinatorial problem is reformulated as an integer linear program. Then, some integrality constraints are relaxed and one of the numerous methods for solving (or approximating) a linear program is applied. For $\mathcal{NP}$-hard optimization problems, any linear relaxation of polynomial size (and any polynomial time solvable relaxation) is bound to have an integrality gap (unless $\mathcal{P} = \mathcal{NP}$). So the quality of the underlying relaxation can have a decisive impact on the performance of the overall algorithm. As a consequence, methods for generating tight lower bounds are significant contributions to elaborated algorithms for combinatorial optimization problems, see for example the long history of research for the Traveling Salesman Problem (TSP) focusing on linear programming [2, 3, 13, 14].

In this work, we improve the linear programming based techniques for the Steiner tree problem in networks, which is the problem of connecting a given subset of the vertices of a weighted graph at minimum cost. It is a classical $\mathcal{NP}$-hard problem [11] with many important applications in network design in general and VLSI design in particular. For background information on this problem, see [5, 10].

For the Steiner problem, linear programming approaches are particularly important, since the best known practical algorithms for optimal solutions, for heuristic Steiner trees, and for preprocessing techniques, which reduce the size of the problem instance without changing an optimal solution, all make frequent use of linear programming techniques [16, 17, 18, 19]. Typical situations where linear programming is used are the computation of lower bounds in the context of an exact algorithm, bound-based reduction techniques [16], and partitioning-based reduction techniques [17]. Especially for large and complex problem instances, very small differences in the integrality gap can cause an enormous additional computational effort in the context of an exact algorithm. Therefore, methods for improving the quality of the lower bounds are very important.

In Section 2, we give some definitions, including the directed cut relaxation, which is the basis for many linear programming approaches for the Steiner problem. Then, we will present two approaches for improving the lower bound provided by this relaxation:

- In Section 3, we introduce the "vertex splitting" technique: We identify locations in the network that contribute to the integrality gap and split up the decisive vertices in these locations. Thereby, we transform the problem instance into one that is equivalent with respect to the integral solution, but the solution of the relaxation may improve.

  This idea is inspired by the column replacement techniques that were introduced by Balas and Padberg [4] and generalized by Haus et. al. [9] and Gentile et. al. [8]. In these and other papers a general technique for solving integer programs is developed. However, these techniques are mainly viewed as primal algorithms, and extensions for combinatorial optimization problems are presented for the Stable Set problem only. Furthermore, these extensions are not yet part of a practical algorithm (the general integer programming techniques have been applied successfully). Thus, we are the first to apply this basic idea in a practical algorithm for a concrete combinatorial optimization problem.

- In Section 4, we show how to adopt the "local cuts" approach, introduced by Applegate, Bixby, Chvátal, and Cook [3] in the context of the TSP: Additional constraints are generated using projection, lifting and optimal solutions of subinstances of the problem. To apply this approach to the Steiner problem, we develop new shrinking operations and separation techniques.

In Section 5, we embed these two approaches into our successful algorithm for solving Steiner tree problems and present some experimental results. Like many other elaborated optimization packages our program consists of many parts (the source code has approximately 30000 instructions, not including the LP-solver code). Thus, this paper describes only a small part of the whole program. However, among other results, we will show that this part is decisive for the solution of the problem instance d15112, which is to our knowledge the largest benchmark Steiner tree instance ever solved. Furthermore, we believe that these new techniques are also interesting for other combinatorial optimization problems.

The other parts of the program package are described in a series of papers [16, 17, 18, 19]. Note that there is no overlapping between these papers and the work presented here.

2 Definitions

The Steiner problem in networks can be stated as follows (see [10] for details): Given an (undirected, connected) network $G = (V, E, c)$ (with vertices $V = \{v_1, \ldots, v_n\}$, edges E and edge weights $c_e > 0$ for all $e \in E$) and a set R, $\emptyset \neq R \subseteq V$, of *required vertices* (or *terminals*), find a minimum weight tree in G that spans R (a *Steiner minimal tree*). If we want to stress that v_i is a terminal, we will write z_i instead of v_i. We also look at a reformulation of this problem using the (bi-)directed version of the graph, because it yields stronger relaxations: Given $G = (V, E, c)$ and R, find a minimum weight arborescence in $\vec{G} = (V, A, c)$ ($A := \{[v_i, v_j], [v_j, v_i] \mid (v_i, v_j) \in E\}$, c defined accordingly) with a terminal (say z_1) as the root that spans $R^{z_1} := R \setminus \{z_1\}$.

A **cut** in $\vec{G} = (V, A, c)$ (or in $G = (V, E, c)$) is defined as a partition $C = \{\overline{W}, W\}$ of V ($\emptyset \subset W \subset V; V = W \dot{\cup} \overline{W}$). We use $\delta^-(W)$ to denote the set of arcs $[v_i, v_j] \in A$ with $v_i \in \overline{W}$ and $v_j \in W$. For simplicity, we write $\delta^-(v_i)$ instead of $\delta^-(\{v_i\})$. The sets $\delta^+(W)$ and, for the undirected version, $\delta(W)$ are defined similarly.

In the integer programming formulations we use (binary) variables $x_{[v_i,v_j]}$ for each arc $[v_i, v_j] \in A$, indicating whether this arc is in the solution ($x_{[v_i,v_j]} = 1$) or not ($x_{[v_i,v_j]} = 0$). For any $B \subseteq A$, $x(B)$ is short for $\sum_{a \in B} x_a$.

For every integer program P, LP denotes the linear relaxation of P, and $v(LP)$ denotes the value of an optimal solution for LP.

Other definitions can be found in [7, 10].

2.1 The Directed Cut Formulation

The directed cut formulation P_C was stated in [24]. An undirected version was already introduced in [1], but the directed variant yields a stronger relaxation.

$$\begin{aligned} c \cdot x &\rightarrow \min, \\ x(\delta^-(W)) &\geq 1 \quad (z_1 \notin W, R \cap W \neq \emptyset), \qquad (1)\\ x &\in \{0,1\}^{|A|}. \qquad (2) \end{aligned}$$

The constraints (1) are called Steiner cut constraints. They guarantee that in any arc set corresponding to a feasible solution, there is a path from z_1 to any other terminal.

There is a group of constraints (see for example [12]) that can make LP_C stronger. We call them flow-balance constraints:

$$x(\delta^-(v_i)) \leq x(\delta^+(v_i)) \qquad (v_i \in V \setminus R). \qquad (3)$$

We denote the linear program that consists of LP_C and (3) by LP_{C+FB}. In [15] we gave a comprehensive overview on relaxations for the Steiner tree problem.

3 Graph Transformation: Vertex Splitting

In this section, we describe a new technique for effectively improving the lower bound corresponding to the directed cut relaxation by manipulating the underlying network.

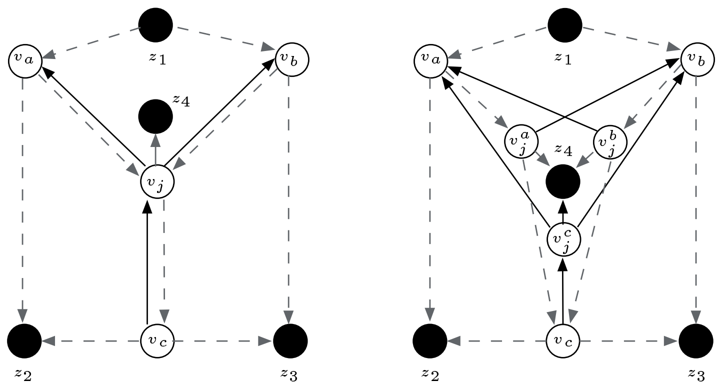

Figure 1: Splitting of vertex v_j. The filled circles are terminals, z_1 is the root, all arcs have cost 1. An optimal Steiner arborescence has value 6 in each network. In the left network $v(LP_{C+FB})$ is 5.5 (set the x-values of the dashed arcs to 0.5 and of $[v_j, z_4]$ to 1), but 6 in the right network (again, set the x-values of the dashed arcs and of $[v_j^a, z_4]$ and $[v_j^b, z_4]$ to 0.5). The difference is that in the left network, there is a situation that is called "rejoining of flows": Flows from z_1 to z_2 and from z_1 to z_3 enter v_j on different arcs, but leave on the same arc, so they are accounted in the x variables only once. Before splitting, the x-value corresponding to the arc $[v_j, v_c]$ is 0.5, after splitting the corresponding x-values sum up to 1.

We use the property that in an optimal directed Steiner tree, each vertex has in-degree at most 1. Implicitly, we realize a case distinction: If an arc $[v_i, v_j]$ is in an optimal Steiner tree, we know that other arcs in $\delta^-(v_j)$ cannot be in the tree. The only necessary operation to realize this case distinction for the Steiner problem is the splitting of a vertex. A vertex v_j is replaced by several vertices v_j^i, one for each arc $[v_i, v_j]$ entering v_j. Each new vertex v_j^i has only one incoming arc $[v_i, v_j^i]$, and essentially the same outgoing arcs as v_j. In Figure 1, the splitting of vertex v_j is depicted. The explanation of the figure also provides some intuition how splitting can be useful. In Section 3.3 we describe how we identify candidates for splitting.

The splitting operation is described formally by the pseudocode below. We maintain an array *orig* that points for each vertex in the transformed network to the vertex in the original network that it derives from. Initially, $orig[v_j] = v_j$ for all $v_j \in V$. With $P(v_i)$ we denote the longest common suffix of all paths from z_1 to v_i after every path is translated back to the original network. The intuition behind this definition is that if v_i is in an optimal Steiner arborescence, $P(v_i)$ must also be in the arborescence after it is translated into the original network. Note that the path $P(v_i)$ consists of vertices in the original network and may contain cycles; in this case, v_i cannot be part of an optimal arborescence. In Figure 1, $P(v_a)$ consists of v_a and $P(v_j^a)$ is the path of length 1 from v_a to v_j. To compute $P(v_i)$, one can reverse all arcs and use breadth-first-search. The main purpose of using $P(v_i)$ is to avoid inserting unnecessary arcs. This can improve the value of and the computation times for the lower bound. It is also necessary for the proof of termination in Section 3.2.

For the ease of presentation, we assume that the root terminal z_1 has no incoming arcs, and that all other terminals have no outgoing arcs. If this is not the case, we simply add copies of the terminals and connect them with appropriate zero cost arcs to the old terminals.

SPLIT-VERTEX$(G, v_j, orig)$: (assuming $v_j \notin R$)
forall $[v_i, v_j] \in \delta^-(v_j)$:
  if $P(v_i)$ *contains a cycle* **or** $orig[v_j]$ *in* $P(v_i)$:
    continue *with next arc in* $\delta^-(v_j)$
  insert a new vertex v_j^i *into* G, $orig[v_j^i] := orig[v_j]$
  insert an arc $[v_i, v_j^i]$ *with cost* $c(v_i, v_j)$ *into* G
  forall $[v_j, v_k] \in \delta^+(v_j)$:
    if $orig[v_k]$ *not in* $P(v_i)$:
      insert an arc $[v_j^i, v_k]$ *with cost* $c(v_j, v_k)$ *into* G
delete v_j
delete all vertices that are not reachable from z_1

3.1 Correctness

In this section, we prove that the transformation is valid, i.e., it does not change the value of an optimal Steiner arborescence.

Lemma 1 *Any optimal Steiner arborescence with root z_1 in the original network can be transformed into a feasible Steiner arborescence with root z_1 in the transformed network with the same cost and vice versa.*

Proof We consider one splitting operation on vertex $v_j \in V \setminus R$, transforming a network G into G'. Repeating the argumentation extends the result to multiple splits. We use a condition (†) for a tree T denoting that for every v_k, v_l in T, it holds: $orig[v_k] = orig[v_l] \Leftrightarrow v_k = v_l$. Note that condition (†) holds for an optimal Steiner arborescence in the original network.

Let T be an optimal Steiner arborescence with root z_1 for G satisfying (†). If $v_j \notin T$, T is part of G' and we are done. If $v_j \in T$, there is exactly one arc $[v_i, v_j] \in T$. When $[v_i, v_j]$ is considered in the splitting, $P(v_i)$ is a subpath of the path from z_1 to v_i in T after it is translated to the original network. Together with (†) follows that neither $orig[v_j]$, nor $orig[v_k]$ for any $[v_j, v_k] \in T$ is in $P(v_i)$. Therefore, all arcs $[v_j, v_k] \in T$ can be replaced by arcs $[v_j^i, v_k]$ and the arc $[v_i, v_j]$ can be replaced by $[v_i, v_j^i]$. The transformed T is part of G', connects all terminals, has the same cost as T and satisfies condition (†).

Now, let T' be an optimal Steiner arborescence for G'. Obviously, T' can be transformed into a feasible solution T with no higher cost for G.

3.2 Termination

In this section, we show that iterating the splitting operation will terminate.

Lemma 2 *For all non-terminals v_j, $P(v_j)$ is the common suffix of all paths $P(v_i)$ appended by $orig[v_j]$ for all $v_i, [v_i, v_j] \in \delta^-(v_j)$.*

Proof As the Line 10 of *SPLIT-VERTEX* guarantees that there is always a path from z_1 to v_j, the claim follows directly from the definition of $P(v_j)$.

Lemma 3 *For any two non-terminals v_s and $v_t, v_s \neq v_t$, $P(v_s)$ is not a suffix of $P(v_t)$.*

Proof Assume the lemma is not true. We choose two vertices v_s and $v_t, v_s \neq v_t, P(v_s)$ is a suffix of $P(v_t)$ such that the length of $P(v_s)$ is minimal. Obviously, $orig[v_s] = orig[v_t]$. Thus, v_s and v_t were inserted in some splits. After these splits, v_s and v_t have in-degree 1. Only splitting a vertex v'_s with $[v'_s, v_s] \in \delta^-(v_s)$ can increase the in-degree of v_s, but $orig[v'_s]$ is the same for all $[v'_s, v_s] \in \delta^-(v_s)$. Together with Lemma 2 for $P(v_s)$ follows that $P(v_s)$ contains at least two vertices. As it is a suffix of $P(v_t)$, this also holds for $P(v_t)$. For any two vertices v'_s, v'_t with $[v'_s, v_s] \in \delta^-(v_s)$ and $[v'_t, v_t] \in \delta^-(v_t)$ it holds that $v'_s \neq v'_t$, $P(v'_s)$ is a suffix of $P(v'_t)$ and it is shorter than $P(v_s)$, a contradiction.

Lemma 4 *After splitting a vertex v_j with in-degree greater than 1, for any newly inserted vertex v_j^i it holds that $P(v_j^i)$ is longer than $P(v_j)$ was before the split.*

Proof Assume that there is a newly inserted vertex v_j^a such that $P(v_j^a)$ is not longer than $P(v_j)$. From Lemma 2 for $P(v_j^a)$ and $P(v_j)$ follows that $P(v_j^a) = P(v_a)$ appended by $orig[v_j]$ and that $P(v_j)$ is a suffix of $P(v_j^a)$. Together with the assumption follows $P(v_j) = P(v_j^a)$. As v_j had in-degree greater than 1 before the split, we know that there was a vertex $v_b, v_b \neq v_a, [v_b, v_j] \in \delta^-(v_j)$. From Lemma 3 follows that $P(v_a)$ was not a suffix of $P(v_b)$. Thus, the common suffix of $P(v_a)$ and $P(v_b)$ did not contain $P(v_a)$. Using Lemma 2 for $P(v_j)$, it follows that $P(v_j)$ did not contain $P(v_a)$, a contradiction to $P(v_j) = P(v_j^a)$.

Lemma 5 *Repeated splitting of vertices with in-degree greater than 1 will stop with a network in which all non-terminals have in-degree 1. As a consequence, there is exactly one path from z_1 to v_i for all non-terminals v_i.*

Proof As long as there is a non-terminal with in-degree greater than 1, we can split it, which will delete the vertex and possibly replace it by some vertices with in-degree 1. We only have to show that this procedure terminates, as a split may increase the in-degree of other vertices.

If splitting a vertex v_j deletes it without inserting any new vertex, we label v_j as *invalid.*

Now, we examine the changes in the network as an arbitrary vertex v_j with in-degree greater than 1 is split. Let v_m be any non-terminal after the split that was not newly inserted. From the definition of $P(v_m)$ follows that $P(v_m)$ can only change if some vertex or arc is not inserted because of the conditions in lines 2 and 7 of *SPLIT-VERTEX* and some paths from z_1 to v_m do not exist any longer. Since there is sill a path from z_1 to v_m left, $P(v_m)$ can only become longer, it may even visit some vertex twice (i.e., $P(v_m)$ contains a cycle). In the latter case, v_m becomes invalid.

From Lemma 1 follows that a transformed optimal tree will always be contained in the current network, thus after at most $|V|$ splits, there will be a split of a valid vertex. If a split is performed on a valid vertex v_j, at least one new vertex v_j^i will be inserted. From Lemma 4 follows that $P(v_j^i)$ is longer than $P(v_j)$ was before the split. But as $P(v_j^i)$ does not contain a cycle (Line 2 of *SPLIT-VERTEX*), its length is bounded by the number of vertices in the original network. Thus, the procedure terminates.

3.3 Implementation Issues

Of course, for a practical application one does not want to split all vertices, which could blow up the network exponentially. In a cutting plane algorithm one first adds violated Steiner cut or flow-balance constraints. They can be found by min-cut computations [16], respectively with a summation of the incoming and outgoing arcs variables of non-terminals. If no such constraint can be found, we search for good candidates for the splitting procedure, i.e., vertices where more than one incoming arc and at least one outgoing arc have an x-value greater than zero. After splitting these vertices, the modified network will be used for the computation of new constraints, using the same algorithms as before. To represent this transformation in the linear program, we add new variables for the newly added arcs, and additional constraints that the x-values for all newly added arcs corresponding to an original arc $[v_i, v_j]$ must sum up to $x_{[v_i,v_j]}$. Using this procedure the constraints calculated for the original network can still be used.

4 Project, Separate, and Lift: Local Cuts

Let $S = (G, R) = (V, E, c, R)$ be an instance of the Steiner problem. Let $\mathcal{ST}(S)$ be the set of all incidence vectors of Steiner trees of S and $\mathcal{SG}(S) = \mathcal{ST}(S) + \mathbb{R}_+^{|E|}$. We call the elements of $\mathcal{SG}(S)$ the Steiner graphs of S. We consider Steiner graphs, since Steiner graphs are invariant under the shrink operation (defined in Section 4.1). Note that

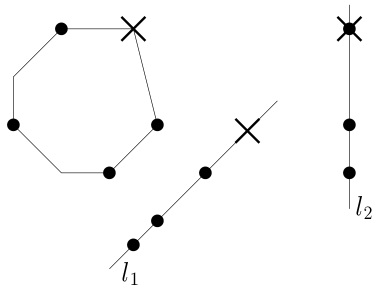

Figure 2: The feasible integer solutions are marked as dots, the fractional solution to separate by the cross. If we project the solutions to the line l_1, we can obtain a valid violated inequality and lift it back to the original space. If we project to the line l_2, the fractional solution falls into the convex hull of the integer solutions and no such inequality can be found.

the values $x_{(v_i,v_j)}$ are not restricted to be integral or bounded. It is obvious that if the objective function is non-negative, there exists a minimum Steiner graph that is a Steiner tree. Thus all vertices of the polyhedron $\mathrm{conv}(\mathcal{SG}(S))$ are Steiner trees. Furthermore, $\mathrm{conv}(\mathcal{SG}(S))$ is full dimensional if G is connected.

From a high level view, local cuts can be described as follows. Assume we want to separate x^* from $\mathrm{conv}(\mathcal{SG}(S))$. Using a linear mapping ϕ, we project the given point x^* into a small-dimensional vector $\phi(x^*)$ and solve the separation problem over $\mathrm{conv}(\phi(\mathcal{SG}(S)))$. If we can find a violated inequality $a \cdot \tilde{x} \geq b$ that separates $\phi(x^*)$ from $\mathrm{conv}(\phi(\mathcal{SG}(S)))$, we know that the linear inequality $a \cdot \phi(x) \geq b$ separates x^* from $\mathrm{conv}(\mathcal{SG}(S))$. The method is illustrated in Figure 2.

To make this method work, we have to choose ϕ such that

1. there is a good chance that $\phi(x^*) \notin \mathrm{conv}(\phi(\mathcal{SG}(S)))$ if $x^* \notin \mathrm{conv}(\mathcal{SG}(S))$,
2. we can solve the separation problem over $\mathrm{conv}(\phi(\mathcal{SG}(S)))$ efficiently and
3. the inequalities $a \cdot \phi(x) \geq b$ are strong.

We choose ϕ in such a way that for every solution $x \in \mathcal{SG}(S)$ of our Steiner problem instance S, the projected $\phi(x)$ is a Steiner graph of a small Steiner problem instance S^ϕ, i.e., $\mathrm{conv}(\phi(\mathcal{SG}(S))) = \mathrm{conv}(\mathcal{SG}(S^\phi))$ for an instance S^ϕ of the Steiner problem. Since our Steiner tree program package tends to be very efficient for solving small Steiner problem instances, we can handle the separation problem, as we will see in Section 4.2.

We use iterative shrinking to obtain the linear mappings. We review the well-known concept of shrinking in the next section. After that, we introduce our separation algorithm for small Steiner graph instances. So far, we always assumed that we are looking at the undirected version of the Steiner problem, since our separation algorithm is much faster for this variant. As seen above, the directed cut relaxation is stronger than the undirected variant. In Section 4.3, we discuss how we can use the directed formulation without solving directed Steiner graph instances in the separation algorithm.

4.1 Shrinking

We define our linear mappings as an iterative application of the following simple, well-known mapping, called shrinking. For the Steiner problem, shrinking was indroduced by Chopra and Rao [6].

Shrinking means to replace two vertices v_a and v_b by a new vertex $\langle v_a, v_b\rangle$ and replace edges (v_i, v_a) and (v_i, v_b) by an edge $(v_i, \langle v_a, v_b\rangle)$ with value $x^*_{(v_i,v_a)} + x^*_{(v_i,v_b)}$ (We assume $x^*_{(v_i,v_j)} = 0$ if $(v_i, v_j) \notin E$). The new vertex $\langle v_a, v_b\rangle$ is in the set of terminals R if v_a or v_b (or both) are in R. This informally defines the mapping ϕ and the instance S^ϕ. Note that for any incidence vector of a Steiner graph for the original problem, the new vector is the incidence vector of a Steiner graph in the reduced problem. Furthermore, for every Steiner graph $\tilde{x}$ in S^ϕ there is a Steiner graph $x \in \mathcal{SG}(S)$ such that $\phi(x) = \tilde{x}$. Thus $\mathrm{conv}(\phi(\mathcal{SG}(S))) = \mathrm{conv}(\mathcal{SG}(S^\phi))$.

Note that if we iteratively shrink a set of vertices $W \subset V$ into one vertex $\langle W\rangle$, the obtained linear mapping is independent of the order in which we apply the shrinks. We denote the unique linear mapping which shrinks a subset $W \subset V$ into one vertex by ϕ^W.

We have developed conditions on x^* under which we can prove that $\phi(x^*)$ is not in the convex hull of $\mathcal{SG}(S^\phi)$ if x^* is not in the convex hull of $\mathcal{SG}(S)$.

Lemma 6 *Let* $x^* \geq 0$.

1. (edge of value 1): *Let* $x^*_{(v_a,v_b)} \geq 1$ *and* $W = \{v_a, v_b\}$. $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow \phi^W(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^W}))$.

2. (non-terminal of degree 2): *Let* v_a *be in* $V \setminus R$ *and the vertices* $(v_1, \dots v_k)$ *in* V *be ordered according to their* $x^*_{(\cdot,v_a)}$ *value (in decreasing order). Furthermore, let* $W = \{v_a, v_1\}$. *If* $x^*_{(v_3,v_a)} = 0$, *then* $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow \phi^W(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^W}))$.

3. (cut of value 1): *Let* W *be such that* $x^*(\delta(W)) = 1$ *and* $\emptyset \neq R \cap W \neq R$. *Let* $\overline{W} = V\setminus W$. $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow \phi^W(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^W})) \wedge \phi^{\overline{W}}(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^{\overline{W}}}))$.

4. (biconnected components): *Let* $U, W \subset V$ *and* $v_a \in V$ *be such that* $U \cup W = V$, $U \cap W = \{v_a\}$ *and* $x^*_{(v_k,v_l)} = 0$ *for all* $v_k \in U \setminus \{v_a\}$ *and* $v_l \in W \setminus \{v_a\}$. *Furthermore, let* $\emptyset \neq R \cap W \neq R$. $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow \phi^U(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^U})) \wedge \phi^W(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^W}))$.

5. (triconnected components): *Let* $U, W \subset V$ *and* $v_a, v_b \in V$ *be such that* $U \cup W = V \setminus \{v_a\}$, $U \cap W = \{v_b\}$ *and* $x^*_{(v_k,v_l)} = 0$ *for all* $v_k \in U \setminus \{v_b\}$ *and* $v_l \in W \setminus \{v_b\}$. *Let furthermore* $x^*(\delta(v_a)) = 1$ *and* $v_a, v_b \in R$. $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow \phi^U(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^U})) \wedge \phi^W(x^*) \in \mathrm{conv}(\mathcal{SG}(S^{\phi^W}))$.

Proof We already argued that if $x^* \in \mathrm{conv}(\mathcal{SG}(S))$ then $\phi(x^*) \in \mathrm{conv}(\mathcal{SG}(S^\phi))$ for every linear mapping obtained by iterative shrinking independent of x^*. Thus we only have to show the reverse direction of the claims, i.e., if $\phi(x^*) \in \mathrm{conv}(\mathcal{SG}(S^\phi))$ (for the last three claims, if both projections are in the convex hull) then $x^* \in \mathrm{conv}(\mathcal{SG}(S))$.

It suffices to prove the claims for the case that x^* is rational.

1. We can find a large integer N and, for $1 \leq i \leq N$, incidence vectors of Steiner trees $\tilde{t}^i$ in S^{ϕ^W} such that $N\phi^W(x^*) \geq \sum_{1\leq i\leq N} \tilde{t}^i$.

   The idea is as follows: We will create Steiner trees $t^{i,j}$ out of $\tilde{t}^i$ by including the edge (v_a, v_b) in every tree and for every edge $(v_k, \langle W\rangle)$ in $\tilde{t}^i$ we use either the edge (v_k, v_a) or (v_k, v_b). The number of Steiner trees in which we use a specific edge (v_k, v_a) or (v_k, v_b) is determined by the ratio between $x^*_{(v_k,v_a)}$ and $x^*_{(v_k,v_b)}$.

   Let M be a large integer such that $Mx^*_{(v_k,v_l)}/\phi^W(x^*)_{(v_k,\langle W\rangle)}$ is integral for every $v_k \in V \setminus W$ and $v_l \in W$. We know that $\phi^W(x^*)_{(\langle W\rangle,v_k)} = x^*_{(v_a,v_k)} + x^*_{(v_b,v_k)}$ for all $v_k \in V \setminus W$. For every $\tilde{t}^i$ and $1 \leq j \leq M$ we define $t^{i,j}$ with

   - $t^{i,j}_{(v_a,v_b)} = 1$,
   - $t^{i,j}_{(v_k,v_l)} = \tilde{t}^i_{(v_k,v_l)}$ for $v_k, v_l \in V \setminus \{v_a, v_b\}$,
   - for $v_k \in V \setminus \{v_a, v_b\}$ we make a case distinction:
     If $j \leq Mx^*_{(v_a,v_k)}/\phi^W(x^*)_{(\langle W\rangle,v_k)}$: $t^{i,j}_{(v_a,v_k)} = \tilde{t}^i_{(\langle W\rangle,v_k)}, t^{i,j}_{(v_b,v_k)} = 0$,
     otherwise: $t^{i,j}_{(v_a,v_k)} = 0, t^{i,j}_{(v_b,v_k)} = \tilde{t}^i_{(\langle W\rangle,v_k)}$.

   As $t^{i,j}_{(v_a,v_k)} + t^{i,j}_{(v_b,v_k)} = \tilde{t}^i_{(\langle W\rangle,v_k)}$, it can be verified that $NMx^* \geq \sum_{1\leq i\leq N} \sum_{1\leq j\leq M} t^{i,j}$.

   It also follows that if $\tilde{t}^i$ contained a path from a vertex v_k to $v_{\langle W\rangle}$, each $t^{i,j}$ contains a path from v_k to v_a and to v_b. As a consequence, each pair of terminals is connected in $t^{i,j}$.

2. We can find a large integer N and, for $1 \leq i \leq N$, incidence vectors of Steiner trees $\tilde{t}^i$ in S^{ϕ^W}, such that $N\phi^W(x^*) \geq \sum_{1\leq i\leq N} \tilde{t}^i$.

   The idea is as follows: We only need to consider the case that $\langle W\rangle$ is used in a tree $\tilde{t}^i$. Since there are at most two edges with positive x^*-values adjacent to v_a, we can replace all edges in the tree $\tilde{t}^i$ adjacent to $\langle W\rangle$ (except $(v_2, \langle W\rangle)$) by edges adjacent to v_1. Further, we have to take care of the edge $(v_2, \langle W\rangle)$, if it is in $\tilde{t}^i$. In this case, we create trees $t^{i,j}$ using either the edge (v_2, v_1) or the two edges (v_2, v_a) and (v_a, v_1). Again the number of trees in which we use the two alternatives is given by the ratio of $x^*_{(v_2,v_1)}$ and $x^*_{(v_2,v_a)}$.

   Let M be a large integer such that $Mx^*_{(v_2,v_l)}/\phi^W(x^*)_{(v_2,\langle W\rangle)}$ is integral for $v_l \in \{v_1, v_a\}$. For every $\tilde{t}^i$ and $1 \leq j \leq M$ we define $t^{i,j}$ with

   - $t^{i,j}_{(v_k,v_l)} = \tilde{t}^i_{(v_k,v_l)}$ for every $v_k, v_l \in V \setminus \{v_a, v_1\}$,
   - $t^{i,j}_{(v_k,v_1)} = \tilde{t}^i_{(v_k,\langle W\rangle)}$ for $v_k \in V \setminus \{v_2\}$,
   - $t^{i,j}_{(v_k,v_a)} = 0$ for $v_k \in V \setminus \{v_2\}$,
   - If $j \leq Mx^*_{(v_1,v_2)}/\phi^W(x^*)_{(\langle W\rangle,v_2)}$: $t^{i,j}_{(v_2,v_1)} = \tilde{t}^i_{(v_2,\langle W\rangle)}$, $t^{i,j}_{(v_2,v_a)} = 0$,
     otherwise: $t^{i,j}_{(v_2,v_1)} = 0$, $t^{i,j}_{(v_2,v_a)} = \tilde{t}^i_{(v_2,\langle W\rangle)}$,
   - $t^{i,j}_{(v_a,v_1)} = t^{i,j}_{(v_2,v_a)}$.

As $x^*_{(v_a,v_1)} \geq x^*_{(v_a,v_2)}$, it can be verified that $NMx^* \geq \sum_{1\leq i\leq N}\sum_{1\leq j\leq M} t^{i,j}$.

If there is an edge $(v_k, \langle W\rangle)$ in $\tilde{t}^i$, then in each $t^{i,j}$ there is either the edge (v_k, v_1) or (in the case that $k = 2$ and j is large enough) the two edges (v_k, v_a) and (v_1, v_a). Thus $t^{i,j}$ is a Steiner tree.

3. We can find a large integer N and, for $1 \leq i \leq N$, Steiner trees t^i in S^{ϕ^W} and Steiner trees $\bar{t}^i$ in $S^{\phi^{\overline{W}}}$ such that $N\phi^W(x^*) \geq \sum_{1\leq i\leq N} t^i$ and $N\phi^{\overline{W}}(x^*) \geq \sum_{1\leq i\leq N} \bar{t}^i$.

   Since $x^*(\delta(W)) = 1$, it follows that in each tree t^i there is exactly one edge in $\delta(\langle W\rangle)$ and in each tree $\bar{t}^i$ there is exactly one edge in $\delta(\langle\overline{W}\rangle)$. For each edge $(v_k, \langle W\rangle), v_k \in \overline{W}$, there are $N\phi^W(x^*)_{(v_k,\langle W\rangle)}$ trees t^i containing this edge. We assign each such tree t^i to one edge $(v_k, v_l), v_l \in W$ such that there are $Nx^*_{(v_k,v_l)}$ trees assigned to this edge. This is possible because $\phi^W(x^*)_{(v_k,\langle W\rangle)} = \sum_{v_l\in W} x^*_{(v_k,v_l)}$. We do the same for all trees $\bar{t}^i$. Now, we join the trees $t^a\setminus\{(v_k, \langle W\rangle)\}$ and $\bar{t}^b\setminus\{(v_l, \langle\overline{W}\rangle)\}$ by an edge (v_k, v_l) to a new tree $\hat{t}^i$ if they are assigned to this edge.

   It can be verified that $Nx^* \geq \sum_{1\leq i\leq N} \hat{t}^i$.

   It remains to show that there is a path between each pair of terminals z_1, z_2 in each tree $\hat{t}^i$, originating from t^a and $\bar{t}^b$, both assigned to an edge (v_k, v_l). If $z_1, z_2 \in \overline{W}$, they were connected in t^a and as t^a contained only one edge $(v_k, \langle W\rangle)$, they are still connected in $\hat{t}^i$. The case $z_1, z_2 \in W$ is similar. For $z_1 \in \overline{W}, z_2 \in W$, we can use the path between z_1 and $\langle W\rangle$ in t^a, the edge (v_k, v_l) and the path between $\langle\overline{W}\rangle$ and z_2 in $\bar{t}^b$.

4. We can find a large integer N and, for $1 \leq i \leq N$, Steiner trees t^i in S^{ϕ^W} and Steiner trees s^i in S^{ϕ^U} such that $N\phi^W(x^*) \geq \sum_{1\leq i\leq N} t^i$ and $N\phi^U(x^*) \geq \sum_{1\leq i\leq N} s^i$.

   We join the trees t^i with $\langle W\rangle$ replaced by v_a and s^i with $\langle U\rangle$ replaced by v_a to a new tree $\hat{t}^i$.

   It can be verified that $Nx^* \geq \sum_{1\leq i\leq N} \hat{t}^i$.

   Since $\hat{t}^i$ contains the complete Steiner trees t^i and s^i with the respective shrunken vertex replaced by v_a, we know that $\hat{t}^i$ is a Steiner tree.

5. We can find a large integer N and, for $1 \leq i \leq N$, Steiner trees t^i in S^{ϕ^W} and Steiner trees s^i in S^{ϕ^U} such that $N\phi^W(x^*) \geq \sum_{1\leq i\leq N} t^i$ and $N\phi^U(x^*) \geq \sum_{1\leq i\leq N} s^i$.

   Note that $\phi^W(x^*)(\delta(v_a)) = 1$ and thus every t^i has exactly one edge adjacent to v_a. Thus there are $i' = N - N\phi^W(x^*)_{(v_a,\langle W\rangle)}$ Steiner trees t^i that do not use the edge $(v_a, \langle W\rangle)$. Let these be the Steiner trees 1 to i'. Analogously there are $i'' = N - N\phi^U(x^*)_{(v_a,\langle U\rangle)}$ Steiner trees s^i that do not use the edge $(v_a, \langle U\rangle)$. Let these be the Steiner trees $i' + 1$ to $i' + i''$. First, we replace $\langle W\rangle$ and $\langle U\rangle$ by v_b in all t^i and s^i.

   For $i \leq i'$ we join t^i and the subgraph $s^i \setminus \{(v_a, \langle U\rangle)\}$ to $\hat{t}^i$.

   For $i' < i \leq i''$ we join the subgraph $t^i \setminus \{(v_a, \langle W\rangle)\}$ and s^i to $\hat{t}^i$.

   Finally, for $i > i' + i''$ we join the subgraph $t^i \setminus \{(v_a, \langle W\rangle)\}$, the subgraph $s^i \setminus \{(v_a, \langle U\rangle)\}$, and the edge (v_a, v_b) to $\hat{t}^i$.

As $\sum_{1\le i\le N} \hat{t}^i_{(v_a,v_b)} = N(\phi^W(x^*)_{(v_a,\langle W\rangle)} + \phi^U(x^*)_{(v_a,\langle U\rangle)} - 1) = N(x^*(\delta(v_a)) + x^*_{(v_a,v_b)} - 1) = Nx^*_{(v_a,v_b)}$, it can be verified that $Nx^* \ge \sum_{1\le i\le N} \hat{t}^i$.

Finally, we show that all $\hat{t}^i$ are Steiner trees. If $i \le i'$, $\hat{t}^i$ contains the complete Steiner tree t^i with $\langle W\rangle$ replaced by v_b. Thus all terminals in $U \cup \{v_a\}$ are connected. Since s^i contains $(v_a, \langle U\rangle)$ and two subtrees connecting each terminal in $W \setminus \{v_b\}$ either to v_a or to v_b and since v_a and v_b are connected in t^i, we know that $\hat{t}^i$ is a Steiner tree. A similar argument holds for $i' < i \le i' + i''$. For $i > i' + i''$, we know that v_a and v_b are connected directly by the edge (v_a, v_b) and every other terminal is connected either to v_a or to v_b. Thus $\hat{t}^i$ is a Steiner tree.

Applying these "exact" shrinks does not project the solution of the current linear program into the projected convex hull of all integer solutions, i.e., if the solution of the current linear program has not reached the value of the integer optimum, we can find a valid, violated constraint in the shrunken graphs. Unfortunately, in many cases the graphs are still too large after applying these shrinks and we have to apply some "heuristic" shrinks afterwards.

In the implementation, we use a parameter *max-component-size*, which is initially 15. If the number of vertices in a graph after applying all "exact" shrinks is not higher than *max-component-size*, we start FIND-FACET (see Section 4.2), otherwise, we start a breadth-first-search from different starting positions, shrink everything except the first *max-component-size* vertices visited by the BFS, try the "exact" shrinks again and start FIND-FACET. If it turns out that we could not find a valid, violated constraint, we increase *max-component-size*. We also tried other "heuristic" shrinks by relaxing "exact" shrinks, e.g., accepting minimum Steiner cuts with value above 1, or edges that have an x-value close to 1. But we could not come up with a definitive conclusion which shrinks are best, and we believe that there is still room for improvement.

As we will see in the next section, our separation algorithm finds a facet of $\mathrm{conv}(\mathcal{SG}(S^\phi))$. As shown in Theorem 4.1 of [6], the lifted inequality is then a facet of $\mathrm{conv}(\mathcal{SG}(S))$.

4.2 Separation: Finding Facets

Assume we want to separate x^* from $\mathrm{conv}(\mathcal{SG}(S))$. Note that we actually separate $\phi(x^*)$ from $\mathrm{conv}(\mathcal{SG}(S^\phi))$, but this problem can be solved with the same algorithm.

As we will see, the separation problem can be formulated as a linear program with a row for every Steiner graph. Trying to solve this linear program using cutting planes, we have the problem that the number of Steiner graphs (contrary to the case of Steiner trees) is infinite and optimal Steiner graphs need not exist. Note that the same complication arises when applying local cuts to the Traveling Salesman Problem.

The solution for the separation problem is much simpler and more elegant for the Steiner tree case than for the Traveling Salesman case. The key is the following Lemma, a slight variation of Lemma 3.1.2 in [6].

Lemma 7 *All facets of* $\mathrm{conv}(\mathcal{SG}(S))$ *different from* $x_{(v_a,v_b)} \ge 0$ *for an edge* $(v_a, v_b) \in E$ *can be written in the form* $a \cdot x \ge 1$ *with* $a \ge 0$.

Thus, if $x^* \notin \mathrm{conv}(\mathcal{SG}(S))$, we can find an inequality of the form $a \cdot x \geq 1$, $a \geq 0$, that separates x^* from $\mathrm{conv}(\mathcal{SG}(S))$. Note that if $a \geq 0$, there is a Steiner tree $t \in \mathcal{SG}(S)$ minimizing $a \cdot t$.

Thus an exact separation algorithm can be stated as follows (the name arises from the fact that the algorithm will find a facet of $\mathrm{conv}(\mathcal{SG}(S))$, as we will see later).

FIND-FACET $(G = (V, E), R, x^*)$

1    $T :=$ *incidence vector of a Steiner tree for* G, R
2   **repeat:**
3   *solve LP*: $\min x^* \cdot \alpha,\ T\alpha \geq 1,\ \alpha \geq 0$ *(basic solution)*
4   **if** $x^* \cdot \alpha \geq 1$:   **return** *"*$x^* \in \mathrm{conv}(\mathcal{SG}(S))$*"*
5   *find minimum Steiner tree* t *for* $G = (V, E, \alpha), R$
6   **if** $t \cdot \alpha < 1$:   *add* t *as a new row to matrix* T
7   **else**:   **return** $\alpha \cdot x \geq 1$

The algorithm terminates, since there are only a finite number of Steiner trees in $\mathcal{ST}(S)$ and as soon as the minimum Steiner tree t computed in Line 5 is already in T, we terminate because $\alpha \cdot t \geq 1$ is an inequality of the linear program solved in Line 3.

Lemma 8 *If FIND-FACET does not return an inequality,* $x^* \in \mathrm{conv}(\mathcal{SG}(S))$.

Proof Consider the dual of the linear program in Line 3: $\max \sum_i \lambda_i, T^T\lambda \leq x^*$, which has the optimal value $x^* \cdot \alpha \geq 1$. We divide λ by $x^* \cdot \alpha$, with the consequence that $\sum_i \lambda_i = 1$. Now, $T^T\lambda$ is a convex combination of Steiner trees and it still holds $T^T\lambda \leq x^*$.

Lemma 9 *If FIND-FACET returns an inequality* $\alpha \cdot x \geq 1$*, this inequality is a valid, separating, and facet-defining inequality.*

Proof The value of the last computed minimum Steiner tree t is $t \cdot \alpha \geq 1$. Therefore, if $x \in \mathcal{SG}(S)$, the value can only be greater and it holds $x \cdot \alpha \geq t \cdot \alpha \geq 1$.

As $x^* \cdot \alpha < 1$, the inequality is separating.

From the basic solution of the linear program, we can extract $|E|$ linearly independent rows that are satisfied with equality. For each such row of the form $\alpha \cdot t \geq 1$, we add the tree t to a set S_λ and for each row $\alpha_e \geq 0$, we add the edge e to a set S_μ. Note that $|S_\lambda| + |S_\mu| = |E|$ and the incidence vectors corresponding to $S_\lambda \cup S_\mu$ are linearly independent.

There is at least one tree t_j in S_λ. For each edge $e \in S_\mu$ we add to S_λ a new Steiner graph t_k that consists of t_j added by the edge e. Since $\alpha_e = 0$ we know that $\alpha \cdot t_k = 1$. Since the incidence vectors corresponding to $S_\lambda \cup S_\mu$ were linearly independent, replacing e with the t_k yields a new set of linearly independent vectors.

Repeating this procedure yields $|E|$ linearly independent $t_i \in S_\lambda$ with $\alpha \cdot t_i = 1$. Thus, $\alpha \cdot x \geq 1$ is a facet.

As in [3], we can improve the running time of the algorithm by using the following fact. If we know some valid inequalities $a \cdot x \geq b$ with $a \cdot x^* = b$ then $x^* \in \mathrm{conv}(\mathcal{SG}(S)) \Leftrightarrow x^* \in \mathrm{conv}(\mathcal{SG}(S) \cap \{x \in \mathbb{R}^{|E|} \mid a \cdot x = b\})$. Thus we can temporarily remove all edges

(v_i, v_j) with $x^*_{(v_i,v_j)} = 0$, since $x^*_{(v_i,v_j)} \geq 0$ is a valid inequality. Call the resulting instance S'. We use our algorithm to find a facet of $\mathrm{conv}(\mathcal{SG}(S'))$. We can use sequential lifting to obtain a facet of $\mathrm{conv}(\mathcal{SG}(S))$. For details see [3] and Theorem 4.2 of [6].

4.3 Directed versus Undirected Formulations

For computing the lower bounds, we focus on the directed cut formulation, because its relaxation is stronger than the undirected variant. However, in the local cut separation algorithm we want to solve undirected Steiner graph instances, since they can be solved much faster.

The solution is to use another linear mapping that maps arc-values of a bidirected Steiner graph instance $\vec{S} = (V, A, c, R)$ to edge-values of an undirected Steiner graph instance $S = (V, E, c', R)$.

We define S by $E = \{(v_i, v_j) \mid [v_i, v_j] \in A\}$ and $c'_{(v_i,v_j)} = c_{[v_i,v_j]} = c_{[v_j,v_i]}$. For a vector $x \in \mathbb{R}^{|A|}$ we define $\psi(x) \in \mathbb{R}^{|E|}$ by $\psi(x)_{(v_i,v_j)} = x_{[v_i,v_j]} + x_{[v_j,v_i]}$.

Lemma 10 *$x^* \in \mathrm{conv}(\mathcal{SG}(\vec{S})) \Rightarrow \psi(x^*) \in \mathrm{conv}(\mathcal{SG}(S))$.*

$\bar{x} \in \mathrm{conv}(\mathcal{SG}(S)) \Rightarrow \exists x^* \in \mathrm{conv}(\mathcal{SG}(\vec{S}))$ with $\psi(x^*) = \bar{x}$.

If $c \cdot x^*$ is smaller than the cost of an optimal Steiner arborescence, then $\psi(x^*) \notin \mathrm{conv}(\mathcal{SG}(S))$.

Proof Let z_1 be the root in the directed formulation. It suffices to prove the claims for the case that x^* is rational. We show the two claims in turn.

If $x^* \in \mathrm{conv}(\mathcal{SG}(\vec{S}))$, we can find a large integer N and directed Steiner trees $t^i \in \mathcal{ST}(\vec{S})$ such that $Nx^* \geq \sum_{1 \leq i \leq N} t^i$. Clearly $N\psi(x^*) \geq \sum_{1 \leq i \leq N} \psi(t^i)$. Furthermore, $\psi(t^i)$ are Steiner graphs, since each directed path in t^i from the root z_1 to a terminal z_k gives an undirected path between z_1 and z_k in $\psi(t^i)$.

If $\bar{x} \in \mathrm{conv}(\mathcal{SG}(S))$, we can find a large integer N and undirected Steiner trees $t^i \in \mathcal{ST}(S)$ such that $N\bar{x} \geq \sum_{1 \leq i \leq N} t^i$. Let $\tilde{t}^i$ be the directed tree obtained by rooting t^i at z_1. Clearly $\tilde{t}^i$ is a directed Steiner tree and $\psi(\tilde{t}^i) = t^i$. Let $x' = N^{-1} \sum_{1 \leq i \leq N} \tilde{t}^i$. We know that $x' \in \mathrm{conv}(\mathcal{SG}(\vec{S}))$ and $\psi(x') \leq \bar{x}$. Thus there exists $x^* \geq x'$ with $x^* \in \mathrm{conv}(\mathcal{SG}(\vec{S}))$ and $\psi(x^*) = \bar{x}$.

Note that we have defined the objective function c' of the undirected Steiner graph instance such that $c' \cdot \psi(x) = c \cdot x$ for all $x \in \mathbb{R}^{|A|}$. Assume $\psi(x^*) \in \mathrm{conv}(\mathcal{SG}(S))$. We know that there is $x' \in \mathrm{conv}(\mathcal{SG}(\vec{S}))$ with $\psi(x') = \psi(x^*)$. Thus there is a Steiner tree $t \in \mathcal{SG}(\vec{S})$ with $c \cdot t \leq c \cdot x' = c' \cdot \psi(x') = c' \cdot \psi(x^*) = c \cdot x^*$.

For lifting the undirected edges to directed arcs, one can use the computation of optimal Steiner arborescences. For the actual implementation, we used a faster lifting using a lower bound to the value of an optimal Steiner arborescence, provided by the fast algorithm DUAL-ASCENT [16, 24]. For producing facets for the directed Steiner problem, one could compute optimal Steiner arborescences in the FIND-FACET algorithm of Section 4.2.

5 Some Experimental Results

In this section, we present experimental results showing the impact of the methods described before. In this paper we confine ourselves to the presentation of some highlights, namely the largest benchmark instances ever solved (Table 1). Experiments on smaller instances show that vertex splitting can also significantly improve the solution time (Table 2). Note that in the TSP context, local cuts were helpful particularly for the solution of very large instances.

We have chosen the approach of applying these techniques together with the reduction methods [16], because this is the way they are actually used in our program package. Note that without the reductions, the impact of these techniques would be even more impressive, but then these instances could not be handled in reasonable time.

All results were obtained with a single-threaded run on a Sunfire 15000 with 900 MHz SPARC III+ CPUs, using the operating system SunOS 5.9. We used the GNU g++ 2.95.3 compiler with the -O4 flag and the LP-solver CPLEX version 8.0.

Instance	Orig. Size		Red.	Red. Size		LP_{C+FB}		+ vertex splitting		+ local cuts	
	$\|V\|$	$\|R\|$	time	$\|V\|$	$\|R\|$	val	time	val	time	val	time
d15112	51886	15112	5h	22666	7465	1553831.5	20.4h	1553995	21.9h	1553998	21.9h
es10000	27019	10000	988s	4061	1563	716141953.5	251s	716174280	284s	—	
fnl4461	17127	4461	995s	8483	2682	182330.8	5299s	182361	6353s	—	
lin37	38418	172	28h	2529	106	99554.5	1810s	99560	1860s	—	

Table 1: Results on large benchmark instances. In all cases, the lower bound reached the value of the integer optimum (and a tree with the same value was found). A dash means that the instance was already solved to optimality without local cuts. For the instance d15112, we used the program package GeoSteiner-3.1 [23] to translate the TSPLIB [20] instance into an instance of the Steiner problem in networks with rectilinear metric. No benchmark instance of this size has been solved before. The SteinLib [21] instances es10000 and fnl4461 were obtained in the same way. Warme et. al. solved the es10000 instance using the MSTH-approach [22] and local cuts. They needed months of cpu time. The instance fnl4461 was the largest previously unsolved geometric instance in SteinLib. The SteinLib instance lin37 originates from some VLSI-layout problem, is not geometric, and was not solved by other authors. Without lower bound improvement techniques, the solution of the instances would take much longer (or was not even possible in case of d15112). The number of vertex splits varied between 8 (lin37), 21 (es10000), 173 (fnl4461) and 321 (d15112). For d15112 only one additional local cut computation was necessary.

6 Concluding Remarks

We presented two theoretically interesting and empirically successful approaches for improving lower bounds for the Steiner tree problem: vertex splitting and local cuts. Vertex splitting is a new technique and improves the lower bounds much faster than the local cut method, but the local cut method has the potential of producing tighter bounds. Vertex splitting, although inspired by a general approach (see Section 1), is not directly transferable to other problems, while local cuts are a more general paradigm. On the other hand, the application needs some effort, e.g., developing proofs for shrinks and implementation using exact arithmetic. A crucial point is the development of heuristic shrinks, where a lot

instance	LP_{C+FB}	LP_{C+FB} + vertex splitting	LP_{C+FB} + local cuts
es1000fst01	23.8	13.7	21.8
es1000fst02	34.4	33.5	33.5
es1000fst03	9.5	9.5	9.4
es1000fst04	15.1	13.7	15.4
es1000fst05	11.4	11.3	11.3
es1000fst06	41.8	20.2	516.2
es1000fst07	5.7	5.7	5.7
es1000fst08	22.2	17.7	17.5
es1000fst09	17.5	14.5	18.6
es1000fst10	5.5	5.6	5.6
es1000fst11	18.9	18.9	18.6
es1000fst12	23.9	19.0	19.4
es1000fst13	6.5	6.5	6.5
es1000fst14	23.9	16.4	65.9
es1000fst15	13.7	13.9	13.7
Average:	18.3	14.7	51.9

Table 2: Average times for optimal solution of instances of the instance group ES1000FST, using our program package for Steiner trees with different variants of lower bound computation. For each instance and each variant the numbers give the average times of 5 runs. Note that using local cuts may slow down the solution, as in some cases the bound-based reduction techniques solve the instance faster using weaker but faster bounds. In the TSP context local cuts were applied successfully only on large instances with long solution times. Looking at the results for each instance one can see that enabling vertex splitting never deteriorates the running time significantly, but sometimes improves it by 50%. For those instances where vertex splitting had a visible impact, there have been 7.6 vertex splits on the average. If local cuts had a visible impact, on the average 28.5 successful and 229.2 unsuccessful FIND-FACET calls were performed. Adding local cuts to vertex splitting did not change the empirical results as in the relevant cases LP_{C+FB}+vertex splitting was strong enough to solve the instances.

of intuition comes into play and we believe that there is room for improvement. Although the local cut method was originally developed for the Traveling Salesman Problem, its application is much clearer for the Steiner tree problem.

Both methods are particularly successful if there are some local deficiencies in the linear programming solution. On constructed pathological instances the lower bounds are still improved significantly, but the progress is not fast enough to solve such instances efficiently.

Another interesting observation is that the power of the vertex splitting approach can be improved by looking at multiple roots simultaneously. In fact, we do not know any instance where repeated vertex splittings would not bring the lower bound to the integer optimum if multiple roots are used. It remains an open problem to find out if this is always the case.

References

[1] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding cuts in the TSP (A preliminary report). Technical report, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, Piscataway, NJ, 1995.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In Michael Jünger and Denis Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*. Springer, 2001.

[4] E. Balas and M. Padberg. On the set-covering problem: II. An algorithm for set partitioning. *Operations Research*, 23:74–90, 1975.

[5] X. Cheng and D.-Z. Du, editors. *Steiner Trees in Industry*, volume 11 of *Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, 2001.

[6] S. Chopra and M. R. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, pages 209–229, 1994.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[8] C. Gentile, U.-U. Haus, M. Köppe, G. Rinaldi, and R. Weismantel. A primal approach to the stable set problem. In R. Möhring and R. Raman, editors, *Algorithms - ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 525–537, Rom, Italy, 2002. Springer.

[9] U.-U. Haus, M. Köppe, and R. Weismantel. The integral basis method for integer programming. *Mathematical Methods of Operations Research*, 53(3):353–361, 2001.

[10] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland, Amsterdam, 1992.

[11] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[12] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

[13] D. Naddef and S. Thienel. Efficient separation routines for the symmetric traveling salesman problem i: general tools and comb separation. *Mathematical Programming*, 92(2):237–255, 2002.

[14] D. Naddef and S. Thienel. Efficient separation routines for the symmetric traveling salesman problem ii: separating multi handle inequalities. *Mathematical Programming*, 92(2):257–283, 2002.

[15] T. Polzin and S. Vahdati Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112:241–261, 2001.

[16] T. Polzin and S. Vahdati Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112:263–300, 2001.

[17] T. Polzin and S. Vahdati Daneshmand. Partitioning techniques for the Steiner problem. Research Report MPI-I-2001-1-006, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2001.

[18] T. Polzin and S. Vahdati Daneshmand. Extending reduction techniques for the steiner tree problem. In R. Möhring and R. Raman, editors, *Algorithms - ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 795–807, Rom, Italy, 2002. Springer.

[19] T. Polzin and S. Vahdati Daneshmand. On Steiner trees and minimum spanning trees in hypergraphs. *Operations Research Letters*, 31, 2003.

[20] G. Reinelt. TSPLIB — a traveling salesman problem library. *ORSA Journal on Computing*, 3:376 – 384, 1991.

[21] SteinLib. `http://elib.zib.de/steinlib`, 1997. T. Koch, A. Martin, and S. Voß.

[22] D. M. Warme, P. Winter, and M. Zachariasen. Exact algorithms for plane Steiner tree problems: A computational study. In D-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 81–116. Kluwer Academic Publishers, 2000.

[23] D. M. Warme, P. Winter, and M. Zachariasen. GeoSteiner 3.1. `http://www.diku.dk/geosteiner/`, 2001.

[24] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting
MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi
MPI-I-2002-1-008	P. Sanders, J.L. Trff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	M. Hoefer	Performance of heuristic and approximation algorithms for the uncapacitated facility location problem
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Trff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property
MPI-I-2002-1-002	F. Grandoni	Incrementally maintaining the number of l-cliques
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials

MPI-I-2001-2-006	H. Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators
MPI-I-2001-2-004	H. de Nivelle	Translation of Resolution Proofs into Higher Order Natural Deduction using Type Theory
MPI-I-2001-2-003	S. Vorobyov	Experiments with Iterative Improvement Algorithms on Completely Unimodel Hypercubes
MPI-I-2001-2-002	P. Maier	A Set-Theoretic Framework for Assume-Guarantee Reasoning
MPI-I-2001-2-001	U. Waldmann	Superposition and Chaining for Totally Ordered Divisible Abelian Groups
MPI-I-2001-1-007	T. Polzin, S. Vahdati	Extending Reduction Techniques for the Steiner Tree Problem: A Combination of Alternative-and Bound-Based Approaches
MPI-I-2001-1-006	T. Polzin, S. Vahdati	Partitioning Techniques for the Steiner Problem
MPI-I-2001-1-005	T. Polzin, S. Vahdati	On Steiner Trees and Minimum Spanning Trees in Hypergraphs
MPI-I-2001-1-004	S. Hert, M. Hoffmann, L. Kettner, S. Pion, M. Seel	An Adaptable and Extensible Geometry Kernel
MPI-I-2001-1-003	M. Seel	Implementation of Planar Nef Polyhedra
MPI-I-2001-1-002	U. Meyer	Directed Single-Source Shortest-Paths in Linear Average-Case Time
MPI-I-2001-1-001	P. Krysta	Approximating Minimum Size 1,2-Connected Networks
MPI-I-2000-4-003	S.W. Choi, H. Seidel	Hyperbolic Hausdorff Distance for Medial Axis Transform
MPI-I-2000-4-002	L.P. Kobbelt, S. Bischoff, K. Khler, R. Schneider, M. Botsch, C. Rssl, J. Vorsatz	Geometric Modeling Based on Polygonal Meshes
MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-2-001	F. Eisenbrand	Short Vectors of Planar Lattices Via Continued Fractions
MPI-I-2000-1-005	M. Seel, K. Mehlhorn	Infimaximal Frames: A Technique for Making Lines Look Like Segments
MPI-I-2000-1-004	K. Mehlhorn, S. Schirra	Generalized and improved constructive separation bound for real algebraic expressions
MPI-I-2000-1-003	P. Fatourou	Low-Contention Depth-First Scheduling of Parallel Computations with Synchronization Variables
MPI-I-2000-1-002	R. Beier, J. Sibeyn	A Powerful Heuristic for Telephone Gossiping
MPI-I-2000-1-001	E. Althaus, O. Kohlbacher, H. Lenhof, P. Mller	A branch and cut algorithm for the optimal solution of the side-chain placement problem
MPI-I-1999-4-001	J. Haber, H. Seidel	A Framework for Evaluating the Quality of Lossy Image Compression
MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages
MPI-I-1999-3-002	J. Raskin, P. Schobbens	The Logic of Event Clocks
MPI-I-1999-3-001	S. Vorobyov	New Lower Bounds for the Expressiveness and the Higher-Order Matching Problem in the Simply Typed Lambda Calculus
MPI-I-1999-2-008	A. Bockmayr, F. Eisenbrand	Cutting Planes and the Elementary Closure in Fixed Dimension
MPI-I-1999-2-007	G. Delzanno, J. Raskin	Symbolic Representation of Upward-closed Sets