A Generic Approach to Solving the Steiner Tree Problem and Variants

Masterarbeit
bei Prof. Dr. Thorsten Koch

vorgelegt von Daniel Markus Rehfeldt[1]
Fachbereich Mathematik der
Technischen Universität Berlin

Berlin, den 9. November 2015

[1]Konrad-Zuse-Zentrum für Informationstechnik Berlin, rehfeldt@zib.de

Eidesstattliche Erklärung (German Affidavit)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbständige und eigenhändige Anfertigung versichert an Eides statt

Berlin, den 9. November 2015

(Daniel Rehfeldt)

Contents

Acknowledgments

I would like to thank my parents for their support and encouragement, not only during my academic studies, but furthermore during my entire life. Had it not been for the – for want of a better word – relentless efforts of my mother throughout my childhood and adolescence to focus my attention on studying, this thesis would not have found its way into existence. I would like to thank Gerald Gamrath for his support and patience in the face of numerous questions and requests, throughout my time at Zuse Institute Berlin. Not even safe from me on parental leave, he would support and encourage me with this thesis until the last minute before submission. I would like to thank Thorsten Koch for coming up with the idea of developing a general-purpose Steiner tree solver and his support throughout the realization. I would like to thank Stephen Maher for his support and suggestions for this thesis. I would like to thank Yuji Shinano for his feedback and help. I would like to thank Gerald, Stephen and Yuji for the great time in Boston, where we enjoyed the endless opportunities of American shopping and watched none other than Kobe Bryant himself play in the TD Garden; and where a previous version of the solver described in this thesis competed in the 11th DIMACS Challenge. Last but certainly not least, I would like to thank Cees Duin who relieved me from my struggles to get hold of an electronic version of his dissertation, which had indeed been written in the old pre-PDF days, by sending my a hard copy from the Netherlands.

1 Introduction

The **Steiner tree problem in graphs** (**SPG**) is one of the classical NP-hard problems [Kar72] and has been extensively studied both theoretically and practically. Given an undirected connected graph $G = (V, E)$, weights $c : E \to \mathbb{Q}_+$ and a set $T \subset V$ of *terminals*, the problem is to find a minimum-weight tree $S \subset G$ that spans T. While the SPG is commonly cited to entail a variety of practical applications [Dui93, HRW92, Pol04, PS02, VD04], problems representable by the classic SPG are rarely encountered in practice. This assertion is evidenced by the fact that from the more than $1,000$ SPG instances collected in the standard benchmark library SteinLib [KMV01] less than a fifth can claim practical origins and even of those most are more suitably formulated as a *rectilinear Steiner minimum tree problem*. Still, real-world applications can be found in problems such as for instance the design of fiber-optic networks [LLL+14].

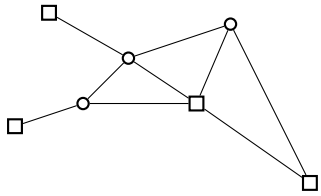
However, when the focus is widened to also consider variations of the SPG, a different picture emerges: Numerous applications can be found that include the SPG as a subproblem or that are formulated as a modified version of it (as will be evinced in the course of this thesis). Despite the strong relationship between the different Steiner tree problem variants, solution approaches employed so far have been prevalently problem specific.

In contrast, the aim of this thesis is to devise and implement a general purpose approach, resulting in a solver that is able to solve both the classical Steiner tree problem and many of its variants without modification. Pursuing this approach, a further objective of this thesis is to solve a wide range of benchmark instances, by means of both known and newly developed methods.

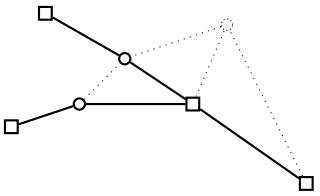
The origins of our thesis can be traced to the announcement of the 11th DIMACS Challenge, dedicated to practical solving methods for the Steiner tree problem and variants. A base was provided by the model and code of the SPG solver Jack-III described in [KM98]. However, being more than 15 years old, Jack-III lacked many modern developments regarding SPG solution methods and did not provide a general MIP framework for easily incorporating additional variants of the Steiner problem. Our resolution to overcome these limitations of Jack-III was to combine the model of [KM98] with the state-of-the-art MIP-framework SCIP [Ach09].

Furthermore, we have developed several Steiner variant specific methods concerning preprocessing and heuristics. These developments have been motivated by the endeavour to render our approach competitive with problem specific state-of-the art solvers. By virtue of the plugin-based structure of SCIP, these adaptations can be easily implemented without modifying the underlying generic solving methods.

The major contributions of this thesis to the existing literature are as follows:



(a) A Steiner tree problem



(b) A possible solution

Figure 1: Illustration of a Steiner tree problem graphs (left) and a possible solution (right). Terminals are drawn as squares, non-terminal nodes as circles.

- The transformation of 11 Steiner problem variants into a general form, using the versatility of MIP models.
- The development of problem specific solving methods, most saliently preprocessing techniques, but also primal heuristics.
- The integration of the new methods and established approaches described in the literature, culminating in a powerful exact Steiner tree solver that exceeds existing problem specific solvers on several benchmark classes.

As an additional central feature, the Steiner problem solving package associated with this thesis is freely available for academic research, setting it apart from all state-of-the-art Steiner tree solvers we are aware of, with the exception of the geometric Steiner problem package GeoSteiner [WWZ00].

Finally, it should not go unnoticed that part of the results presented in this thesis have already been published in [GKM$^+$15] in the course of the 11th DIMACS Challenge. Likewise, a previous version of the solver introduced in this thesis was submitted to the Challenge and achieved a first place in the category *rooted prize-collecting Steiner tree problem*.

1.1 Preliminaries and Definitions

As a preliminary to this thesis, we deem it of interest to point out that concerning the various theorems, lemmata and corollaries introduced hereinafter, we generally follow the precept to formally establish only those that have been devised by the author of this work. References to proofs for the remainder will be provided.

Concerning definitions, in this passage some important basic concepts for this thesis are introduced. Unfortunately, notations in the literature on graph theory vary widely and even for Steiner problems a common, coherent parlance is non-existing. In this work, we attempt to keep the notations mostly in accordance with [Pol04].

For a given undirected graph $G = (V, E)$ define $n := |V|$ and $m := |E|$ and for a directed graph $D = (V, A)$ likewise $n := |V|$ and $m := |A|$. In this thesis, graphs are unexceptionally simple, i.e. without parallel edges or arcs, and finite, i.e. $n, m < \infty$ holds. We refer to the vertices and edges of a subgraph $G' \subset G$ as $V[G']$ and $E[G']$ respectively and analogously to the vertices and arcs of a directed subgraph $D' \subset D$ as $V[D']$ and $A[D']$.
For two adjacent vertices $v_i, v_j \in V$ an edge is denoted by e or $\{v_i, v_j\}$ and an (directed) arc by a or (v_i, v_j). For a graph endowed with edge or arc costs $c : E \to \mathbb{Q}$ or $c : A \to \mathbb{Q}$, respectively, the cost of an edge $e = \{v_i, v_j\}$ is denoted by c_e or c_{ij}, the cost of an arc $a = (v_i, v_j)$ by c_a or c_{ij}. The triple (V, E, c) is called **network**.

Let $W \subset V$; the **cut** induced by W is defined as $\delta(W) := \big\{\{w, \overline{w}\} \in E \mid w \in W, \overline{w} \in V \setminus W\big\}$. For a directed graph we distinguish between $\delta^+(W) := \big\{(w, \overline{w}) \in A \mid w \in W, \overline{w} \in V \setminus W\big\}$ and $\delta^-(W) := \delta^+(V \setminus W)$. If $W = \{v_i\}$, we write $\delta(v_i)$ instead of $\delta(\{v_i\})$. In the undirected case the **degree** of v_i is defined as $|\delta(v_i)|$, i.e. the number of incident edges. In the directed case we discern between the **indegree** $|\delta^-(v_i)|$ and the **outdegree** $|\delta^+(v_i)|$.

In an undirected graph a **path** is a subgraph that can be written as an alternating sequence of vertices and edges $v_{i_0}, e_{j_0}, ..., v_{i_{k-1}}, e_{j_{k-1}}, v_{i_k}$, all distinct from one another, such that $e_{j_q} := \{v_{i_q}, v_{i_{q+1}}\}, q = 0, ...k-1$. In a directed graph a **directed path** or **dipath** is an alternating sequence of distinct arcs and vertices $v_{i_0}, a_{j_0}, ..., v_{i_{k-1}}, a_{j_{k-1}}, v_{i_k}$ with $a_{j_q} := (v_{i_q}, v_{i_{q+1}})$. Occasionally it is convenient to regard P as a subgraph; this allows the representation of all vertices and edges in P as $V[P]$ and $E[P]$, respectively. We denote the subpath of P between two vertices $v_r, v_s \in V[P]$ by $P(v_r, v_s)$. Assuming that P starts in v_r and ends in v_s, its "interior" is defined by $P^\circ := (V[P] \setminus \{v_r, v_s\}, E[P])$. The **length** or **cost** of a (directed) path P in a graph endowed with a cost function c is defined as $C(P) := \sum_{e \in E[P]} c_e$. The **distance** between two vertices v_i, v_j is the length of a shortest (directed) path starting in v_i and ending in v_j and is denoted by $d(v_i, v_j)$ or occasionally $\vec{d}(v_i, v_j)$ in the directed case. Based on this notion, in the undirected case

for $W \subset V$ the **distance network** to (G, c) can be defined as: $D_G(W) = (W, W \times W, d)$.

Now consider a Steiner tree problem (V, E, T, c). Thereupon, we define $s := |T|$ and for the sake of simplicity $V := \{v_1, ..., v_n\}$ as well as $T := \{t_1, ..., t_s\}$. Note that in the literature the vertices in T are commonly referred to as **terminals**, while vertices in $V \setminus T$ are called **Steiner vertices**. Furthermore, we define the distance function $\underline{d}(v_i, v_j)$ as the length of a shortest path between v_i and v_j without intermediary terminals. On a related note, in [Dui93] an $O(m + n \log n)$ algorithm was introduced to compute for each non-terminal v_i a constant number of $\underline{d}$-nearest terminals $v_{i,1}, v_{i,2}, ..., v_{i,k}$ (if existent) along with the corresponding paths. We will refer to this procedure as **Duin's nearest terminals algorithm**.

A somewhat less basic concept, which will frequently recur in the course of this thesis, is the following: A **Voronoi diagram** to G is a partition $\{N(t) \mid t \in T\}$ of V (i.e. $V = \bigcup_{t \in T} N(t)$ and $N(t) \cap N(t') = \emptyset$ for $t, t' \in T$, $t \neq t'$) such that:

$$v \in N(t) \Rightarrow d(v, t) \leq d(v, t') \text{ for all } t' \in T.$$

If $v_j \in N(t_i)$, t_i is called the **base** of v_j, denoted by $base(v_j)$. The set $N(t_i)$ is called **Voronoi region** of t_i and an edge $\{v_j, v_k\}$ such that $base(v_j) \neq base(v_k)$ is called **Voronoi-boundary edge**. Given G, T and c, the Voronoi diagram can be computed in $O(m + n \log n)$ by slightly adapting Dijkstra's algorithm [FT84] in such a way that all terminals are considered as start vertices of distance zero [Meh88]. The concept of Voronoi diagrams can be used to compute a minimum spanning tree $S_D(T)$ to the Distance Network $D_G(T)$ in $O(m + n \log n)$ [Meh88].

Finally, the directed equivalent of the SPG the **Steiner arborescence problem** (**SAP**) [HRW92] is defined as follows: Given a directed graph $D = (V, A)$, costs $c : A \to \mathbb{Q}_{\geq 0}$, a set $T \subset V$ of terminals and a root $r \in T$, a directed tree (or arborescence) $S = (V_S, A_S) \subset D$ is required such that: first, for all $t \in T$ the tree S contains exactly one directed path from r to t and second, $\sum_{a \in A_S} c_a$ is minimized. We define $T^r := T \setminus \{r\}$. An SPG can be transformed to an SAP by replacing each edge by two anti-parallel arcs of the same cost and distinguishing an arbitrary terminal as the root. This procedure results in a one-to-one correspondence between the respective solution sets; more details will be provided in Section 3.1.

1.2 Background and Complexity

The focus of this thesis is set on practical solving methods for the SPG and variants. Therefore, results of more theoretical nature such as tight approximation algorithms (which have been shown to be of little practical value [CGSW14, GHNP01]) are only touched upon. Surveys on such theoretical aspects can be found in [CGSW14, PS02].

1.2.1 A Brief History

Historically, the Steiner problem in graphs is derived from a problem that has become known as the **Euclidean Steiner tree problem**: Given $k \in \mathbb{N}$ points in the plane, connect them by lines of minimum total length in a manner such that any two points are interconnected by line segments either directly or using intermediary points. Its history can be traced back to the 17th century when P. Fermat formulated the special case of $k = 3$. With Fermat's name already linked to a more renowned problem, J. Steiner, who would pose the same problem more than one hundred years later, eventually became its namebearer [Dui93, Pol04]. However, since past a certain point the clarity of history inevitably falls victim to the fog of legend, no absolute certitude can be claimed. Notwithstanding, the Steiner problem in graphs as it is known today can be found in a publication by S. Hakimi in 1971 [Hak71]. Since then, hundreds of articles concerning the Steiner tree problem in graphs and variants have been published; see [HRW92] for a comprehensive, albeit somewhat outdated survey, or [Hau] for an up-to-date one.

A more detailed account of the history of the Steiner tree problem can be found in [BGTZ14].

1.2.2 Complexity

The decision variant of the SPG, with edge weights in the natural numbers, is strongly NP-complete and the optimization variant NP-hard [Kar72]. However, there are several polynomially solvable special cases, the arguably two most important ones being $|T| = 2$, which corresponds to finding a shortest path between two vertices, and $|T| = n$, which corresponds to finding a minimal spanning tree in G. Further polynomially solvable special cases can be found in [HRW92].

Concerning approximability, it has been demonstrated that the SPG is APX-complete, even with its edge weight being in $\{1, 2\}$ [BP89]. This means that there exists an $\epsilon > 0$ (which in the case of the SPG can be chosen as $\frac{96}{95}$ [CC08]) such that finding a $(1 + \epsilon)$-approximation to the problem is NP-hard. Although approximation algorithms have been designed that come as close as 1.39 to an optimal solution value [BGRS13], their empirical results are distinctly inferior to heuristic methods, see [CGSW14].

For the Steiner problem variants many complexity results can be derived from the SPG (more detail is provided in the respective subsections). A comprehensive compilation concerning both the complexity and approximability of Steiner problem variants can be found in [Hau].

2 The Steiner Tree Problem in Graphs

Being the archetype of all variants discussed in this thesis, the Steiner tree problem in graphs will be presented in the comparatively most detail. This proceeding is further justified by the fact that the approach described in the following establishes a blueprint for solving the different Steiner variants introduced in Section 3.

Based upon an integer programming formulation of the SPG, our solving approach encompasses three major components:

- First, preprocessing in the shape of reduction techniques; key to the performance of our solver.
- Second, primal heuristics; indispensable to find good or even optimal solutions to hard problems.
- Third, a branch-and-cut procedure used to compute a lower bound and prove optimality; the core of our solver.

2.1 Formulating the Problem: Integer Programming

For practical problems, often a mathematical description is not enough, but one seeks to obtain a reformulation that allows a solution by means of computers. To this end, we discuss several formulations of the SPG as integer programs (IPs). Thereupon, linear relaxations of these reformulations can be solved or approximated. This is not only crucial for computing lower bounds, but forms moreover the foundation for a powerful propagator which will be introduced in Section 2.5. The reader not familiar with the concept of integer programming may find some guidance in [BW05].

On the look-out for (mixed) integer programming formulations of the SPG, the seeker soon finds himself faced with a plenitude of different suggestions. This thesis does not attempt to provide a coherent overview on all these formulations, and indeed falls far short of doing so, but merely offers a hopefully insightful glimpse by introducing four basic formulations and moreover the, more extensive, formulation used in the solver. Nevertheless, it should be noted that the strength of the linear relaxation of a formulation is of pivotal importance to the practical performance of superimposed algorithms, so it is well worthwhile to study the quality of the different approaches. A both empirical and theoretical discussion of most common formulations and the hierarchy among them can be found in [Pol04]. In their conclusion, the authors suggested the use of the formulation employed in our solver.

In order to discuss the strength of different formulations some technicalities are inevitable. For this purpose, let P_0 be an integer program and assume that it is to be minimized. We denote the corresponding linear relaxation by LP_0. Furthermore, the values of the optimal solutions to P_0 and LP_0 are denoted by $v(P_0)$ and $v(LP_0)$ respectively. Given two relaxations R_1 and R_2 of the same optimization problem, R_1 is **stronger** than R_2 if its optimal solution value is not higher than that of R_2 for all instances of the underlying problem. If R_2 is also stronger than R_1, the relaxations are **equivalent**, otherwise R_1 is **strictly stronger** than R_2.

2.1.1 Cut Formulations

A natural way to formulate the SPG as an integer program is by associating with each edge $e \in E$ a binary variable x_e, indicating whether e is contained in the Steiner tree ($x_e = 1$) or not ($x_e = 0$). This conception paves the way for the first, arguably straightforward formulation [Ane80]:

Formulation 1. *Undirected Cut Formulation (P_{UC})*

$$\min c^T x \tag{1}$$

$$x(\delta(W)) \geq 1 \qquad \text{for all } W \subset V,\ 0 < |W \cap T| < |T|, \tag{2}$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E. \tag{3}$$

Note that we define $x(F) := \sum_{e \in F} x_e$ for any $F \subset E$. Thereupon, one verifies that the constraints (2) ensure the existence of paths from each terminal to all others in a feasible solution. In this way, it can be readily demonstrated that the SPG can be solved by means of the undirected cut formulation [Ane80].

A different approach is spawned by considering the transformed SAP to an SPG, see Section 1.1. In line with the previous formulation, associate with each arc $a \in A$ a binary variable y_a indicating whether a is contained in the Steiner arborescence ($y_a = 1$) or not ($y_a = 0$). A directed formulation [Won84] can thereupon be stated as:

Formulation 2. *Directed Cut Formulation (P_{DC})*

$$\begin{aligned} &min \quad c^T y && && (4)\\ &y(\delta^-(W)) \;\geq\; 1 && \text{for all } W \subset V,\ r \notin W, W \cap T \neq \emptyset, && (5)\\ &y_a \;\in\; \{0,1\} && \text{for all } a \in A. && (6) \end{aligned}$$

The constraints (5) make sure that all feasible solutions contain directed paths from the root to each additional terminal.

The relation between the directed and undirected formulation, including polyhedral aspects, has been widely discussed in the literature [CR94, GM93, MW95]. We briefly state the most important results here:

- $v(LP_{UC}) \geq v(LP_{DC})$ and $\sup\left\{\frac{v(LP_{UC})}{v(LP_{DC})}\right\} = 2$ [Dui93].
- For a transformed SAP the value $v(LP_{DC})$ is independent of the choice of the root [GM93].

The undirected formulation can be tightened by the *Steiner partition inequalities* introduced in [GM90] and the *odd hole inequalities* [CR94], but still LP_{DC} remains strictly stronger than LP_{UC}. The heretofore results suggest to favor the directed over the undirected formulation and indeed we will subsequently do so.

2.1.2 Flow Formulations

Another way to formulate the SPG is to again consider the equivalent SAP and construct a subgraph in which a $|T^r|$ units efflux is originated in r, with one unit reaching every terminal other than the root. Introducing additional flow variables z_{ij} to indicate the flow on an arc $(i,j) \in A$, one obtains the two following formulations [Won84].

Formulation 3. *Flow Formulation (P_F)*

$$min \quad c^T y \tag{7}$$

$$\sum_{(r,j)\in\delta^+(r)} z_{rj} = |T^r|, \tag{8}$$

$$\sum_{(j,i)\in\delta^-(i)} z_{ji} - \sum_{(i,j)\in\delta^+(i)} z_{ij} = 1 \qquad \text{for all } i \in T^r, \tag{9}$$

$$\sum_{(j,i)\in\delta^-(i)} z_{ji} - \sum_{(i,j)\in\delta^+(i)} z_{ij} = 0 \qquad \text{for all } i \in V \setminus T, \tag{10}$$

$$z_{ij} \leq |T^r| y_{ij} \qquad \text{for all } (i,j) \in A, \tag{11}$$

$$z \in \mathbb{R}^{|A|}_{\geq 0}, y \in \{0,1\}^{|A|}. \tag{12}$$

Constraint (8) guarantees a $|T^r|$ unit efflux from r, the constraints (9) a net flow of one unit into each terminal other than the root and the constraints (10) the flow conservation at the non-terminal vertices. Finally, (11) ensures that each arc with positive flow is selected as a part of the Steiner arborescence.

However, a major drawback of this formulation is the weakness of the constraints $y_{ij} \leq |T^r| x_{ij}$, leading to an effete LP-relaxation: Since the cost of all selected arcs is to be minimized, in an optimal solution to LP_F each arc variable y_{ij} is set to $\frac{z_{ij}}{|T^r|}$ which can be much smaller than one. Seeking to amend this disadvantage, [Won84] suggested an extended formulation that affiliates with each terminal $t \in T^r$ a separate flow w^t, also known as multicommodity flow [VW10].

Formulation 4. *Multicommodity Flow Formulation (P_{MF})*

$$min \quad c^T y \tag{13}$$

$$\tag{14}$$

$$\sum_{(r,j)\in\delta^+(r)} w^t_{rj} = 1 \qquad \text{for all } t \in T^r, \tag{15}$$

$$\sum_{(j,i)\in\delta^-(i)} w^t_{ji} - \sum_{(i,j)\in\delta^+(i)} w^t_{ij} = 0 \qquad \text{for all } i \in V \setminus T, \tag{16}$$

$$\sum_{(j,t)\in\delta^-(t)} w^t_{jk} - \sum_{(k,j)\in\delta^+(t)} w^t_{tj} = 1 \qquad \text{for all } t \in T^r, \tag{17}$$

$$w^t_{ij} \leq y_{ij} \qquad \text{for all } (i,j) \in A, t \in T^r, \tag{18}$$

$$w \in \mathbb{R}^{|T|\times|A|}_{\geq 0}, y \in \{0,1\}^{|A|}. \tag{19}$$

The constraints of P_{MF} are similar to those of P_F, but the LP-relaxation of the former is strictly stronger and empirically considerably tighter than that of the original formulation, since the relation between the flow variables

(z and w, respectively) and the arc variables (y) is more accurately modelled by the constraint (18).

An interesting result is that not only $v(LP_{MF}) = v(LP_{DC})$ holds, but even a bijection can be established between the feasible solutions (y, w) to LP_{MF} and y to LP_{DC}, see [Won84].

2.1.3 Flow Balance Inequalities

In [KM98] a new group of flow based constraints was suggested to strengthen the directed cut formulation. Although not all of them improve the LP-relaxation, these constraints lead to a considerable advantage in practical solving by column generation and have been frequently employed in advanced Steiner problem solvers [KM98, LLL+14, Lju04, Pol04].

Formulation 5. *Flow-Balance Directed Cut Formulation (P_{CF})*

$$min \quad c^T y \tag{20}$$

$$y(\delta^-(W)) \geq 1 \qquad \text{for all } W \subset V, r \notin W, W \cap T \neq \emptyset, \tag{21}$$

$$y(\delta^-(v)) \begin{cases} = & 0, \text{if } v = r; \\ = & 1, \text{if } v \in T^r; \\ \leq & 1, \text{if } v \in N; \end{cases} \quad \text{for all } v \in V, \tag{22}$$

$$y(\delta^-(v)) \leq y(\delta^+(v)) \qquad \text{for all } v \in V \setminus T, \tag{23}$$

$$y(\delta^-(v)) \geq y_a \qquad \text{for all } a \in \delta^+(v), v \in V \setminus T, \tag{24}$$

$$y_a \in \{0, 1\} \qquad \text{for all } a \in A. \tag{25}$$

By adding the constraints (23) to P_{DC} the corresponding LP-relaxation is rendered stronger than that of P_{DC}, see [Dui93]. The remainder of the additional constraints do not improve the value of the LP-relaxation as shown in [Pol04].

Formulation P_{CF} is the one used in our solver. However, due to the exponential number of constraints (21), the problem is solved by a branch-and-cut approach that is elaborated in Section 2.4.

2.2 Simplifying the Problem: Reduction Techniques

When it comes to solving NP-hard problems, reductions can play a crucial role as a preprocessing tool, as for example with the travelling salesman problem [ABCC11]. The SPG is no exception: With various publications [BP87, Bea84, DV89, HRW92], most saliently Duin's magnum opus [Dui93], having set the stage, reduction techniques were often a pivotal ingredient in practical solving techniques in the 1990's decade. In the wake of those developments, several techniques were again improved in terms of both efficiency and capability by Polzin and Daneshmand [Pol04, VD04], additionally closing some gaps that had been left open. The methods deployed by them could reduce the number of edges for the tested instances (all from the SteinLib) by 78 percent. Furthermore, their solver, which heavily relies on reduction techniques, has remained until today the fastest SPG solver described in the literature (except for certain artificially constructed instances as in the PUC test set, designed to defy preprocessing).

Being one of its three main components (besides heuristics and branch-and-cut), reduction techniques likewise occupy a prominent position in the solving approach described in this thesis, not only for the SPG, but also for its variants. Subsequently, several important reduction techniques from the literature are introduced (and several extensions are suggested). Naturally, they fall far short of being exhaustive; for further representations we refer to [Dui93, Pol04, VD04]. However, many of the techniques introduced in the literature are special cases of other ones [Dui93, HRW92]. In this context, we attempt to avoid any redundancy in the presented methods.

Finally, it should be marked that the subsequently introduced reduction techniques are not only important for solving the Steiner tree problem in graphs, but furthermore pave the way for several of the Steiner variant specific, novel reduction methods described in the sections 3.1.1, 3.4.2, 3.5.2 and 3.8.1.

2.2.1 Definitions

Reduction techniques, in general, strive to reduce the size of a given instance while allowing a re-transformation of all solutions to the reduced problem back to the original one and preserving at least one optimal solution (if there exists one). In the case of the SPG this means reducing the number of both edges and nodes. Several additional definitions are necessary to conceive the techniques described in this section. They are largely stated in accordance with [Pol04]. In the following, let (V, E, T, c) be an SPG and select two arbitrary but distinct vertices $v_i \in V$ and $v_j \in V$.
Let $\{v_i, v_j\} \in E$, thereupon v_j can be **contracted** into v_i by performing the following steps:

- For each edge $\{v_j, v_k\} \in E$, $k \neq i$: if $\{v_i, v_k\} \notin E$, add this edge

endowed with costs c_{jk} to E; if $\{v_i, v_k\} \in E$ and additionally $c_{ik} > c_{jk}$, set $c_{ik} = c_{jk}$. Denote the set of all newly inserted or modified (with respect to their costs) edges by $E_L \subset \delta(v_i)$.

- If $v_j \in T$, add v_i to T.
- Delete v_j and all incident edges.

Let (V, E, T, c) be an SPG, $\{v_i, v_j\} \in E$ and (V', E', T', c') the problem resulting from contracting v_j into v_i. A solution S' to the problem resulting from such a contraction can be easily transformed to a solution S to the original problem: If $E_L \cap E[S'] = \emptyset$, then S' can be adopted unaltered; otherwise, first add $\{v_i, v_j\}$ and v_j to S' and second extract each $e_l \in E_L \cap E[S']$, concurrently inserting the corresponding original edge. Further, if v_j is a terminal in the original graph, then $\{v_i, v_j\}$ is included in S.

Anther important concept that is pivotal to several reduction techniques is a generalization of the "regular" distance (i.e. the length of a shortest path between two vertices): First, define an **elementary path** as a path containing terminals only (but not necessarily) at its endpoints. A path P in G can be broken into one or more elementary paths. The length of a longest elementary path in P is called the **Steiner distance** of P. Accordingly, the **bottleneck Steiner distance** (also referred to as **special distance** [KM98]) $s(v_i, v_j)$ between two (distinct) vertices $v_i, v_j \in V$ is the minimum Steiner distance taken over all paths between v_i and v_j. Similarly, the **restricted bottleneck Steiner distance** $\overline{s}(v_i, v_j)$ is the bottleneck Steiner distance between v_i and v_j excluding the edge $\{v_i, v_j\}$. A picturesque description of the bottleneck Steiner distance was given in [KM98]: Consider (V, E, T, c) as a road system, with the terminals T as petrol stations and c the length of the roads E. Imagine a driver who acquires to reach v_i starting from v_j (or vice versa). In this case, $s(v_i, v_j)$ is the minimum distance he or she needs to be able to drive without replenishing their tank in order to reach their destination,

2.2.2 Basic Reductions

The subsequent five tests originate from [Bea84] and [Dui93]:

Non-terminal of Degree one ($\mathbf{NTD_1}$): A non-terminal of degree one and its incident edge can be deleted.

Non-terminal of Degree two ($\mathbf{NTD_2}$): A non-terminal v_i of degree two and its incident edges $\{v_i, v_j\}$, $\{v_i, v_k\}$ can be substituted by a single edge $\{v_j, v_k\}$ of cost $c_{jk} := c_{ij} + c_{ik}$. If this results in two parallel edges, only the one of smaller cost, or in the case of equal costs an arbitrary one, is retained.

Terminal of Degree one ($\mathbf{TD_1}$): If there is a terminal $t_i \in T$ of degree one with adjacent vertex v_j, the latter can be contracted into t_i.

Terminal of Degree two (**TD_2**): Let $t_i, t_j \in T$, such that $\delta(t_i) = \{e_k, e_l\}$ with $c_{e_k} \leq c_{e_l}$ and $e_k = \{t_i, t_j\}$. Then t_j can be contracted into t_i.

Minimum Terminal Edge (**MTE**): If there is a $\{t_i, t_j\} \in E$ that is of minimum weight among all edges incident to t_i and further satisfies $t_i, t_j \in T$, then t_j can be contracted into t_i.

The successive execution of those five tests will be referred to as **Degree Test (DT)**. Since in each of the first four tests only vertices of maximum degree two are checked and in MTE each edge needs to be scrutinized at most once, the worst-case complexity of DT is of $O(m)$; under the assumption that deleting, inserting and contracting an edge can be realized in $O(1)$.

2.2.3 Alternative Based Reductions

In this subsection several tests are introduced that utilize the existence of alternative solutions. They can be dichotomized into exclusion and inclusion tests: The former use the argument that for any solution containing a specified part of the graph (e.g. an edge) there is a solution not containing this part of smaller or equal cost. The latter use the converse argumentation: For any solution not containing a specified part of the graph an additional solution can be found that contains this part and is of less or equal cost. First, the exclusion tests are introduced. Remarkably enough, all following alternative-based tests can be realized in $O(m + n \log n)$ [Pol04].

The first and arguably most effective alternative-based test is spawned by the following lemma. Due to its significance and to offer some insight into the concept of the bottleneck Steiner distance, which will resurface in varying shapes throughout the remainder of this thesis, we for once aberrate from our precept not to provide proofs already available in the literature.

Lemma 1. *Every edge $\{v_i, v_j\} \in E$ with $c_{ij} > s(v_i, v_j)$ can be discarded.*

Proof. Suppose that a minimum Steiner tree contains an edge $\{v_i, v_j\}$ with $c_{ij} > s(v_i, v_j)$ and let S be such a tree. Removing $\{v_i, v_j\}$ leaves S divided into two components S_1 and S_2. By definition of s, there is at least one path P between v_i and v_j with Steiner distance $s(v_i, v_j)$. Furthermore, on this path there is at least one elementary path P' that joins S_1 and S_2. Since $c_{ij} > s(v_i, v_j)$, incorporating S_1, S_2 and P' yields a Steiner tree of smaller cost than S. But this contradicts the initial assumption that S is of minimum weight. □

The lemma was first introduced in [DV89]. Since computing all exact bottleneck Steiner distances can be very time consuming, we use a test working with upper bounds that was suggested in [Pol04] and is delineated in the following:

For two terminals t_i and t_j, the bottleneck Steiner distance $s(t_i, t_j)$ can be calculated by determining a longest edge on the fundamental path between

t_i and t_j in the spanning tree $S_D(T)$, which can be constructed in $O(m + n \log n)$, see Section 1.1. By transforming the problem of finding such an edge for each pair of adjacent terminals to an instance of the off-line nearest common ancestor problem [Tar79], an overall bound of $O(m + n \log n)$ can be preserved.

For two non-terminals v_i and v_j an upper bound on $s(v_i, v_j)$ is used: To each non-terminal v_r the (constant) k $\underline{d}$-nearest terminals $v_{r,1}, ..., v_{r,k}$ are initially computed and the upper bound

$$\hat{s}(v_i, v_j) := \min_{a,b \in \{1,...,k\}} \left\{ \max\{\underline{d}(v_i, v_{i,a}), s(v_{i,a}, v_{i,b}), \underline{d}(v_j, v_{i,b})\} \right\}$$

is used instead of $s(v_i, v_j)$. The $\hat{s}$ values are not pre-computed since this could destroy the $O(m+n \log n)$ bound and usually not all of the k^2 possible combinations have to be examined: For instance, this is the case if the test condition is already satisfied before the end of the computation. Moreover, one can for example utilize the lower bound:

$$\tilde{s}(v_i, v_j) := \max\{d(v_i, base(v_i)), d(v_j, base(v_j))\}.$$

If v_i and v_j are part of the same Voronoi region, $\tilde{s}(v_i, v_j) = \hat{s}(v_i, v_j)$ holds. If v_i and v_j belong to different Voronoi region and additionally $c_{ij} < \tilde{s}(v_i, v_j)$ holds, the test condition cannot be satisfied. The associated test is called **Bottleneck Steiner Distance** (**SD**).

An affiliated lemma gives rise to an often fast test [Pol04]:

Lemma 2. *Let c_{max} be the maximum cost of an edge in $S_D(T)$. Every edge $\{v_i, v_j\} \in E$ such that $c_{ij} > c_{max}$ can be discarded.*

The associated test is occasionally very effective and can moreover eliminate edges that cannot be removed by the (original) SD test [Pol04].

Scrutinizing the tests described so far, one observes that for examining whether an edge can be deleted, only paths containing at least one terminal are considered. Therefore, a simple test, just computing a shortest path between the endpoints of an edge [HRW92] might lead to additional eliminations. Since this test is rather time consuming, we suggest a more efficient version, which moreover can find paths that contain exactly one intermediary terminal and are at the same time of smaller Steiner distance than any path found by the SD test. Given an edge $e_l := \{v_i, v_j\}$, we run a modified version of Dijkstra's algorithm, terminating as soon as a predefined (constant) number of edges has been processed or the distance of a scanned vertex exceeds c_{ij}. The further modifications are as follows: Starting from v_i, the edge e_l is continually ignored and the algorithm does not proceed from terminals (other than v_i). If v_j has been labelled and the length of the corresponding path between v_j and v_i is not higher than c_{ij}, the edge e_l can already be eliminated. Otherwise, we run the analogous limited version of

Dijkstra's algorithm from v_j, additionally stopping at all vertices that were scanned in the course of the first run. Finally, we iterate over all vertices $v_i^{(k)}$ labelled or scanned during the first execution of Dijkstra's algorithm. If a $v_i^{(k)}$ was labelled or scanned during the second run and further the Steiner distance of the corresponding path (from v_j to $v_i^{(k)}$ to v_i) is not higher than c_{e_l}, the edge e_l can be deleted. We will henceforth refer to this, novel, procedure as **Bottleneck Steiner Distance Circuit (SDC)** test.

Another observation is that both Lemma 1 and 2 can be extended to the case of equality [Pol04, VD04]. For Lemma 1 this is achieved by using the restricted bottleneck distance: An edge $\{v_i, v_j\} \in E$ can be discarded if $c_{ij} \geq \overline{s}(v_i, v_j)$. But deleting this edge may result in altered restricted bottleneck distances, which would require a recalculation of all distances after each elimination. However, all cases that allow the elimination of an edge without changing the (restricted) Steiner bottleneck distances can be identified; to avoid tedious argumentation we refer to [Pol04] for this. For Lemma 2 the situation is similar, but has, to the best of our knowledge, not been discussed in the literature yet. Therefore, we establish a corollary in order to eliminate edges $e_i \in E$ such that $c_{e_i} = c_{max}$. For this purpose, let $S_G(T)$ be the subgraph of G corresponding to a minimum spanning tree $S_D(T)$ of the distance network $(T, T \times T, d)$.

Corollary 3. *Let c_{max} be the maximal cost of an edge in $S_D(T)$. Every edge $e_i \in E$ not contained in $S_G(T)$ and satisfying $c_{e_i} \geq c_{max}$ can be discarded.*

Proof. Suppose that there is a minimum Steiner tree S containing e_i. By removing e_i the tree S is divided into two components S_1 and S_2. As S has been assumed to be of minimum weight, there is at least one terminal t_j contained in S_1 and one terminal t_k contained in S_2. Let P be a path in G corresponding to the path in $S_D(T)$ between t_j and t_k. Since P connects S_1 and S_2, there are vertices $v_1' \in V[S_1] \cap V[P]$ and $v_2' \in V[S_2] \cap V[P]$ such that $P(v_1', v_2')$ includes no additional vertices of S_1 or S_2. In particular, $P(v_1', v_2')$ does not contain any terminals. Therefore, the length of $P(v_1', v_2')$ is at most c_{max}. Further, due to the premises of the corollary, $P(v_1', v_2')$, which is a subset of $S_G(T)$, does not contain e_i. Hence, S_1 and S_2 can be reconnected by $P(v_1', v_2')$ to an optimal Steiner tree not containing e_i. □

The associated test is denoted by **bottleneck Steiner Distance Spanning Tree (SDST)** test. Since after its execution the graph might be unconnected, we subsequently delete all vertices that are not reachable from a terminal.

The bottleneck Steiner distance can further be utilized for another classical reduction test: **Non-Terminals of Degree k** ($\mathbf{NTD_k}$) which was introduced in [DV89] and is based on the following lemma:

Lemma 4. *Let $v_i \in V \setminus T$. The vertex v_i is of degree at most two in at least one minimum Steiner tree if for each set Δ, with $|\Delta| \geq 3$, of vertices adjacent to v_i it holds that: the (summed) cost of all edges in $\delta(v_i) \cap \delta(\Delta)$ is not less than the weight of a minimum spanning tree for the network $(\Delta, \Delta \times \Delta, s)$.*

A prove is provided in [DV89]. If the condition is satisfied, v_i and all incident edges can be discarded, while for each two vertices v_k and v_j adjacent to v_i an edge $\{v_k, v_j\}$ with cost $c_{ik}+c_{ij}$ is inserted. In the case of two parallel edges, only one of minimum cost is retained. Albeit allowing the elimination of a vertex, the NTD_k test might concurrently require additional edges to be added. However, these edges can often be removed by the SD test [Pol04]. To avoid the, possibly computationally non-trivial, insertion and subsequent elimination of an edge, it is sensible to enquire beforehand whether a new edge could be eliminated by the SD test.

Having introduced several exclusion test, we go on to discuss inclusion tests, the first one [HRW92] being:

Lemma 5. *Let t_i be a terminal of degree at least two and let $\{t_i, v_i'\}$ and $\{t_i, v_i''\}$ be a shortest and a second shortest edge incident to t_i. The edge $\{t_i, v_i'\}$ is contained in at least one minimum Steiner tree if there is a terminal t_j such that $t_j \neq t_i$ and:*

$$c_{\{t_i,v_i''\}} \geq c_{\{t_i,v_i'\}} + d(v_i', t_j). \tag{26}$$

In [Pol04] an efficient computation of the distances $d(v_i', t_j)$ is given, based once again on Voronoi regions: Let $\mathit{distance}(t_i)$ be the length of a shortest path from t_i to a terminal t_k with $t_k \neq t_i$, containing the edge $\{t_i, v_i'\}$. By initially setting $\mathit{distance}(t_i) := \infty$ for $i = 1, ..., s$, these values can be easily computed while building the Voronoi diagram: Each time a Voronoi-boundary edge $\{v_q, v_r\}$ with $v_q \in N(t_i)$ and $v_r \in N(t_j)$, $t_j \neq t_i$ is visited, it is checked whether v_q is a successor of v_i' in the shortest paths tree rooted in t_i (which can be realized by marking the successors of v_i' while computing the Voronoi diagram). If this is the case, $\mathit{distance}(t_i)$ is updated to $\min\{\mathit{distance}(t_i), d(t_i, v_q)+c_{qr}+d(v_r, t_j)\}$. Also, although not mentioned in [Pol04], in case of ties during the computation of the Voronoi region $N(t_i)$, the vertices having v_q as a successor should be favoured, since otherwise the $\mathit{distance}(t_i)$ value might get larger than necessary. These deliberations give rise to the following [Pol04]:

Lemma 6. *Assume that the premises of Lemma 5 hold. Condition* (26) *is satisfied for a terminal t_j distinct from t_i if and only if*

$$c_{\{t_i,v_i''\}} \geq c_{\{t_i,v_i'\}} + d(v_i', \mathit{base}(v_i'))$$

if $v_i' \notin N(t_i)$ *and*

$$c_{\{t_i,v_i''\}} \geq distance(t_i)$$

otherwise.

The affiliated test, which contracts v_i' into t_i if successful, is denoted by **Nearest Vertex (NV)**. By virtue of Lemma 6, its worst-case complexity is of $O(m + n \log n)$, see [Pol04]. Additionally, we suggest a novel extension of the NV test that can be implemented with minimal extra work and does not alter the complexity (nor the empirical run time).

Lemma 7. *Let t_i be a terminal of degree at least two, $e_i' = \{t_i, v_i'\}$ a shortest edge and $e_i'' = \{t_i, v_i''\}$ a distinct second edge incident to t_i. The edge e_i' is contained in at least one minimum Steiner tree if there is a terminal t_j with $t_j \neq t_i$ and $t_j \neq v_i''$ such that*

$$c_e \geq c_{e_i'} + d(v_i', t_j) \quad \forall e \in \delta(\{t_i, v_i''\}) \setminus \{e_i'\} \tag{27}$$

is satisfied.

Proof. Suppose that there is a Steiner tree $S = (V_S, E_S)$ such that $e_i' \notin E_S$. Let $\tilde{e}$ be the first edge on the unique path from t_i to t_j in S such that $\tilde{e} \neq e_i''$. Removing $\tilde{e}$ one obtains a tree S_1 containing t_i and a tree S_2 containing t_j. These two trees can be reconnected by a path consisting of e_i' and a shortest path between v_i' and S_2. Denote the thereby obtained tree (after possibly deleting cycle edges) by S'. By virtue of (27) one can acknowledge that $c_{\tilde{e}} \geq c_{e_i'} + d(v_i', t_j)$ holds and go on to infer:

$$C(S') \leq C(S) + c_{e_i'} + d(v_i', t_j) - c_{\tilde{e}} \leq C(S).$$

Consequently, there is at least one minimum Steiner tree containing e_i'. □

In order to verify condition (27), we introduce an approach that is similar to the one described in Lemma 6:

Lemma 8. *Assume that the premises of Lemma 7 hold. Condition* (27) *is satisfied for a $t_j \in T \setminus \{t_i, v_i''\}$ if and only if*

$$c_e \geq c_{\{t_i,v_i'\}} + d(v_i', base(v_i')) \tag{28}$$

if $v_i' \notin N(t_i)$ and

$$c_e \geq distance(t_i) \tag{29}$$

otherwise for all $e \in \delta(\{t_i, v_i''\}) \setminus \{e_i'\}$.

Proof. Let $e \in \delta(\{t_i, v_i'\}) \setminus \{e_i'\}$.

Sufficiency. Suppose that condition (28) or (29) of Lemma 8 is satisfied for a terminal t_i: If $v_i' \notin N(t_i)$, let $t_j := base(v_i')$. Because of $v_i' \notin N(t_i)$, it holds that $t_j = base(v_i') \neq t_i$. Furthermore, a shortest path between v_i' and t_j, being of length $d(v_i', base(v_i'))$, cannot contain any edge incident to v_i'', since first (28) holds and second no intermediary terminal can be contained in the path. Hence, $v_i'' \neq t_j$ and condition (27) is satisfied.

On the other hand, if $v_i' \in N(t_i)$, a terminal t_j exists with $c_{\{t_i, v_i'\}} + d(v_i', t_j) = distance(t_i) \leq c_e$. Moreover, a path corresponding to $distance(t_i)$ cannot contain any edge incident to v_i'', as (29) has been posited. Consequently, condition (27) holds for t_j.

Necessity. Suppose that condition (27) of Lemma 7 holds for vertices $t_i, t_j \in T$ and $v_i', v_i'' \in V$, satisfying the premises of the lemma. If $v_i' \notin N(t_i)$, it can be inferred that:

$$c_e \geq c_{e_i'} + d(v_i', t_j) \geq c_{e_i'} + d(v_i', base(v_i')).$$

Hence, condition (28) is satisfied. Finally, consider the case $v_i' \in N(t_i)$. The path P corresponding to $distance(t_i)$ includes the edge e_i' by definition and its length is therefore at most $c_{e_i'} + d(v_i', t_j)$. Consequently:

$$c_e \geq c_{e_i'} + d(v_i', t_j) \geq distance(t_i),$$

and condition (29) is satisfied. □

We incorporate this condition into the NV test as follows: For each terminal $t_i \in T$, the third shortest edge $e_i''' = \{t_i, v_i'''\}$ is computed as well. If the NV test is successful, we proceed as before. Otherwise, if t_i is of degree two or it holds that the edge e_i''' satisfies condition (28) or (29), respectively, we check for each edge $e \in \delta(v_i'') \cap \delta(\{t_i, v_i''\})$ whether (28) or (29) holds. If this is the case, we contract e_i' into t_i. By restricting the extended test to terminals t_i such that $|\delta(t_i)| \geq K * |\delta(v_i'')|$ with $K \in \mathbb{N}$ constant, the additional costs are of $O(m)$, preserving the total worst-case complexity of $O(m + n \log n)$.

A similar idea can be used for an additional inclusion test denoted by **Short Links (SL)** [Pol04]:

Lemma 9. *Let $t_i \in T$ and $\{v_1, v_1'\}$ and $\{v_2, v_2'\}$ be a shortest and a second shortest Voronoi-boundary edge of $N(t_i)$. The edge $\{v_1, v_1'\}$ belongs to at least one minimum Steiner tree if $c_{\{v_2, v_2'\}} \geq d(t_i, v_1) + c(v_1, v_1') + d(v_1', base(v_1'))$.*

SL can likewise be performed in $O(m + n \log n)$. Furthermore, the test can be extended similarly to NV, but since the accompanying experimental results were predominantly poor, no elaboration is provided here.

2.2.4 Bound Based Reductions

Voronoi regions can be a potent tool in the context of Steiner problems, as has already become evident with the SL test. Subsequently, they are employed to determine a lower bound for an optimal solution that is assumed to satisfy certain additional constraints (e.g. containing a specific edge).

Prelusively, given a Voronoi decomposition N and a terminal $t_i \in T$, we define $radius(t_i)$ to be the cost of a shortest path containing t_i and leaving $N(t_i)$ [Pol04]. These values can be obtained while building the Voronoi diagram, by updating them whenever an edge with endpoints in different Vornonoi regions is being processed. For the sake of notation, it will henceforth be assumed that the terminals are ordered according to non-decreasing *radius* values. Further, recall that for each $v_i \in V \setminus T$ the variables $v_{i,1}$, $v_{i,2}$ and $v_{i,3}$ represent the nearest, second nearest and third nearest terminal to v_i without using intermediary terminals.

All subsequent lemmata except for the novel Lemma 12 can be found in [Pol04]. It should be mentioned that these lemmata were originally stated by using the customary distance function d. However, the proofs provided in [Pol04] can be reused to verify the hereinafter stated, stronger versions using the $\underline{d}$-distance.

Lemma 10. *Let $v_i \in V \setminus T$. If there is a minimum Steiner tree $S = (V_S, E_S)$ such that $v_i \in V_S$, then $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=1}^{s-2} radius(t_q)$ is a lower bound on the weight of S.*

Each Steiner node v_i such that the affiliated lower bound stated in Lemma 10 exceeds a known upper bound can be eliminated. Moreover, if a solution S corresponding to the upper bound is given and v_i is not contained in it, the latter can already be eliminated if the lower bound stated in Lemma 10 is equal to the cost of S. A similar proposition holds for edges in a minimum Steiner tree:

Lemma 11. *Let $\{v_i, v_j\} \in E_S$. If there is a minimum Steiner tree $S = (V_S, E_S)$ such that $\{v_i, v_j\} \in E_S$, then $c_{\{v_i,v_j\}} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{s-2} radius(t_q)$ is a lower bound on the weight of S.*

Furthermore, a stronger version of the antecedent lemma holds, which has not been stated in the literature yet:

Lemma 12. *Let $\{v_i, v_j\} \in E$. If there is minimum Steiner tree $S = (V_S, E_S)$ such that $\{v_i, v_j\} \in E_S$, then L defined by*

$$L := c_{\{v_i,v_j\}} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{s-2} radius(t_q) \tag{30}$$

if $base(v_i) \neq base(v_j)$ *and*

$$L := c_{\{v_i,v_j\}} + \min\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}\} + \sum_{q=1}^{s-2} radius(t_q) \quad (31)$$

otherwise, is a lower bound on the weight of S*.*

Proof. Let $S = (V_S, E_S)$ be a minimum Steiner tree with $\{v_i, v_j\} \in E_S$. Denote the set of all paths in S between v_i and terminals reachable without using the edge $\{v_i, v_j\}$ by $\mathcal{P}_{v_i}$, and analogously the set of all paths in S between v_j and the remaining terminals by $\mathcal{P}_{v_j}$. Additionally, set $\mathcal{P} := \mathcal{P}_{v_i} \cup \mathcal{P}_{v_j}$ and denote by $P_r \in \mathcal{P}$, $r \in \{1, ..., s\}$, the path to terminal t_r (starting in either v_i or v_j).

First, note that a path in $\mathcal{P}_{v_i}$ cannot have edges in common with any path in $\mathcal{P}_{v_j}$, and vice versa, since S is cycle-free. Second, two distinct paths in $\mathcal{P}_{v_i}$ can only have a subpath containing v_i in common, but no additional edges, since this would likewise require a circle in S. The analogous property holds for two paths that are both in $\mathcal{P}_{v_j}$.

Having stated those preliminaries, one can go on to validate the actual statement of the lemma: Choose two paths $P_k \in \mathcal{P}_{v_i}$ and $P_l \in \mathcal{P}_{v_j}$ such that they jointly include a minimum number of Voronoi-boundary edges. For all $r \in \{1, ..., s\} \setminus \{k, l\}$, denote by P'_r the subpath of P_r between t_r and the first vertex not in $N(t_r)$. Suppose that P_k has an edge $e_p \in E_S$ in common with a P'_r: According to the prelusive observations, this would imply the existence of a joint subpath including v_i and e_p. But in this case P_k would contain at least one more edge with endpoints in different Voronoi regions (in order to be able to reach t_k which is by definition not in $N(t_r)$). Furthermore, P_r cannot have any edge in common with P_l, as S is cycle-free. Consequently, P_r would have initially been chosen instead of P_k. Following the same line of argumentation, one validates that likewise P_l has no edge in common with any P'_r.

Conclusively, the paths P_k, P_l and all P'_r are edge-disjoint and their combined cost is therefore a lower bound on the weight of S. To see that L is a lower bound on the combined cost of P_k, P_l and all P'_r, and therefore on S, note that the summed cost of all P'_r is at least $\sum_{q=1}^{s-2} radius(t_q)$ and differentiate between two cases:

First, in the case of $base(v_i) \neq base(v_j)$, the cost of P_k plus the cost of P_l is at least $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1})$, so:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e \\ &\geq c_{\{v_i,v_j\}} + C(P_k) + C(P_l) + \left(\sum_{r \in \{1,\ldots,s\} \setminus \{k,l\}} C(P'_r) \right) \\ &\geq c_{\{v_i,v_j\}} + C(P_k) + C(P_l) + \sum_{q=1}^{s-2} radius(t_q) \\ &\geq c_{\{v_i,v_j\}} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{s-2} radius(t_q). \end{aligned}$$

Therefore, (30) is a lower bound on the weight of T.

Second, in the complementary case of $base(v_i) = base(v_j)$, let $t_b := base(v_i) = base(v_j)$ and note that $\underline{d}(v_i, v_{i,1}) = \underline{d}(v_i, t_b)$ and $\underline{d}(v_j, v_{j,1}) = \underline{d}(v_j, t_b)$. Since not both P_k and P_l can contain t_b (since this would require a cycle in S), their combined cost is at least $\min\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}\}$. Therefore:

$$\begin{aligned} C(S) &\geq c_{\{v_i,v_j\}} + C(P_k) + C(P_l) + \sum_{q=1}^{s-2} radius(t_q) \\ &\geq c_{\{v_i,v_j\}} + \min\left\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1})\right\} \\ &\quad + \sum_{q=1}^{s-2} radius(t_q). \end{aligned}$$

Consequently, L as defined in (31) is a lower bound on the weight of S. □

Besides attempting to directly eliminate a vertex or an edge, one can test whether it is possible to substitute vertices by new edges, analogously to the NTD_k test. As before, this procedure can be extended to the case of equality if a solution corresponding to the upper bound is known. The basis is provided by the following lemma:

Lemma 13. *Let $v_i \in V \setminus T$. If there is a minimum Steiner tree S such that v_i is of degree at least three in S, then $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \underline{d}(v_i, v_{i,3}) + \sum_{q=1}^{s-3} radius(t_q)$ is a lower bound on the weight of S.*

Although the antecedent four lemmata already allow for considerable reductions of a number of instances, sometimes an even stronger bound can be achieved:

Lemma 14. *Let $G' = (T, E')$ be graph in which two vertices t_i and t_j (which correspond to terminals in G) are adjacent if and only if there is an edge*

$\{v_k, v_l\} \in E$ such that $v_k \in N(t_i)$ and $v_l \in N(t_j)$. Additionally, define a cost function d' on E' by

$$d'(t_i, t_j) := \min\{\min\{d(t_i, v_k), d(t_j, v_l)\} + c_{kl} \mid \{v_k, v_l\} \in E \cap (N(t_i) \times N(t_j))\}.$$

The weight, with respect to d', of a minimum spanning tree for G' is a lower bound on the weight of any Steiner tree for (V, E, T, c)

The lemma can be extended to test conditions: Let v_i be a Steiner vertex. The weight of a minimum spanning tree for G' minus the length of its longest edge plus $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2})$ is a lower bound on the weight of any minimum Steiner tree containing v_i. Analogously, a lower bound on the weight of any minimum Steiner tree containing an edge $\{v_i, v_j\}$ can be obtained. The graph G' can be determined while computing the Voronoi regions, and a corresponding minimum spanning tree can be computed in $O(m + s \log s)$ time.

For computing an upper bound we use the RSP heuristic that will be introduced in Section 2.3. The use of the heuristic on the one hand demolishes the $O(m + n \log n)$ time bound so far retained for all tests in this section, but is on the other hand empirically fast for almost all instances that we tested, see Section 2.6. We denote the test associated with the introduced bound-based reduction approaches by **Bound** (**BND**[1]) test. Since a heuristic solution is available, the test also covers the case of lower and upper bound being equal.

2.2.5 Implementation

For the implementation an adjacency representation of the graph is used, with all arcs stored in a single array, see [Koc95]. This representation allows to quickly insert, in $O(1)$, and delete, in $O(\max(\delta(v_i), \delta(v_j))$, any arc (v_i, v_j). The technical details of the realization of the various reduction tests are omitted here, but it should be noted that for each test all actions are performed in a single pass (and not restarted after each operation, e.g. the elimination of an edge).

Reiteration and Ordering

Studying different combinations and orderings of reduction techniques, Polzin concluded in [Pol04] that the tests are not very sensitive to the order of their execution. However, this assumption is not true for the total reduction time.

The underlying precept for the ordering of the reduction methods is to perform the faster ones first so that the more expensive tests are applied to, hopefully, substantially reduced graphs. Furthermore, it seems reasonable

[1] Any resemblance of the abbreviation to existing organizations is purely coincidental.

to perform the two SD test variants prior to the NTD_k tests, since the former reduces, if successful, the degrees of vertices. Additionally, first performing the SD test allows to reuse the computed $\underline{d}$-distances for the NTD_k tests, see [Dui93]. Similarly, due to the NTD_k and SD variant tests deleting edges (and possibly replacing pairs of edges by a single one), they are performed prior to the NV and SL tests. Also, the latter two can be empirically relatively expensive, since they result in the contraction of edges.

All reduction tests are arrayed in a loop that is reiterated as long as a constant proportion (for our solver 0.5 percent) of edges was eliminated during the last run. Thereby, one obtains the same asymptotic time bound as the most expensive performed reduction test; with the exception of the BND test this bound is of $O(m+n\log n)$. Furthermore, during a succeeding loop each test is performed only if it could eliminate a constant proportion (0.1 percent) of edges during the previous run. For the BND test, the (time-consuming) computation of an upper bound is only performed during the first run. Additionally, the BND test is executed if and only if at most three percent of all vertices are terminals; we have observed that the test is otherwise of very little effect.

The ordering that is by default used in our solver is reported in Figure 2.

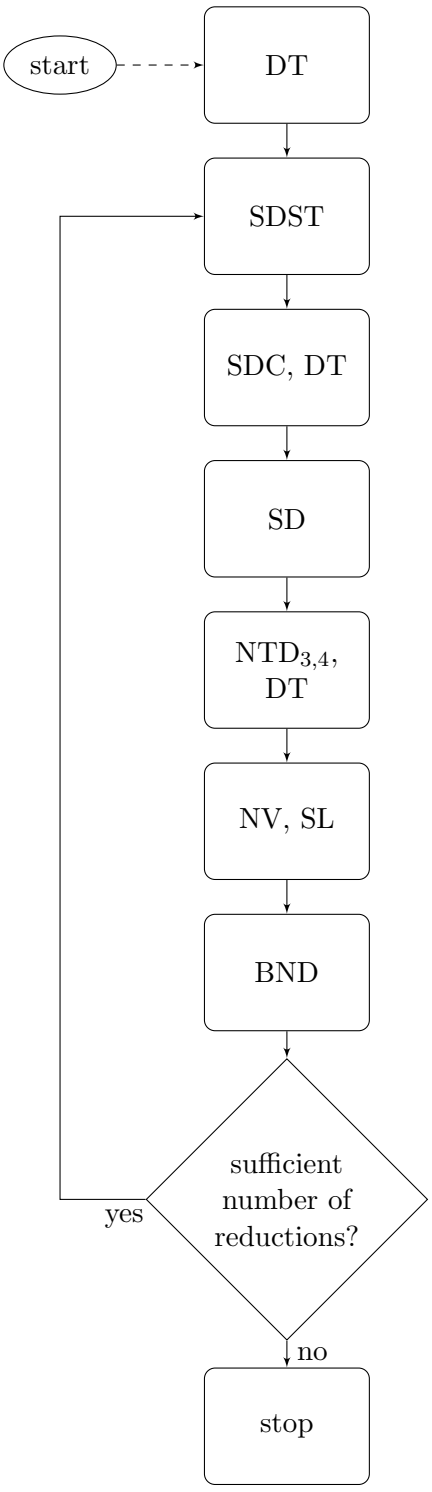


Figure 2: Illustration of the reduction package; once no more than 0.1 percent of all edges could be eliminated within a method-block the latter is disabled for the remainder of the reduction process.

2.3 Finding a Solution: Primal Heuristics

With the Steiner tree problem in graphs being NP-hard it hardly comes as a surprise that a wide range of research has focused on developing efficient primal heuristics. While attempting to obtain Steiner trees of small weight is arguably worthwhile in its own right, an additional virtue arises from the concomitantly yielded upper bound on the optimal solution value: The disposability of sharp upper bounds is indispensable for bound-based reductions, see Section 2.2.4, and more importantly for an efficient branch-and-bound based exact solving approach. Indeed, without using SPG specific heuristics we are unable to find any feasible solution to a number of instances that can otherwise be solved to optimality, as will be demonstrated at the end of this section.

Naturally, only a very small share among the plethora of primal heuristics trialled in the literature is covered in this thesis. The reader wishing to acquire a more comprehensive insight is referred to [dAW02, PUW14, Pol04, RUW01].

2.3.1 Constructive Heuristics

As the name suggests, **constructive heuristics** build up a new solution from the scratch, repeatedly extending the current sub-solution. In this context, the **repetitive shortest path heuristic (RSPH)** can arguably be named as *the* classical and empirically successful primal SPG heuristic. Already introduced in 1980 [TA80], it has found its way into various publications, e.g. [HRW92, PD01, dAW02, Pol04, PUW14] and has moreover been notably improved by [dAW02] and [Pol04] in terms of empirical run time.

The basic concept of the heuristic is very coherent: Starting with a single vertex, in each step the current subtree is connected to a nearest terminal by a shortest path. This procedure is reiterated until all terminals are spanned. Finally, the generated subgraph S is **pruned**, i.e. it is substituted by a minimum spanning tree constructed on the vertices of S and non-terminals of degree one are, repeatedly, removed. To render the overall procedure more potent, the heuristic is started from several distinct vertices.

In its original form the heuristic is implemented by initially computing shortest paths from each terminal to all other vertices and afterwards successively building up a Steiner tree by utilizing the previously obtained shortest paths. The worst-case run time for such a construction is of $O(s(m+n\log n))$, which can be readily verified [Pol04]. In [dAW02] the empirically quite effective observation was made that a modified version of Dijkstra's algorithm [FT84] can be used alternatively. This algorithm is not introduced in full detail here; however, a comprehensible delineation is provided: To grow a Steiner tree starting from a single vertex, one can use Dijkstra's

algorithm, pausing whenever a terminal has been scanned and joining the shortest path to this terminal with the incumbent tree. Throughout the execution of Dijkstra's algorithm, with each vertex $v_i \in V$ an upper bound π_i on the distance between the current subtree and v_i is associated.

The decisive consideration made in [dAW02] was that it is not necessary to reset the π values whenever a terminal has been connected to the current subtree, since the tentative distances are still valid upper bounds on the distances from the new subtree. Instead, one merely needs to reinsert every vertex v_i on the path between the previous tree and the newly incorporated terminal into the priority queue of Dijkstra's algorithm (setting $\pi_i = 0$). Although the worst-case complexity remains at $O(s(m + n \log n))$, empirically the run time is drastically improved [dAW02, Pol04]. We denote this variant, also using different starting points, by **one-phase RSPH**.

A different approach was suggested in [PD01], based on the following considerations: Let $t_i \in T$, $v_j \in V \setminus \{t_i\}$ and consider a shortest path P between t_i and v_j computed in the first phase of the original RSPH implementation. Aspiring to find an a priori criterion for P to not be required during the second phase of RSPH, the following observations can be made: If P contains an intermediary terminal t_k, inserting either the subpath between v_j and t_k or t_i and t_k is always preferable to inserting the whole path P. Similarly, if v_j is contained in a Voronoi region of a terminal $t_k \neq t_i$ and $d(t_i, t_k) \leq d(t_i, v_j)$ is satisfied, instead of inserting P, one could insert the subpath between v_j and t_k or the subpath between t_i and t_k respectively, both of them being of not higher cost than P. Consequently, it is not necessary to obtain shortest paths from each terminal t_i to all other vertices, but one may terminate Dijkstra's algorithm, starting from t_i, as soon as $N(t_i)$ and all neighbouring terminals have been scanned. Furthermore, no paths containing intermediary terminals need to be considered. This variant is denoted by **two-phase RSPH**. We omit the implemental details and refer to [PD01] for this purpose.

Another important observation [KM98] is that the heuristic may not only be used initially, but moreover, with altered edge weights, during the branch-and-cut: Given an optimal LP solution $x \in \mathbb{Q}^E$, the heuristic is called with the edge weights $(1 - x_e) \cdot c_e$ for all $e \in E$. In this way, a stimulus for the heuristic to choose edges contained in the LP solution is provided.

In our implementation one-phase RSPH is customarily used, the two-phase variant is only preferred if at least one third of the vertices are terminals (which rarely is the case for most published test instances). Furthermore, terminals are preferred as starting points as for example suggested in [KM98]. We perform the initial run of the heuristic, with no LP solution available, with 100 start vertices, preferring terminals.

Additionally, the heuristic is called, with altered edge weights, before and after the processing of a branch-and-bound node, after each cut loop

and after each LP solving during a cut loop; each with 16 start vertices. For this purpose we use a novel preference criterion: Given an optimal LP solution $x \in \mathbb{Q}^E$, we affiliate with each vertex v_i the value $\alpha_i := \sum_{e \in \delta(v_i)} x_e$. Let $\alpha^T_{max} := \max_{v_i \in T} \alpha_i$. Thereupon, we associate with each terminal $v_i \in T$ the value α'_i, which is obtained by adding a pseudo-random uniformly distributed number in $[0, \alpha^T_{max}]$ to α_i. Thereby, it is attempted to give a preference to terminals of high α_i value, but still allowing the selection of each terminal. Having set these values, we restrict the first ten runs to terminals, selecting the ones of highest α'_i amount. For the next five runs the not yet selected vertices of highest α_i values are used. Finally, the last run is conducted with the start vertex that has led to the best incumbent solution found by RSPH in the course of the whole heretofore solving process.

Furthermore, ties occurring during the computation and the selection of the shortest paths are broken pseudo-randomly and when no new LP solution is available, each edge cost is additionally multiplied by a pseudo-random uniformly distributed number between 1.0 and 2.5 (the sole exception being the initial run).

2.3.2 Local Search Heuristics

Instead of constructing a solution de novo, as done by RSPH, a different heuristic approach is to improve already existent solutions. Such **local search heuristics** [KV07] have successfully been applied to numerous hard computational problems. A prominent example is the Helsgaun implementation of the Lin-Kernigham heuristic for the travelling salesman problem. This heuristic was not only able to produce optimal solutions to all solved problems it was tested on (as of October 2015, [Hel]), but furthermore allowed to improve the best known solutions for a series of large-scale instances with unknown optima, among them a $1,904,711$-cities instance [Hel09].

Given a solution S to a problem, a local search algorithm examines a **neighborhood** of S, i.e. a set of solutions obtainable from S by performing a predefined set of operations. This examination encompasses finding an improving solution, i.e. one of smaller cost in the case of a minimization problem, or proving that no such solution exists. Subsequently, three local search heuristics from [UW10] that have been incorporated into our solver are sketched. Their, intricate, implementation will not be discussed here, but it should be noted that all of them can be performed with a worst-case complexity of $O(m + n \log n)$. The easiest one is **vertex insertion (V)**, which examines whether an additional vertex v_i can be merged with an existing Steiner tree $S = (V_S, E_S)$ in such a way that the minimum spanning tree on $V_S \cup \{v_i\}$ (which likewise constitutes a Steiner tree) is of smaller cost than S. The last two heuristics use the concept of **key-vertices**, which are terminals or vertices of degree at least three in S. Correspondingly, a **key-path** is a path in S connecting two key-vertices and otherwise containing

only non-key-vertices. In **key-path exchange** it is attempted to replace existing key-paths by less costly ones. Similarly, for **key-vertex elimination** in each step a non-terminal key-vertex and all adjoining key-paths (except for the key-vertices at their respective ends) are extracted and an attempt is made to reconnect the disconnected subtrees at a lower cost. As in [UW10], the execution of vertex insertion followed by the joint execution of key-path exchange and key-vertex elimination is denoted by **VQ**. In our solver, VQ is called for a newly found solution whenever the latter is among the five best known solutions. This number is dynamically reduced if VQ is displaying little effectiveness on the instance being solved.

2.3.3 Recombination Heuristics

Recombination heuristics, better known as **genetic algorithms** (since they are said to mimic the process of evolutionary natural selection), have been widely applied to a vast number of optimization and search problems [ES03, Hin08]. Broadly speaking, solutions out of given pool are, repeatedly, altered and combined, with the ultimate goal of obtaining a new solution that excels all of its predecessors.

Accordingly, we developed a heuristic, denoted simply by **recombination heuristic (RCH)**, for the SPG that comprises in essence a recombination of several known solutions. In the following, RCH is described in the context of an SPG, but it can be naturally extended to cover the different Steiner problem variants discussed in this thesis. Preliminarily, we define the set of solutions to be considered for recombination by $\mathcal{L}$; in the case of our solver $\mathcal{L}$ comprises the, at most 50, best found solutions of the current solving process.

The heart of RCH is the **n-merging** ($n \geq 2$) operation, subsequently defined for a given solution S: This solution is merged with pseudo-randomly selected $n-1$ solutions out of $\mathcal{L} \setminus \{S\}$ to form a new graph G_S consisting of all edges and vertices that are part of at least one of the n solutions. Applying the reduction techniques introduced in Section 2.2 to G_S, we obtain a reduced graph G'_S. Next, a solution to G'_S is computed in two steps.

First, the cost of each edge is pseudo-randomly reduced for each solution it appears in, as suggested by [RUW01] and RSPH is employed. To this end, we compute the α_i values to each vertex v_i of G'_S (considering the new edge weights as an LP solution) and perform 50 runs of RSPH using the vertices of highest α_i values as starting points. Thereupon, denote the best found solution by S'.

Second, after retrieving the original edge costs, VQ is employed on S'. Finally, S' is re-transformed to the original solution space.

The RC heuristic is clustered around the n-merging operation: Given a new solution S, in one *run* consecutively three 2-, two 3- and 4-, and one 5-merges are performed. When a solution S' is generated during an i-merging

with a smaller cost than S, we set $S := S'$ and attempt to add S' to $\mathcal{L}$. Moreover, in this case a new run is started after the conclusion of the current one. The total number of runs is limited to ten. RCH is called whenever r new solutions have been found compared to its last execution. Initially, r is set to 5 (the minimum number of solutions to be available before executing RCH) and modified throughout the solution process, setting $r := 0$ if a solution has been improved during the execution of RCH and $r := r + 1$ otherwise.

Finally, it should not go unmentioned that recombination heuristics for the SPG were already suggested in [RUW01] and [PUW14]. Nevertheless, they differ widely from the heretofore introduced RC heuristic: First, in neither of these publications reduction techniques were used. Additionally, the authors merge merely two solutions and furthermore deviate in several details, e.g. they do not apply a criterion for selecting start vertices for RSPH.

2.4 Solving to Optimality: Branch and Cut

Since the employed model P_{CF} potentially contains an exponential number of constraints, see Section 2.1, starting with the associated LP relaxation becomes prohibitive already for medium-seized instances. Therefore, a branch-and-cut algorithm is employed. In general, branch-and-cut can be seen as a modification of branch-and-bound, starting with only a subset of the model constraints and employing cutting planes at each (or certain designated) branch-and-bound nodes to tighten the LP relaxations [Mit02].

The algorithm employed in this thesis is largely in accordance with the approach described in [KM98].

2.4.1 Separation Methods

The initialization of the branch-and-cut solving process at the root node is done be setting up an LP that contains the constraints (23) and the trivial inequalities $0 \leq y_a \leq 1$ for all $a \in A$. Thereafter, separation routines for the cut inequalities (21) are deployed. For this purpose, four types of methods are implemented in both Jack-III and SCIP-Jack; a more detailed report can be found in [KM98].

Generic Cuts

Considering the edge values of an LP solution as capacities, we only need to check for each $t_i \in T^r$ whether a minimal (r, t_i)-cut is less than one, in order to find violated inequalities. If such a cut exists, it corresponds to a violated inequality, otherwise there is none. When using the *highest label preflow-push algorithm* described in [GT88], the overall running time for determining minimal (r, t_i)-cuts for each $t_i \in T^r$ is of $O(\sqrt{m} + n^2)$.

Back Cuts

The number of separated inequalities can be increased by reversing the flow of each arc and additionally examining the (t_i, r)-cuts for each $t_i \in T^r$. On the downside, however, the above mentioned highest label preflow-push algorithm cannot be used, because for each (t_i, r)-cut the source vertex is changed.

Nested Cuts

The number of separated inequalities can also be increased by **nesting** the cuts: Having found a minimal cut between the root and a $t_i \in T^r$, all corresponding variables in the current LP solution are temporarily locked to one and another attempt to find a (r, t_i)-cut is initiated. The procedure is reiterated until the flow between r and t_i is at least one. This nesting approach can be combined with the back-cuts.

Creep Flow

A different strategy is to attempt to determine a cut of minimal cardinality. This can be done by adding a very small capacity $\epsilon > 0$ (which is defined by SCIP) to all arcs. While this deteriorates the empirical running time for computing a minimal cut, as more arcs have to be considered, the time required for re-optimizing the linear program is substantially decreased [KM98, Pol04].

2.4.2 Branching

Despite the tightness of the employed model, for certain hard instances branching is still necessary. For this purpose we have developed a simple SPG specific branching-rule, which is nevertheless empirically stronger than its refined generic counterparts natively implemented in SCIP. For a more general discussion on branching we refer to [AKM04].

As opposed to customary approaches, instead of branching on variables, i.e. in the case of the SPG on arcs, we employ **vertex branching** [HRW92]. This means that during the branch-and-bound procedure a predefined Steiner vertex is rendered a terminal in one child node and excluded in the complementary node. To determine such a Steiner vertex we suggest the following criterion:

Let $y \in [0,1]^A$ be an LP solution at the current node during the branch-and-cut. Select a vertex $v_i \in V \setminus T$ to branch upon, such that:

$$\Big| \sum_{e \in \delta^-(v_i)} y_e - 0.5 \Big| \tag{32}$$

is minimal among all Steiner vertices. The idea underlying (32) is to keep the branch-and-bound tree balanced. Also, branching on vertices as compared to arcs exerts more influence on the model, since excluding a vertex forces the solver to ignore all of its incident arcs, while just discarding a single arc bears a lower impact. Furthermore, including a vertex as opposed to an edge leads to an enhanced flexibility, since the vertex can usually be contained in different ways in a feasible solution, possibly setting the stage for a better (local) upper bound.

2.5 Implementation

Since this thesis is considered to be of prevalently mathematical nature, we confine ourselves to presenting a synopsis of the implementation and refer to the online documentation [Reh] of SCIP-Jack for more details.

2.5.1 SCIP-Jack: A General Purpose Solver for Steiner Tree Problems

SCIP

The foundation of our Steiner problem package is the academic MIP solver SCIP. Besides being one of the fastest non-commercial MIP solvers [Mit15], SCIP is a general branch-and-cut framework. Its plugin-based design provides a simple method of extension to handle a variety of specific problem classes. Furthermore, SCIP renders ubiquitous mastery of the solving procedure possible and allows the access to detailed processual information.

SCIP is implemented as a C callable library (although C++ wrapper classes are provided for user defined plugins). It should be noted that SCIP does not come with a native LP solver. However, an interface is provided instead, allowing the usage of common commercial solvers such as CPLEX[2] or Gurobi[3]. Furthermore, the LP solver SoPlex [Wun96] is distributed along with SCIP in the *SCIP optimization suite* [BGG+12], which is likewise freely available for academic purposes.

For more information on SCIP see [Ach09].

Jack

Jack, or more precisely *Jack-III*, is a solver for Steiner tree problems in graphs described by Thorsten Koch and Alexander Martin [KM98] in 1998. It uses a branch-and-cut approach based on Formulation 5. Furthermore, it comprises several preprocessing techniques, e.g. DT and computationally expensive but more effective versions of the SD and BD_3 tests. Also, the original version of RSPH, see Section 2.3, is used. At the time of publishing Jack-III was able to solve all problem instances that had hitherto been discussed in the literature to optimality. Since then it has frequently been referred to and the separation approach using Formulation 5 has successfully been reutilized for several different solvers, e.g. [Lju04, Pol04].

Jack-III was prevalently hand-tailored and implemented in C. However, for solving linear programs CPLEX 4.0 was used.

[2] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[3] http://www.gurobi.com/products/gurobi-optimizer/

SCIP-Jack

SCIP-Jack was spawned by the endeavour to embed the hand-tailored Jack-III into a state-of-the-art MIP-framework. The use of a general MIP solver renders the model to be solved highly pliant, which is of pivotal importance to the generic solving approach employed in this thesis. Furthermore, a general framework was presumed to facilitate the incorporation of both, algorithms published in the literature since the creation of Jack-III and ad hoc developed methods (as for example the numerous novel techniques described in this thesis). SCIP, being freely available for academic research and providing the above described features, was a natural choice. Another advantage of this decision comes with the general solution methods natively implemented in SCIP, e.g. cutting planes, being perpetually kept up-to-date by the continuous improvements of the framework.

Concerning the implementation of SCIP-Jack, it indubitably bears mentioning that the original embedding of Jack-III into SCIP was carried out by Gerald Gamrath, Thorsten Koch and Michael Winkler. However, the resulting initial version of SCIP-Jack did not include any of the modifications and extensions described in this thesis. Furthermore, Gerald Gamrath, Stephan Maher and Yuji Shinano were involved in the further development of SCIP-Jack; in this context, each solving method that was not implemented or developed by the author of this thesis is marked as such.

The first plugins to be implemented were a *reader* to read problem instances and *problem data* to store the graph and build the model within SCIP. Within these plugins the reading methods and data structures of Jack-III could be adopted largely unaltered. However, for handling the different Steiner variants introduced in Section 3, extensions were necessary. The heart of the new implementation is a *constraint handler* that checks solutions for feasibility and separates any violated model constraints. The separation methods already existent in Jack-III are deployed for this purpose, while SCIP provides a filtering of cuts to improve numerical stability and dynamic ageing of the generated cuts. By default, only the creep flow strategy (together with the flow inequalities) is used, since it proved to be empirically superior to other combinations, evidenced by both experiments conducted with SCIP-Jack and results reported in [KM98] for Jack-III. However, by means of parameters, each separation method introduced in Section 2.4.1 can be deployed at will by the user.

Additionally, native general-purpose separation methods of SCIP such as Gomory and mixed-integer rounding cuts are employed.

Jack-III already included a large number of reduction methods for the SPG, as described in [KM98]. However, with the developments described in Section 2.2, only the, basic, DT reductions were reused for SCIP-Jack.

For the branch-and-bound search a new plugin was implemented that includes the Steiner specific vertex branching introduced in Section 2.4.2.

Node selection is performed with respect to a best estimate criterion – interleaved with best bound and depth-first search phases [Ach07].

The three SPG specific primal heuristics described in Section 2.2.5 have been implemented in SCIP-Jack as separate plugins, the heuristics integrated in Jack-III are not used.

Figure 3: Depiction of a skipjack tuna, see [Wik].

Finally, also a Steiner problem specific propagator [Ach07] was implemented, based on an idea first published in [Dui93]:

Let v_i be a vertex in an SAP instance and assume that reduced costs are given. Considering the reduced costs as arc weights, v_i can be eliminated if the length of a shortest path from the root to v_i plus the length of a shortest path from v_i to a terminal other than the root exceeds the difference between the best known primal bound and the current dual bound. If the criterion is fulfilled for a vertex v_i, all variables corresponding to incident arcs are fixed to zero. An analogous test can be performed for eliminating single arcs. These two tests allow to eliminate a large number of vertices and edges, as illustrated in [Pol04]. By virtue of this test, the primal and dual solving methods form a distinctly symbiotic relation: An improved dual bound, empirically, allows the RSPH heuristic to calculate better primal solutions, while the thereby provided upper bounds enable the propagator to fix more variables, which facilitates the underlying LP.

2.5.2 Parallel Computing

Yet another advantage of employing SCIP as an underlying framework is the availability of the two parallel extensions ParaSCIP [SAB+12] and FiberSCIP [SHVW13], which are built by using the *Ubiquity Generator Framework* (UG) [SHVW13].

For parallelizing a SCIP based problem specific solver, adding a small glue code and creating a link to one of the UG libraries is already sufficient. This glue code consists of an additional class with a function that issues calls to include all SCIP plugins required for the sequential version of the code. Thereby, no modification to the sequential version of the problem specific solver is required. In this way, users are enabled to obtain their own problem specific parallel optimization solver, which can conduct parallel tree searches both on a shared memory computing environment and on a distributed one.

Using these means, we have created a parallel version of SCIP-Jack. Since this extension has been obtained in corporation with Yuji Shinano, the creator of UG, we do not provide more detailed information, but refer to [GKM+15] instead. However, we note that by using this approach, it is possible to employ large supercomputing resources to solve computationally difficult Steiner tree problems with SCIP-Jack. Some results, including improved bounds for several unsolved problem instances from the literature, will be provided in Section 4.1.1.

2.6 Computational Results and Discussion

When attempting to evaluate the quality of algorithms for NP-hard problems (like primal heuristics) one is commonly faced with a predicament: On the one hand, just assessing the algorithms on a (confined) set of test instances might be incomplete, since with other types of problem structure ostensibly prolific methods might fail dismally. On the other hand, establishing guaranteed performance ratios cannot be regarded a valid substitute, since such results are prevalently too pessimistic from an empirical point of view, due to their worst-case character or a lack of suitable proving techniques.

Since this thesis is focused on practical solving methods, we have decided for empirical evaluation using benchmark instances, but try to keep the range of problem types reasonably wide, as to cover many disparate structures within the instances. The probably most extensive and frequently used [KM98, Pol04, VD04, PUW14] SPG benchmark library is the SteinLib [KMV01]. However, in the course of the 11th DIMACS Challenge additional, often real-world derived, test sets were made publicly available. Since using all available instances in each section would result in an unreasonably lengthy and likely tedious work, a confined set of test instances has been selected for intermediary results. However, a testing on a more comprehensive set of instances, using the complete SCIP-Jack package with its final default configurations as well as comparisons with other state-of-the-art solvers, is provided in Section 4.

For testing the computational results of SCIP-Jack, the focus in this section lies on the impact of the reduction techniques and the primal heuristics. To this end, three sets from the library SteinLib and one published in the 11th DIMACS Challenge are used. Table 1 provides a brief description of the used instance classes. In each line the name of the corresponding test set is given in the first column. The next column lists the number of instances within the test set, thereafter the range of the number of vertices for the individual instances within the class is given. Furthermore, the *Status* indicates whether all instances of a class have been solved in the literature. Conclusively, some elementary characteristics including the origins of the test set are provided. It should be noted that the vienna-I instances come already preprocessed by methods described as being "simple" in [LLL+14].

All computational experiments were performed on a cluster of Intel Xeon X5672 CPUs with 3.20 GHz and 48 GB RAM, running Kubuntu 14.04 and using SCIP 3.2. Underlying, CPLEX 12.6.2 is employed as LP solver since it is empirically faster than the academic LP solver SoPlex, see [Mit15]. For all computations a time limit of two hours is set.

If an instance is not solved to optimality within the time limit, the gap is reported, which is defined as $100 \cdot \frac{\text{pb}-\text{db}}{\text{pb}}\%$ for the final primal and dual bound pb and db, respectively. If no lower or upper bound has been computed, the gap is set to 100 percent. The average gap is obtained as an

Name	Instances	$\|V\|$	Status	Description
E	20	2500	solved	Sparse, randomly generated instances with integer edge-weights between 1 and 10 [Bea89].
I320	100	320	solved	Sparse, randomly generated graphs constructed to defy preprocessing [Dui93].
vienna-I	85	7886 - 17881	solved	Sparse, real-world instances, generated from telecommunication networks [LLL+14].
PUC	50	64-4096	unsolved	Hard instances, hypercubes from code covering and bipartite graphs [RdAR+01].

Table 1: Classes of SPG test instances.

arithmetic mean while averages of both the number of branch-and-bound nodes and the solving time are computed by taking the **shifted geometric mean** [Ach07]: Given values $t_1, ..., t_k \in \mathbb{R}_{\geq 0}$, and a **shift** $s \in \mathbb{R}_{\geq 0}$, the shifted geometric mean is defined as

$$\sqrt[k]{\prod_{i=1}^{k}(t_i + s)} - s. \tag{33}$$

Compared to the arithmetic average, the use of a geometric mean brings the benefit of reducing the influence of very hard instances. On the other hand, the use of a shift is motivated by the endeavour to reduce the effect of very easy instances on the mean value. In the subsequent computations, for both the number of nodes and the solving time a shift of $s = 10$ is used if not specified otherwise.

A synopsis of the performance of SCIP-Jack on the four SPG test classes is provided in Table 2. Each line within the table provides aggregated results for the class specified in the first column. The second column lists the number of instances in the set, the third column states the number of instances solved to optimality within the time limit. Next, the average number of branch-and-bound nodes and the average run time for the solved instances are reported. In contrast, the final two columns give the average number of branch-and-bound nodes and the optimality gap of all instances that could not be solved to optimality within the time limit. For the Steiner problem variants discussed in Section 3 similar tables are provided. The *Timeout* columns are omitted if all instances have been solved to optimality. More intricate, instance-wise, computational results can be found in Appendix C.

The E test set can be solved quickly by SCIP-Jack, only the e18 instance requires comparably more time, being solved in 388 seconds. Of the I320 class 95 instances can be solved to optimality in an average time of 35.5 seconds. The gap of all unsolved instances is smaller than 0.5 percent. The results for the vienna-I test set are less satisfactory, with only about 60 percent of the instances being solved. However, the average optimality gap is smaller than 0.1 percent. Interestingly, only one of the unsolved instances

goes beyond the root node and all other 14 problems that require branching are solved to optimality, none requiring more than 65 branch-and-bound nodes.

Finally, the PUC class appears to be notably more difficult for SCIP-Jack than all preceding test sets. This phenomenon is unsurprising given that more than half of the instances in this set still remain unsolved in the literature. SCIP-Jack only solves nine of the 50 instances and none at the root node. Notably, the number of branch-and-bound nodes for the solved instances is more than four times higher than for the unsolved ones. The results suggest that while the employed model P_{CF} proved to be tight for the other three test sets, often allowing optimal solving at the root node, this is not true for the PUC instances.

The results demonstrate the ability of SCIP-Jack to solve 174 out of 225 SPG benchmark instances, but at the same time its limitations come into the open.

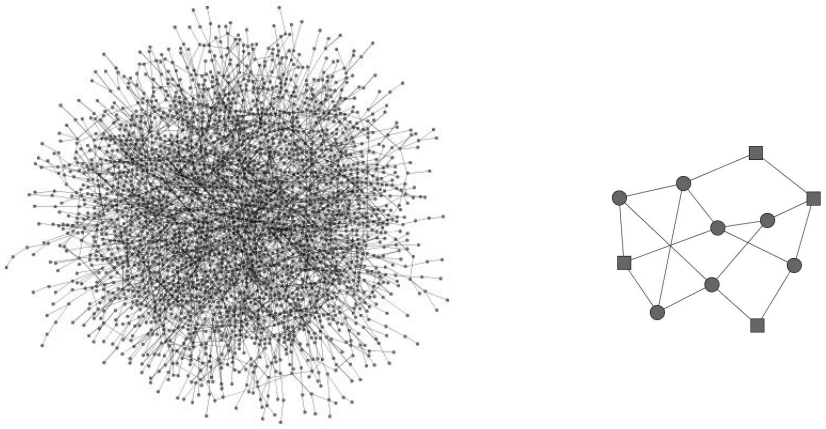
			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
E	20	20	2.0	5.0	–	–
I320	100	95	23.7	35.5	1342.2	0.3
vienna-I	85	50	2.8	431.8	2.2	0.1
PUC	50	9	581.3	88.5	137.0	3.9

Table 2: Computational results for SPG instances.

2.6.1 Impact of Reduction Techniques

In order to measure the strength of each reduction method we do not evaluate it as an individual component, but as part of the entire reduction package. In this way, for each method we report the strength of the reduction set without it. This approach might be considered tendentious as it merely regards the strength of a technique against the backdrop of the particular reduction package employed in this thesis (and its eventual idiosyncrasies), but at the same time it allows to dissect the overall reduction approach more precisely; this exclusion based procedure has already been applied by other authors, for instance in [Ach07]. It should be noted that for measuring the average number of reduction the arithmetic mean is used (also to allow comparisons with results from the literature), while for the run time we resort to the shifted geometric mean with shift $s = 1$.

The results of employing the entire reduction package are reported in Table 3. As a comparison, the results of the basic $O(m + n \log n)$ reduction package employed in [Pol04] are provided. The reported execution times are similar to ours, however they were computed on a SPARC-III CPU with 900 MHz, so on a common machine we expect their reduction package to



(a) e01 original (b) e01 reduced

Figure 4: Illustration of an instance of test set E before and after presolving (terminals are drawn as squares).[a]

[a]Created using *yEd* (http://www.yworks.com/en/products/yfiles/yed/)

be faster. We put this down to the fact that our reduction methods are reiterated more extensively (until less than 0.1 % of all edges are removed) and, to a lesser degree, that in [Pol04] more efficient implementations of certain reduction methods are used; e.g. our package does not include the nearest-common-ancestor sub-method for the SD test.

Also in terms of average numbers of reductions the results in [Pol04] are superior, which is non-surprising as the authors employed a larger number of methods. Nevertheless, our reduction methods prove to be notably strong on all instance classes except for the synthetic, hard PUC set. Even though also the I320 test set was designed to defy preprocessing, its number of edges and vertices can still be notably reduced. Most salient are the reductions for the E test set with less than 14 percent of the edges remaining, but also the real-world, large vienna-I instances can be heavily reduced, to less than a third of their original size. Figure 4 visualizes the strength of the reduction techniques on the instance e01 of class E: Of a problem originally encompassing 2500 vertices and 3125 edges, an instance consisting of only seven Steiner vertices and four terminals remains, being solvable by hand.

Class	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]	Remaining Edges[%] in [Pol04]
E	24.4	13.6	1.1	3.0
I320	83.4	82.5	0.2	71.8
vienna-I	31.2	30.0	14.8	(no results)
PUC	98.4	99.2	0.2	99.3

Table 3: Computational results of the SPG reduction package.

Having established the overall picture, the results illustrated in Table 4 allow us to cut through the surface and analyze the impact of the individual components within the reduction package: The first four rows list the percentual change in the number of remaining edges resulting from successively excluding each reduction method; the last four lines provide the corresponding percentual change with respect to the run time of the entire reduction package. In this way, the first column of each line states the test set to be considered. Ensuing, each of the next seven columns provides the result of excluding the affiliated reduction method specified in the head of the table. It should be noted that positive values signify a favourable impact of the respective reduction method on the overall package: With respect to the reduced edges a positive value denotes a higher amount of remaining edges when disabling the specified method; likewise, a positive percentage for the overall preprocessing time indicates that the reduction package requires more time as a consequence of excluding the respective method.

First, it can be remarked that each exclusion of a method diminishes the strength of the reduction package. However, the leverage varies widely. The strongest impact is exerted by the DT test: Concerning the reduction of edges it outnumbers all other methods on each of the four test sets. However, for the E set it is also notably time-consuming. Next is the SD test: While it is not effective on the I320 and PUC sets (designed to defy preprocessing), it demonstrates its strength on the other two. Not only is it the second most effective test in terms of reduced edges, but it also decreases the execution time of the reduction package for both E and vienna-I.

Shifting the point of view towards the other end, the SDC test inevitable catches the attention. Although it can claim credit for a certain reduction of edges, it is also notably time-consuming. A decrease of the constant upper bound or a later execution of the method within the reduction loop might seem reasonable, but also decreases its strength. Of even less effect is the SDST test with only 0.2 percent on the E class and less than 0.1 otherwise.

Addressing efficiency, the BND test sticks out: It manifests a solid effectiveness at minimal extra cost, defying its poor theoretical worst-case complexity.

Finally, having performed additional experiments to evaluate our novel extensions, we note that the modification of the NV test described in Section 2.2.3 eliminates eight percent more edges, compared to the original test. Additionally, the extension of the BND test, see Lemma 12, allows to eliminate around 13 percent more edges on average, as compared to the original test. For both extensions less than one percent of additional execution time is required (compared to the original NV and BND tests).

Concluding, the employed reductions are notably efficient (and indeed indispensable for the overall exact solving results) on all but one of the tested instance sets. Nevertheless, several additional reduction methods could be incorporated, see [Pol04] and computational enhancements of existing ones

	Class	DT	SDST	SD	SDC	$NTD_{3,4}$	NV,SL	BND
Edges	E	**+205.2**	**+0.2**	**+148.8**	**+2.9**	**+41.8**	**+18.6**	**+105.2**
	I320	**+12.1**	**+0.0**	0	**+0.1**	**+1.0**	**+0.1**	**+7.8**
	vienna-I	**+132.5**	**+0.0**	**+86.4**	**+14.3**	**+35.9**	**+12.4**	0
	PUC	**+0.8**	0	0	**+0.0**	0	0	0
Time	E	-41.7	-14.6	**+12.2**	-34.4	-22.0	-9.9	**+0.9**
	I320	-4.4	-2.1	-4.9	-68.7	-8.4	-2.1	**+0.7**
	vienna-I	**+1.6**	-0.5	**+7.2**	-10.2	**+4.7**	**+3.8**	-0.1
	PUC	-7.8	-8.0	-17.5	-68.6	-19.0	-7.4	-7.5

Table 4: Computational results of the Steiner problem presolving. Each column reports the results of the reduction package without the specified method. The values denote the percentual change with respect to the default settings (see Table 3).

could be made to improve their efficiency (for example the nearest-common-ancestor approach for the SD test).

2.6.2 Impact of Primal Heuristics

To avoid the tedious enumeration of all possible combinations, for measuring the efficiency of the heuristics we will just consider three reasonably natural arrangements of the three (types of) heuristics: First, merely RSPH, second RSPH along with VQ and third RSPH, VQ and RCH. The selected order can be justified by the fact that VQ starts from a feasible solution, and RCH both integrates VQ and requires the existence of solutions. The computations are performed on instances already reduced by the standard presolving of SCIP-Jack, see Section 2.2.5. Only instances are considered that cannot already be solved by the reduction techniques alone.

The experimental results illustrated in Table 5 are subdivided into four consecutive blocks consisting of four lines each. The first block lists the results obtained from disabling all Steiner problem heuristics. Accordingly, the next ones state the results of using the RSP, the VQ and finally the RC heuristic in an accumulative manner. In each line, the test set to be examined is specified in the first column, followed by the number of non-trivial instances that the set contains after the preprocessing. Next, the number of solved instances and the arithmetic mean of the gap among the unsolved problems is reported. Finally, the last column states the average execution time of all non-trivial instances within the set, given as the shifted geometric mean.

Scrutinizing the first four lines of Table 5, displaying the results of using neither of the three Steiner heuristics, one can see that all E and most of the I320 instances are solved, with the gap for the unsolved instances being small. However, for the vienna-I set the gap is very high, which is due to the fact that for 31 instances no primal bound could be computed (and the

	Name	Instances	Solved	∅ Gap [%]	∅ Time [s]
none	E	18	18	–	9.7
	I320	100	94	0.28	54.7
	vienna-I	85	53	96.88	1232.4
	PUC	50	9	76.07	3189.2
RSPH	E	18	18	–	5.4
	I320	100	95	0.36	44.2
	vienna-I	85	53	0.15	1172.3
	PUC	50	9	5.08	3290.8
+VQ	E	18	18	–	4.8
	I320	100	95	0.34	56.4
	vienna-I	85	53	0.15	1248.0
	PUC	50	9	4.47	3313.5
+RCH	E	18	18	–	5.6
	I320	100	95	0.28	46.7
	vienna-I	85	53	0.13	1225.6
	PUC	50	9	3.89	3237.5

Table 5: Computational results of employing the Steiner problem heuristics in accumulative stages.

gap is set to 100 percent). Likewise, to 31 instances of the PUC class no primal bound could be computed. Accordingly, the average gap is notably high.

When the RSP heuristic is used the image changes, but mostly with respect to the average gap: It is vastly reduced for both the vienna-I and PUC class, owed to the fact that to all instances a feasible primal solution could be found. For the I320 test the gap is higher, owned to the fact that one more instance can be solved to optimality; for the remaining, mutually, unsolved instances the gap is smaller when employing RSPH. Finally, the average execution time is reduced by around 45 percent on the E class, by 20 percent on I320 and by five percent on vienna-I. In contrast, it increases for the PUC test set by three percent.

Additionally employing the local heuristics VQ, the change is limited: For the E test set the time is reduced, but for the rest of the test classes it has increased slightly. However, the average gap for both the I320 and PUC instances is lessened.

Finally, the combination of all three heuristics, as illustrated in the last four lines of Table 5, displays the lowest average gaps compared to all three prior compounds. However, with respect to the run time the results are only moderately better compared to not employing Steiner heuristics and only one more problem could be solved to optimality. The results seem to suggest that the combination of all three heuristics is more potent when it comes to solving hard instances. This behaviour becomes most pronounced with the PUC test set, which contains 29 instances not solved to optimality

in the literature yet.

Summarizing, the combination of RSPH, VQ and RC considerably helps generate good primal solutions for hard instances and enables SCIP-Jack to keep the optimality gap for all except the PUC instances below 0.6 percent. Supplementary to the results already presented, in 98 % of all cases the first solution was found by RSPH. For all other instances SCIP constructed a feasible trivial solution prior to the first call of RSPH, which was significantly improved by the execution of this heuristic.

2.6.3 Branching

This subsection is concerned with providing empirical information on an additional solving component of SCIP-Jack, namely the novel Steiner vertex branching rule introduced in Section 2.4.2.

For measuring the effectiveness of the branching rule, it is compared with the branch-and-bound search organized by SCIP, formerly used in SCIP-Jack [GKM+15]: This procedure encompasses a default hybrid branching rule [AB09], that combines strong branching and pseudo costs with history information. For both the old and new branching strategy, node selection is performed with respect to a best estimate criterion – interleaved with best bound and depth-first search phases [Ach07].

For the purpose of comparison, all instances of the classes E, I320, vienna-I and PUC (see Table 1) that required branching were collected. Of these 89 instances, one is from set E, 41 are from I320, 18 from vienna-I and 29 from PUC. The results leave little ambiguity: Table 6 shows that the novel Steiner vertex branching rule allows to solve seven more problems to optimality. Moreover, although more unsolved problems remain for the Hybrid branching strategy, the average gap is still higher than for the Steiner vertex branching (where the unsolved problems should be harder on average); on the mutually unsolved instances the gap resulting from applying the new Steiner vertex branching is even more than 30 percent smaller than the gap resulting from the Hybrid strategy.

The notable differences between the two branching strategies come despite the fact that on average almost half of the time is spend at the root node, i.e. prior to the respective branching rule exerting any effect. Finally, it can be seen that the number of required nodes is only moderately smaller for the Steiner vertex branching.

			Optimal		Timeout	
Branching	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
Hybrid (old)	89	57	72.3	425.0	2129.2	2.0
Vertex (new)	89	64	96.4	343.7	2001.2	1.8

Table 6: Computational results for SGP instances requiring branching.

Complementary to Table 6, the overall performances of the two branching rules are illustrated in Figure 5. The shifted geometric execution time (of all instances) is provided along with the number of solved instances and the (arithmetic) average gap of the unsolved instances. It can be seen that the average time for the instances solved by using Hybrid branching is almost 50 percent higher, as compared to the results obtained by using Steiner vertex branching.

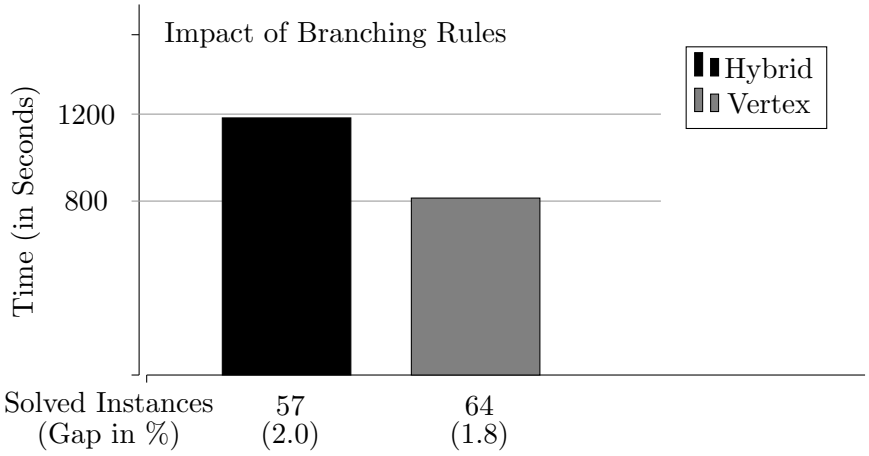


Figure 5: Results of Hybrid and Steiner vertex branching on 89 SPG instances within E, I320, I, PUC. The run time is given as the shifted geometric mean on all instances, with a limit of two hours. The gap is stated as the arithmetic mean of all respectively unsolved instances.

Finally, we expect to achieve a further improvement of the performance of SCIP-Jack in the future by refining the introduced Steiner vertex branching approach. Nevertheless, the results suggest that the newly developed Steiner branching rule is already superior to the sophisticated but generic default branching of SCIP.

3 Variants of the Steiner Tree Problem

With SCIP-Jack being able to cope with SPG instances, the next step is to extend the scope and set about solving Steiner problem variants more frequently derived from real-world applications. In order to verify the validity of the hereinafter introduced transformations, it will be continually proceeded as follows:

First, a one-to-one correspondence between the solution sets of the original and the transformed problem is demonstrated. Second, a linear relation between the respective solutions values is established. This relation implies that a problem can be solved on its transformed solution space.

The time limit for the computational experiments within this section is set to two hours.

3.1 The Steiner Arborescence Problem

Observing that the SPG can be solved as a Steiner arborescence problem, it can be verified that the SAP is NP-hard as well. Further theoretical results, such as on hardness and approximability, can be found in [CCC+98, HK03]. Since it constitutes the backbone for solving both the Steiner problem in graphs, see Section 2, and most Steiner problem variants discussed in the remainder of this thesis, the SAP occupies a central position in our generic solving approach. For this reason, a formalized description of the transformation already introduced in Section 2.1 is provided subsequently. In contrast to the SPG, arcs can be of zero cost, since such arcs can in general not be contracted without loosing optimal solutions.

Transformation 1 (SPG to SAP)**.**
Input: *An SPG* $P = (V, E, T, c)$
Output: *An SAP* $P' = (V', A', T', c', r')$

1. *Set* $V' := V$, $T' := T$, $A' := \{(v, w) \in V' \times V' : \{v, w\} \in E\}$.
2. *Define* $c' : A' \to \mathbb{Q}_{\geq 0}$ *by* $c'_a = c_{\{v,w\}}$, *for* $a = (v, w) \in A'$.
3. *Choose a root* $r' \in T'$ *arbitrarily.*
4. ***Return*** (V', A', T', c', r').

Although the transformation is well-known, e.g. [KM98], it has apparently not been formally established yet.

Lemma 15 (SPG to SAP)**.** *Let* $P = (V, E, T, c)$ *be an SPG and* $P' = (V', A', T', c')$ *an SAP obtained by applying Transformation 1 on* P. *Denote by* $\mathcal{S}$ *and* $\mathcal{S}'$ *the sets of solutions to* P *and* P' *respectively. Thereupon,* $\mathcal{S}'$ *can be mapped bijectively onto* $\mathcal{S}$ *by assigning each* $(V'_{S'}, A'_{S'}) \in \mathcal{S}'$ *a* $(V_S, E_S) \in \mathcal{S}$ *such that*

$$V_S := \{v \in V : \ v \in V'_{S'}\}, \tag{34}$$
$$E_S := \{\{v, w\} \in E : \ (v, w) \in A'_{S'} \ \text{or} \ (w, v) \in A'_{S'}\}. \tag{35}$$

The objective value is preserved.

Proof. First, it can be observed that (34) and (35) form indeed a mapping $\mathcal{S}' \to \mathcal{S}$: Each arc of a solution to P' is substituted by its undirected counterpart and the vertices are maintained. Since for an SAP all terminals have to be spanned by an directed tree, the corresponding undirected tree likewise contains all terminals.

To see the one-to-one correspondence, let $S = (V_S, E_S) \in \mathcal{S}$ and proceed as follows:

Surjective. In the following, a tree $S' = (V'_{S'}, A'_{S'}) \in \mathcal{S}'$ is constructed that is mapped by (34) and (35) to S, demonstrating that the mapping is a surjection. Initially, set $V'_{S'} := V_S$ and $A'_{S'} := \emptyset$. Traverse (V_S, E_S), e.g. using breadth-first search, starting from r' and add for each $w \in V_S$ visited from $v \in V_S$ the arc (v, w) to $A'_{S'}$. $S' := (V'_{S'}, A'_{S'})$ is a solution to P' as it consists of directed paths from r' to all terminals and by applying (34) and (35) we obtain S.

Injective. S' is the only solution to P' that is mapped by (34) and (35) to S: Each $\tilde{S}' \in \mathcal{S}' \setminus \{S'\}$ contains at least one arc (v, w) such that $(v, w) \notin A'_{S'}$ and $(w, v) \notin A'_{S'}$, since only substituting arcs in $A'_{S'}$ by there anti-parallel counterparts would not allow directed paths from the root to all vertices $V'_{S'}$. Consequently, the solution $\tilde{S} \in \mathcal{S}$ corresponding to $\tilde{S}'$ contains the edge $\{v, w\}$, whereas S does not. Therefore, $\tilde{S}'$ is not mapped onto S.

Finally, since for each $\{v, w\} \in E_S$ either $(v, w) \in A'_{S'}$ or $(w, v) \in A'_{S'}$ and vice versa, the costs of S' and S are equal. □

3.1.1 Reductions

Although various publications have addressed the Steiner arborescence problem, surprisingly we could not find any that discussed reduction techniques. The reason for this phenomenon might be sought in a lack of publicly available and computationally challenging test instances. However, it might also be due to the, at first glance perhaps surprising, fact that developing strong reduction techniques for the SAP is much more delicate than for its undirected kinsman, the SPG. The orientation of the tree renders the SAP much less pliant to strategies such as alternative-based reduction, as will become evident in the course of this section. Nevertheless, we have succeeded in developing several, novel, reduction methods, which will be presented hereinafter.

Throughout this subsection it will be presupposed that an SAP instance $P_{SAP} = (V, A, T, c, r)$ is given and that $r = t_1$. Furthermore, the reader is reminded of the consistently used notation $T^r := T \setminus \{r\}$.

Basic Reductions

By observing that to each non-terminal contained in an optimal solution there is at least one incoming and one (non anti-parallel) outgoing arc, the first four reduction techniques ensue forthwith. Although not explicitly mentioned, these four reductions were already used in [GKM+15].

***Non-terminal of in-degree zero (NTI$_0$)**: A non-terminal v_i with $\delta^-(v_i) = \emptyset$ can be deleted along with all incident arcs.*

***Non-terminal of out-degree zero (NTO$_0$)**: A non-terminal v_i with $\delta^+(v_i) = \emptyset$ can be deleted along with all incident arcs.*

***Non-terminal with one neighbour (NTN$_1$)**: A non-terminal v_i with exactly one adjacent vertex can be deleted along with its incident arcs.*

***Non-terminal with two neighbours (NTN$_2$)**: A non-terminal v_i with $1 \leq |\delta^-(v_i)| \leq 2$ as well as $1 \leq |\delta^+(v_i)| \leq 2$ and exactly two adjacent vertices v_j, v_k can be deleted, adding:*

1. *The arc (v_k, v_j) of cost $c_{ki} + c_{ij}$ if $(v_k, v_i) \in A$ and $(v_i, v_j) \in A$.*
2. *The arc (v_j, v_k) of cost $c_{ji} + c_{ik}$ if $(v_j, v_i) \in A$ and $(v_i, v_k) \in A$.*

In the case of two parallel arcs, one of minimal cost is retained.

Proof. If an arc in $|\delta^-(v_i)|$, say (v_j, v_i), is contained in an optimal solution, so is (v_i, v_k), since otherwise v_i and (v_j, v_i) could be deleted to obtain a solution of smaller weight. □

Apart from these four methods, all reduction tests presented for the SAP are newly developed. The first of them can be readily devised by considering that each Steiner arborescence contains by definition directed paths from the root to all further terminals:

***Terminal of in-degree one (TI$_1$)**: Let $t_i \in T^r$. If $\delta^-(t_i) = \{(v_j, t_i)\}$, then (v_j, t_i) can be contracted into t_i.*

Similarly, there can be no path in a Steiner arborescence ending in r since this would imply the existence of a cycle. Therefore, the following reduction method is valid:

***In-root arc (IRA**[4]*)**: All incoming arcs $\delta^-(r)$ of the root can be deleted.*

Finally, one can dispose of those vertices that cannot be part of all optimal solutions:

***Unreachable non-terminal (UNT)**: Let $v_i \in V \setminus T$. If there is no directed path from r to v_i or no directed path from v_i to a $t_j \in T^r$, then v_i can be deleted.*

Proof. If v_i is not contained in any directed path from r, it cannot be contained in any Steiner arborescence. On the other hand, if there is no direct path from v_i to a terminal other than the root, then to any Steiner arborescence S that contains v_i there is one of no higher cost obtained by removing v_i and its ingoing arc and considering the connected component containing r. Hence, there is at least one minimal Steiner arborescence not containing v_i. □

[4]Any resemblance of the abbreviation to existing organizations is purely coincidental.

The UNT test can be implemented by running a breadth-first search starting from r, only considering outgoing arcs for each vertex being processed and finally deleting all unvisited vertices. Thereafter, a breadth-first search using all terminals except for the root as start vertices and only considering incoming arcs is performed. Finally, all vertices that were not visited in the course of the search are deleted.

The successive execution of the above tests in the listed order is denoted **Basic Reduction** (**BR**). All tests other than UNT process each vertex and each arc at most once, but the latter requires executing a breadth-first search. Hence, the overall worst-time complexity is of $\Theta(m)$, assuming that the underlying graph is connected.

Alternative Based Reductions

For the remaining reduction methods to the SAP it will be presupposed that the given instance P_{SAP} is feasible. This assumption is tantamount to the condition that for each non-root terminal $t_i \in T^r$ there is at least one directed path from r to t_i. The latter can be easily validated while performing the UNT test.

Lemma 16. *Let $t_i \in T^r$, $\vec{P}_i$ a (directed) path from r to t_i and a'_i the last arc of $\vec{P}_i$. If*

$$C(\vec{P}_i) \leq \min_{a \in \delta^-(t_i) \setminus \{a'_i\}} c_a \tag{36}$$

holds, then there is at least one minimal Steiner arborescence that contains a'_i.

Proof. Suppose the existence of a Steiner arborescence $S = (V_S, A_S)$ not containing a'_i. Consequently, there is an arc $a''_i \in A_S \cap \delta(t_i)$, $a''_i \neq a'_i$. By removing a''_i from S one obtains an arborescence S_1 rooted in r and an arborescence S_2 rooted in t_i. Denote by $S' = (V_{S'}, A_{S'})$ the subgraph yielded from reconnecting S_1 and S_2 by $\vec{P}_i$ and removing all arcs anti-parallel to arcs of $\vec{P}_i$. Ensuing, to each vertex v_k of S there is a directed path from r to v_k in S'. To render S' cycle-free, which is not necessarily the case, repeatedly remove from each vertex of S' endowed with at least two incoming arcs one of the latter. Since each arc is of non-negative weight, this procedure cannot increase the cost of S'. As soon as for each $v_l \in V_{S'}$ the condition $|\delta^-_{S'}(v_l)| \leq 1$ is satisfied, S' has become a tree.

Since S' additionally contains all terminals, it is a Steiner arborescence. Finally, (36) implies that

$$C(\vec{P}_i) \leq c_{a''_i},$$

so $C(S') \leq C(S)$ can be inferred. □

The affiliated test, denoted by **Root Proximity Terminal** (**RPT**), first runs Dijkstra's algorithm from the root until all terminals are scanned. Next, for each terminal $t_i \in T^r$ it is checked whether the premises of Lemma 16 are satisfied and if so, the corresponding arc is contracted into t_i. The worst-case complexity is dominated by Dijkstra's algorithm and is therefore of $O(m + n \log n)$. Empirically, the run time can be improved by terminating Dijkstra's algorithm as soon as the minimal distance between the root and any scanned vertex is higher than:

$$\max_{i \in \{2,\dots,s\}} \Big\{ \min_{\{a'_1, a'_2\} \subset \delta^-(t_i)} \{\max\{c_{a'_1}, c_{a'_2}\}\} \Big\}. \tag{37}$$

This value can be easily computed in $\Theta(m)$ by checking the two incoming arcs of smallest cost to each terminal.

Lemma 17. *Let $t_i, t_j \in T^r$ such that both $(t_i, t_j) \in A$ and $(t_j, t_i) \in A$. If*

$$c_{(t_i,t_j)} \leq \min_{a \in \delta^-(t_j)} c_a \tag{38}$$

and

$$c_{(t_j,t_i)} \leq \min_{a \in \delta^-(t_i)} c_a, \tag{39}$$

then t_i can be contracted into t_j.

Proof. Let S be a Steiner arborescence containing neither (t_i, t_j) nor (t_j, t_i). Further, denote by $\vec{P}_i$ the directed path in S from r to t_i and by $\vec{P}_j$ the one between r and t_j. Due to S being cycle-free, three cases can be distinguished:

1. $V[\vec{P}_i] \cap V[\vec{P}_j] = \{r\}$,
2. $V[\vec{P}_i] \subset V[\vec{P}_j]$,
3. $V[\vec{P}_i] \supset V[\vec{P}_j]$.

By virtue of symmetry, only the first two cases need to be considered. So suppose that $\vec{P}_i$ and $\vec{P}_j$ have only the root in common. Removing the, unique, arc of $\vec{P}_i$ going into t_i, one can divide S into an arbrorescence S_1 rooted in r, and containing t_j, and an arborescence S_2 rooted in t_i. By inserting the arc (t_j, t_i), which is due to the premises of the claim of not higher cost than the removed arc, a new arborescence S' is obtained and $C(S) \geq C(S')$ holds.
Now suppose that $\vec{P}_i$ is a subpath of $\vec{P}_j$. By removing the arc of $\vec{P}_j$ incident to t_j, the Steiner arborescence S is divided into an arborescence S_1 rooted in r and an arborescence S_2 rooted in t_j. Those can again be incorporated to a Steiner arborescence of equal or smaller cost than S by inserting the arc (t_i, t_j). □

We refer to the test based upon Lemma 17 as **Close Terminals** (**CT**). It can be performed by first computing the shortest incoming arc to each terminal and subsequently checking to each terminal all incoming arcs. Since thereby each arc is processed at most twice, the overall worst-case complexity is of $\Theta(m)$.

Unfortunately, the concept of the bottleneck Steiner distance cannot be directly transferred to the SAP: Consider an arc $(v_i, v_j) \in A$. One might wonder whether, analogously to Lemma 1, this arc can be deleted if there is a directed path from v_i to v_j other than (v_i, v_j) such that the distance between two subsequent terminals (or between a terminal and one of the endpoints of the path) is smaller than $c_{(v_i,v_j)}$. However, this is not true: Assume that there is a Steiner arborescence S containing (v_i, v_j). Removing (v_i, v_j) would segment S into one arborescence S_r rooted in r and one arborescence S_{v_j} rooted in v_j. Reconnecting the trees by a directed path that ends in a vertex of S_{v_j} other than v_j would require to reorientate S_{v_j}, which could lead to additional costs or might even be impossible due to a missing bi-direction. Hence, the bottleneck Steiner distance approach is not valid for the SAP.

However, at least a degenerate, but considerably less potent, version of the bottleneck Steiner distance test holds for the SAP:

Lemma 18. *Let $(v_i, v_j) \in A$. If there is a directed path $\vec{P}$ from v_i to v_j of cost $C(\vec{P}) \leq c_{ij}$, then (v_i, v_j) can be discarded.*

Proof. Suppose that there is a Steiner arborescence S containing the arc (v_i, v_j). By removing (v_i, v_j), S can be divided into an aborescence S_1 rooted in r and containing v_i and and aborescence S_2 rooted in v_j. Reconnect S_1 and S_2 to a subgraph $S' = (V_{S'}, A_{S'})$ by inserting $\vec{P}$ and removing all arcs in A_S anti-parallel to arcs of $\vec{P}$. Thereupon, S' contains to each vertex $v_k \in V_{S'}$ a directed path from r to v_k. If S' is not cycle-free, remove from each vertex $v_k \in V'_{S'}$ of in-degree greater than one in S' one of its incoming arcs. By continuing until each vertex of S' is of in-degree at most one, S' is rendered a Steiner aborescence. Since all arcs are of non-negative weight, removing any of them cannot increase the cost of S'. Therefore, it can be inferred from $C(\vec{P}) \leq c_{ij}$ that $C(S') \leq C(S)$ holds. □

We realize the corresponding test by adapting SDC, see Section 2.2.3, in a straightforward way: Consider each arc $(v_i, v_j) \in A$. First, execute a limited run of Dijkstra's algorithm starting from v_i, ignoring (v_i, v_j) and using outgoing arcs only. Second, initiate the algorithm anew starting from v_j, this time considering exclusively ingoing arcs and ignoring (v_j, v_i). If during this second run, a vertex is labelled that was labelled in the first run, check whether the corresponding path between v_i and v_j is of no higher cost than c_{ij}. In this case, (v_i, v_j) is eliminated.

Bound Based Reductions

Owed to the orientation of paths in the SAP, we adapt a few definitions: Let $v_i, v_j \in V$. Next, let $\underline{\vec{d}}(v_i, v_j)$ be the length of a shortest directed path from v_i to v_j not containing any intermediary terminal. We set $\underline{\vec{d}}(v_i, v_j) := \infty$ if such a path does not exist. For a digraph $D = (V, A)$ we denote by $\overline{G}_D = (\overline{V}_D, \overline{E}_D)$ an undirected graph such that $\overline{V}_D := V$, $\overline{E}_D := \big\{\{v_i, v_j\} \in V \times V \mid (v_i, v_j) \in A \vee (v_j, v_i) \in A\big\}$. Further, we define edge costs $\overline{c}$ such that $\overline{c}_{\{v_i,v_j\}} := \min\{c_{(v_i,v_j)}, c_{(v_j,v_i)}\}$ for $\{v_i, v_j\} \in \overline{E}$ (we set $c_{(v_i,v_j)} := \infty$ if $(v_i, v_j) \notin A$ but $(v_j, v_i) \in A$). Furthermore, we stipulate that all leaves of a Steiner arborescence need to be terminals. Otherwise, due to zero-cost arcs being admissible, a minimum Steiner arborescence might contain non-terminal vertices of degree one, which would thwart the following lemmata. For practical purposes this precondition is of no relevance, since a Steiner vertex that occurs as a leaf in at least one minimum Steiner arborescence can be forthwith discarded.

Having performed the UNT test, one can start from the premise that to each non-terminal vertex there is at least one directed path to a terminal in T^r. These preliminaries allow to define a variation of the Voronoi diagram introduced in Section 1: A **rooted inward Voronoi diagram** to D is a partition $\big\{N(t) \mid t \in T\big\}$ of V such that:

$$N(r) := \{r\}$$

and

$$v \in N(t) \Rightarrow \vec{d}(v, t) \leq \vec{d}(v, t') \text{ for all } t' \in T^r.$$

Note that if the IRA test has been performed, no path corresponding to $\vec{d}(v, t)$ with $v \in V \setminus \{r\}$ and $t \in T^r$ can contain r.

The heretofore considerations already set the stage for implementing a simple reduction criterion:

Lemma 19. *Let $v_i \in V \setminus T$. If there is a minimum Steiner arborescence $S = (V_S, A_S)$ such that $v_i \in V_S$, then $\vec{d}(r, v_i) + \min_{t \in T^r} \underline{\vec{d}}(v_i, t)$ is a lower bound on the cost of S.*

Proof. Assume that the premises of the lemma are satisfied and let S be a minimum Steiner arborescence containing v_i. This assumption implies the existence of a directed path from r to v_i in S of cost at least $d(r, v_i)$ and an additional, arc-disjoint, directed path in S, starting in v_i and leading to a terminal other than r (since otherwise there would be a non-terminal leaf in S). Consequently, it can be inferred that $\vec{d}(r, v_i) + \min_{t \in T^r} \underline{\vec{d}}(v_i, t)$ is indeed a lower bound on the cost of S. □

Given an upper bound, the associated test can be realized by first using Dijkstra's algorithm with source r to obtain $\vec{d}(r, v_i)$ to each $v_i \in V \setminus$

T and second computing the rooted inward Voronoi diagram to obtain $\min_{t\in T^r} \vec{\underline{d}}(v_i,t) = \vec{\underline{d}}(v_i, base(v_i))$ (assuming that the IRA test was performed priorly). If the upper bound is exceeded, v_i can be deleted. The test is denoted by **Prohibitive Non-Terminal (PNT)**.

Based on this rooted inward Voronoi diagram we denote by $inradius(t_i)$ the length of a shortest directed path ending in a terminal $t_i \in T^r$ and having its starting point outside of $N(t_i)$. With this concept at hand, we are in a position to develop a reduction method similar to the BND test for the SPG, see Section 2.2.

Hereinafter, in an attempt to set bounds to a runaway notation, it shall be assumed that for $k \in \{2,...,s\}$ the $inradius(t_k)$ values are non-decreasing (recall that $t_1 = r$). Furthermore, in the remainder of this section it will be assumed that D is endowed with a rooted inward Voronoi diagram, its regions denoted by N, and that the IRA test has been performed. These preliminaries finally enables us to establish two lemmata that allow to examine whether certain specified vertices or arcs can possibly be part of an optimal solution (and to eliminate them if this is not the case).

Lemma 20. *Let $v_i \in V \setminus T$. If there is a minimum Steiner arborescence $S = (V_S, A_S)$ such that $v_i \in V_S$, then*

$$\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=2}^{s-2} inradius(t_q), \tag{40}$$

with $\underline{d}$ denoting the distances in $\overline{G}_D$, is a lower bound on the weight of S.

Proof. Let $v_i \in V \setminus T$ and assume that this vertex is contained in a minimum Steiner tree $S = (V_S, A_S)$. This solution can be easily transferred to a tree $\overline{S}$ in $\overline{G}_D$ by replacing each arc $(v_i, v_j) \in A_S$ by the edge $\{v_i, v_j\}$. We call each edge of $\overline{S}$ such that its corresponding arc in S has its endpoints in different Voronoi regions **induced Voronoi-boundary edge**.

Thereupon, denote an undirected path in $\overline{S}$ between v_i and a $t_j \in T^r$ by P_j and the collection of all such paths by $\mathcal{P}$. Analogously to the proof of Lemma 10 it can be verified that there are at least two edge-disjoint paths in $\mathcal{P}$. Let P_j and P_k be two such paths, with a minimal combined number of induced Voronoi-boundary edges. Moreover, for all $u \in \{2,...,s\} \setminus \{j,k\}$ denote by $\vec{P}_u$ the directed path in S between r and t_u. Further, let $\vec{P}'_u = \vec{P}(v_q, t_u)$ such that $V[\vec{P}(v_q,t_u)] \setminus N(t_u) = \{v_q\}$. Since the Voronoi regions are pairwise arc- (and vertex-) disjoint, the $\vec{P}'_u$ are arc-disjoint as well. Given a directed path $\vec{P}$ in D, denote its undirected equivalent in $\overline{G}_D$ by P, i.e. $\overline{V}_D[P] = V[\vec{P}]$, $\overline{E}_D[P] = \big\{\{v_i,v_j\} \in \overline{E}_D \mid (v_i,v_j) \in A[\vec{P}] \vee (v_j,v_i) \in A[\vec{P}]\big\}$.

Suppose that P_j has an edge in common with a path P'_u. Since S is cycle free this would imply that the subpaths of P_j and P_u up to this edge were identical. For the same reason P_k cannot have an edge in common

with P_u. However, in this case P_j would contain more induced Voronoi-boundary edges than P'_u (to reach t_j which is by definition not in $N(t_u)$). Consequently P_u would have initially been selected instead of P_j.

Following the same line of argumentation, one verifies that likewise P_k cannot have any edge in common with a P'_u. Hence, P_j, P_k and all P'_u are edge-disjoint and therefore:

$$\begin{aligned} C(S) &\geq \overline{C}(P_j) + \overline{C}(P_k) + \sum_{u \in \{2,\dots,s\} \setminus \{j,k\}} C(\vec{P}'_u) \\ &\geq \overline{C}(P_j) + \overline{C}(P_k) + \sum_{q=2}^{s-2} inradius(t_q) \\ &\geq \underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=2}^{s-2} inradius(t_q). \end{aligned}$$

Thereby the claim is established. □

Lemma 21. *Let $(v_i, v_j) \in A$. If there is a minimum Steiner arborescence $S = (V_S, A_S)$ such that $(v_i, v_j) \in A_S$, then L defined by*

$$L := c_{ij} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=2}^{s-2} inradius(t_q) \tag{41}$$

if $v_{i,1} \neq v_{j,1}$ and

$$L := c_{ij} + \min\left\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1})\right\} + \sum_{q=2}^{s-2} inradius(t_q) \tag{42}$$

otherwise, is a lower bound on the weight of S. Again, the $\underline{d}$-distances are stated with respect to $\overline{G}_D$.

Proof. Let $(v_i, v_j) \in A$ and assume that this arc is contained in a minimum Steiner arborescence $S = (V_S, A_S)$. As in the proof to Lemma 20, denote by $\overline{S}$ the corresponding tree in $\overline{G}_D$. Thereupon, denote for each terminal $t_k \in T^r$ the, unique, undirected path not containing the edge $\{v_i, v_j\}$ in $\overline{S}$ to either v_i or v_j by P_k and the collection of all such paths by $\mathcal{P}$.

Analogously to the proof of Lemma 11 it can be verified that there are at least two edge-disjoint paths in $\mathcal{P}$. Let P_k and P_l be two such paths, with a minimal combined number of induced Voronoi-boundary edges. For all $u \in \{2, ..., s\} \setminus \{k, l\}$ define $\vec{P}_u$, $\vec{P}'_u$ and P'_u analogously to the proof of Lemma 20. In the same manner it can be verified that P_k, P_l and all P'_u are edge-disjoint.

In the case of $v_{i,1} \neq v_{j,1}$ the combined cost of P_k and P_l is at least $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1})$, so:

$$\begin{aligned} C(S) &\geq c_{ij} + \overline{C}(P_k) + \overline{C}(P_l) + \sum_{u \in \{2,\ldots,s\} \setminus \{k,l\}} C(\vec{P}'_u) \\ &\geq c_{ij} + \overline{C}(P_k) + \overline{C}(P_l) + \sum_{q=2}^{s-2} inradius(t_q) \\ &\geq c_{ij} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=2}^{s-2} inradius(t_q). \end{aligned}$$

On the other hand, considering that not both P_j and P_k can contain $base(v_i)$ if $base(v_i) = base(v_j)$, the combined cost of P_j and P_k is at least:

$$\min \left\{ \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}) \right\}.$$

Hence:

$$\begin{aligned} C(S) &\geq c_{ij} + \overline{C}(P_k) + \overline{C}(P_l) + \sum_{u \in \{2,\ldots,s\} \setminus \{k,l\}} C(\vec{P}'_u) \\ &\geq c_{ij} + \overline{C}(P_k) + \overline{C}(P_l) + \sum_{q=2}^{s-2} inradius(t_q) \\ &\geq c_{ij} + \min \left\{ \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}) \right\} \\ &\quad + \sum_{q=2}^{s-2} inradius(t_q). \end{aligned}$$

Consequently, the lemma is established. □

The test associated with Lemma 20 and Lemma 21 is denoted by **Bound** (**BND**) test. The required upper bound is computed using the RSP heuristic. As in Section 2.2, the BND test covers the case of an equal lower and upper bound.

Ordering

In consistency with the procedure for the SPG, tests of low run time are generally performed prior to expensive ones. Likewise, all reduction tests are arrayed in a loop that is reiterated as long as a constant proportion (0.5 percent) of arcs or vertices has been eliminated during the last run. Finally, for the BND test an upper bound is only computed during the first run. The final ordering is reported in Figure 6.

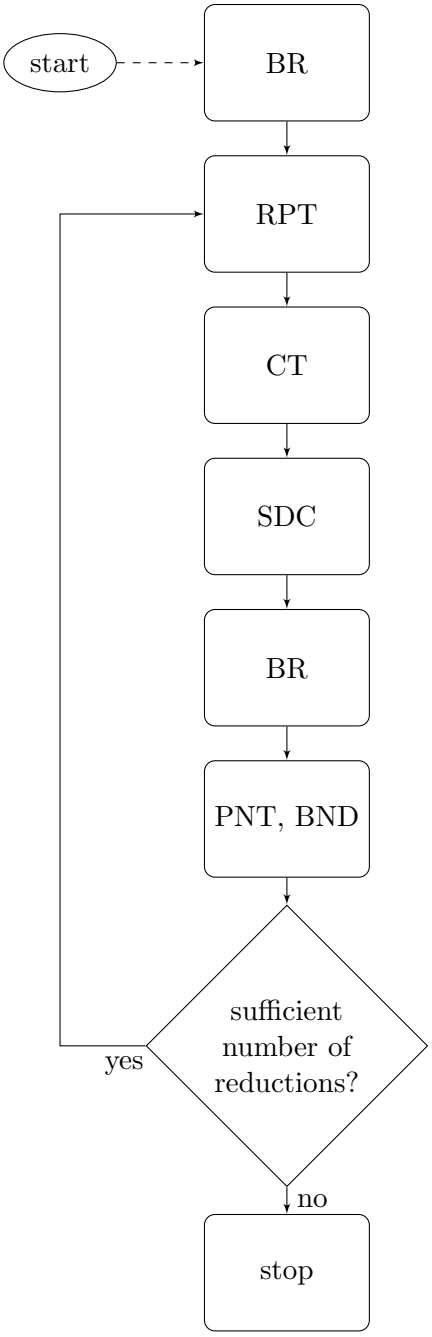


Figure 6: Illustration of the SAP reduction package

3.1.2 Heuristics

The RSPH and RC heuristics can be used for general SAPs with only minor modifications. However, due to the missing bi-direction with equal cost, the VQ heuristic cannot be applied. Also, if the RSPH was used unaltered, the resulting subgraph might have to be reorientated, which might either be impossible due to a missing bi-direction or lead to higher costs. Therefore, we needed to resort to the empirically expensive, original implementation of RSPH described in Section 2.3.1 and apply the following adaptations: First, during the execution of Dijkstra's algorithm starting from the root only outgoing arcs, and for the ones starting from the other terminals only ingoing arcs are considered. Second, if the start vertex is not the root, it is always connected to the root in the first iteration.

3.1.3 Implementation and Computational Results

As each SPG instance to be solved by SCIP-Jack is transformed to an SAP, the branch-and-cut substruction described in Section 2.4 can be used unaltered. The same is true for the implemented propagator, see Section 2.5.

Name	Instances	$\|V\|$	Status	Description
Gene	10	335-602	solved	Sparse, non-bidirectional, instances with $c \equiv 1$. Derived from a genetic application [JKC+00].
Gene2002	9	297-484	solved	

Table 7: Classes of SAP instances.

For the computational experiments, unfortunately only Steiner arborescence problems of small size were available, distributed on two test sets. An overview is provided in Table 7. The largest instance encompasses 602 nodes, 1716 arcs and 86 terminals. SCIP-Jack is able to solve all instances within fractions of a second, not requiring any branching, see Table 8.

			Optimal	
Class	Instances	Solved	∅ Nodes	∅ Time [s]
Gene	10	10	1.0	0.2
Gene2002	9	9	1.0	0.1

Table 8: Computational results for the SAP instances.

Due to the size of the instances and the corresponding very fast solving, we do not deem it useful to provide detailed information on the performance of the primal heuristics. We merely mark that in all but five cases the solution found by RSPH before the actual solving process started was already optimal.

Likewise, only summarized results on the performance of the reduction methods are provided. The reason, besides the low solving time, lies with

Class	Remaining Vertices[%]	Remaining Arcs[%]	∅ Time [s]
Gene	57.13	67.43	0.2
Gene2002	59.06	69.14	0.1

Table 9: Computational results of the SAP reduction package. The percentage of remaining arcs and vertices is stated with respect to the arithmetic mean, the time is given as a shifted geometric mean (with $s = 1$).

the special structure of the instances, which only contain forward arcs of weight one and a relatively high number of terminals. Therefore, only the BR and RPT reductions are effective. The former is accountable for almost 90 percent of the reduced arcs. Notwithstanding the limited applicability of the reduction methods, Table 9 reveals that less than 60 percent of vertices remain after presolving for both the Gene and the Gene2002 test set. Additionally, less than 70 percent of the arcs remain. For none of the instances more than 0.3 seconds of reduction time is required.

3.2 The Node Weighted Steiner Problem

The **node-weighted Steiner tree problem** (**NWSTP**) is a generalization of the Steiner tree problem in graphs in such a way that, in addition to the edges, also the vertices are assigned non-negative weights. The objective is to interconnect all terminals while minimizing the weight summed over both vertices and edges spanned by the corresponding tree. First introduced in [Seg87], it has been the subject of several publications [DHK14, GK99, MR01], although mostly of theoretical nature.

The NWSTP is formally stated by: Given an undirected graph $G = (V, E)$, vertex costs $p : V \to \mathbb{Q}_{\geq 0}$, edge costs $c : E \to \mathbb{Q}_{\geq 0}$, and a set $T \subset V$ of terminals, the objective is to find a tree $S = (V_S, E_S)$ that spans T while minimizing:

$$C(S) := \sum_{e \in E_S} c_e + \sum_{v \in V_S} p_v.$$

3.2.1 Transformation

The key prerequisite for transforming an NWSTP to an SAP is the observation that in a directed tree there cannot be more than one arc going into the same vertex. Otherwise, two directed paths starting in the root would end in the same vertex, hence forming a cycle (in the undirected sense). Exploiting this consideration, we first substitute each edge of a given SPG by two anti-parallel arcs and second add the weight of each vertex to all of its ingoing incident arcs:

Transformation 2 (NWSTP to SAP)**.**
Input: *An NWSTP* $P = (V, E, T, c, p)$
Output: *An SAP* $P' = (V', A', T', c', r')$

1. *Set* $V' := V$, $T' := T$, $A' := \{(v, w) \in V' \times V' : \{v, w\} \in E\}$.
2. *Define* $c' : A' \to \mathbb{Q}_{\geq 0}$ *by* $c'_a = c_{\{v,w\}} + p_w$, *for* $a = (v, w) \in A'$.
3. *Choose a root* $r' \in T'$ *arbitrarily.*
4. ***Return*** (V', A', T', c', r').

Lemma 22 (NWSTP to SAP)**.** *Let* $P = (V, E, T, c, p)$ *be an NWSTP and* $P' = (V', A', T', c')$ *an SAP obtained by applying Transformation 2 on* P. *Denote by* $\mathcal{S}$ *and* $\mathcal{S}'$ *the set of solutions to* P *and* P' *respectively.* $\mathcal{S}'$ *can be bijectively mapped onto* $\mathcal{S}$ *by assigning each* $(V'_{S'}, A'_{S'}) \in \mathcal{S}'$ *a* $(V_S, E_S) \in \mathcal{S}$ *such that*

$$V_S := \left\{v \in V : \ v \in V'_{S'}\right\} \tag{43}$$
$$E_S := \left\{\{v, w\} \in E : \ (v, w) \in A'_{S'} \text{ or } (w, v) \in A'_{S'}\right\}. \tag{44}$$

Furthermore, it holds that:

$$c'(A'_{S'}) + p_{r'} = c(E_S) + p(V_S). \tag{45}$$

Proof. Proving that (43) and (44) form a bijection is equivalent to the procedure in the proof of Lemma 15, since compared to the latter only the weights are altered. Now let $S' \in \mathcal{S}'$ and S be the solution to P obtained by applying the mapping (43) and (44) on S'. To acknowledge (45) one readily observes that for each node of S' except for the root there is exactly one incoming arc. Since for each arc (v, w) of S' it holds that $c'_{(v,w)} = c_{\{v,w\}} + p_w$, one infers that:

$$\sum_{(v,w)\in A'_{S'}} c'_{(v,w)} = \sum_{(v,w)\in A'_{S'}} \left(c_{\{v,w\}} + p_w\right) = \sum_{\{v,w\}\in E_S} c_{\{v,w\}} + \sum_{w\in V_S} p_w - p_{r'},$$

which implies (45). □

The resulting problem is an SAP, which can be handled by SCIP-Jack using the configurations described in the Steiner arborescence section. It should be noted that a transformation of an NWSTP to an SAP was already suggested in [Fis91], but only for negative node weights.

3.2.2 Implementation and Computational Results

In our implementation, reductions are only applied to the transformed SAP. Although we would claim that NWSTP specific reduction techniques, which could be designed similarly to the ones for the PCSPG, would be more effective. Yet, due to limited time and the absence of a large number of, real-world, test instances, we have not developed any problem specific presolving. However, several generic reduction techniques described in [GKM+15] are used. Furthermore, the subsequent results have already been published in the same publication.

Two NWSTP instances derived from a computational biology application published as part of the DIMACS Challenge were tested. The two instances differ drastically in their size: While the first one contains more than 200,000 nodes—55,000 of them terminals—and almost 2.5 million edges, the smaller instance comprises merely 386 nodes, 1477 edges, and 35 terminals.

The size of the first instance proves to be prohibitive for SCIP-Jack. The large number of edges significantly degrades the performance of the preprocessing routines. Additionally, the memory requirements of this instance quickly exceeds the limits of SCIP-Jack when applying the default settings within our heretofore used computational environment. To evaluate the ability of SCIP-Jack to solve this particular instance, a run time of 72 hours was used on a machine with 386 GB memory. After the application of the reduction techniques, the resulting graph contains 187,933 nodes and

986,703 edges. This equates to a 8.6 % and 60.4 % decrease in the number of nodes and edges respectively. SCIP-Jack fails to solve this instance to optimality, but it manages to come as close as 0.0049% or closer to an optimal solution. The much smaller second instance is solved by SCIP-Jack in the root node within 0.1 seconds.

3.3 The Rectilinear Steiner Minimum Tree Problem

The **rectilinear Steiner minimum tree problem** (**RSMTP**) can be stated as follows: Given a number of $k \in \mathbb{N}$ points in the plane, find a shortest tree consisting just of vertical and horizontal line segments and containing all k points. The *RSMTP* is NP-hard, as proven in [GJ77]. Perhaps the most famous of all Steiner variants, it has been the subject of various publications, e.g. [Ema10, HRW92, WWZ00, ZR00]. Furthermore, a generalization of the problem to the d-dimensional case, with natural $d \geq 2$, has been introduced, involving real-world applications in up to eight dimensions, e.g. in cancer research [CSHH$^+$13].

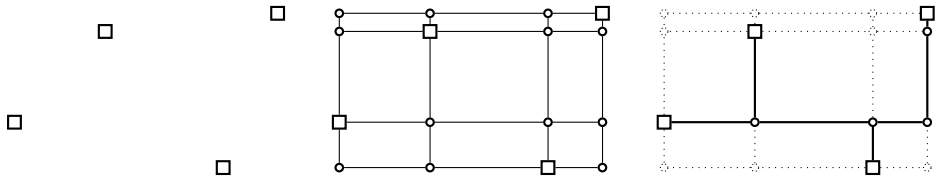
A variant of the RSMTP is the **obstacle-avoiding rectilinear Steiner minimum tree problem** (**OARSMTP**). This problems includes the additional condition that the minimum-length rectilinear tree does not pass through the interior of any specified axis-aligned rectangles, denoted as **obstacles**.

3.3.1 Transformation

Already in 1966, Hanan [Han66] suggested to reduce the RSMTP to the **Hanan grid** obtained by constructing vertical and horizontal lines through each given point of the RSMTP. Thereby, the RSMTP can be reduced to an SPG, since there is at least one solution to an RSMTP that is a subgraph of the grid. In the remainder of this subsection, both this construction and its multi-dimensional generalisation [Sny92] will be exploited in order to adapt the RSMTP to our solver.

Given a d-dimensional, $d \in \mathbb{N}\backslash\{1\}$, RSMTP instance represented by a set of $k \in \mathbb{N}$ points in $\mathbb{Q}^d$, the first step on our way to solving the problem is to construct a d-dimensional Hanan grid. Thereupon, an SPG $P = (V, E, T, c)$ can be built up in a straightforward way: The vertex set V is constructed by assigning each intersection point of the grid a vertex. Analogously, E is obtained by interconnecting the vertices V according to the grid structure, i.e. in such a way that two vertices in V are adjacent if and only if their corresponding intersection points are adjacent in the grid. The set of terminals T consists of all vertices in V corresponding to one of the original k RSMTP points. Finally, the cost of an edge $\{v_i, v_j\} \in E$ is defined by the (Euclidean) distance of the two d-dimensional points in the grid corresponding to v_i and v_j respectively. In [Sny92] it was demonstrated that at least one optimal solution is preserved by this procedure and that the RSMTP can be solved on the resulting SPG. Employing SCIP-Jack to do so, we obtain a solution S. Finally, S is re-transformed to a solution S_{RSMT} to the RSMTP by adding for each edge in S the corresponding line segment (in $\mathbb{Q}^d$) to S_{RSMT}.

SCIP-Jack is easily extended to solve the obstacle-avoiding rectilinear



(a) given points (b) the Hanan grid (c) a possible solution

Figure 7: Illustration of an RSMTP (left), the SPG obtained by the Hanan grid (center) and a solution to the RSMTP (right).

Name	Instances	$\|T\|$	Status	Description
Solids	5	8-20	solved	Three dimensional instances with the points being the vertices of the five platonic solids.
Estein50	15	50	solved	Instances consisting of 50 points each, chosen at random in the unit square [Bea92].
Estein60	15	60	solved	Instances consisting of 60 points each, chosen at random in the unit square [Bea92].
Cancer	14	20-110	unsolved	Instances in up to eight dimensions, used in cancer research to model the progress of a tumor [CSHH$^+$13].

Table 10: Classes of RSMTP instances.

Steiner minimum tree problem, by removing all vertices (along with their incident edges) from the Hanan grid that are located in the interior of an obstacle.

Finally, it must be noted that the related *Euclidean Steiner tree problem*, which differs from the RSMTP insofar as the rectilinear distance is replaced with the Euclidean distance, cannot be handled by our solver. It seems difficult to render this problem solvable for SCIP-Jack, as a clear structure, such as the Hanan grid for the RSMTP, is ostensibly missing.

3.3.2 Implementation and Computational Results

There are a few necessary remarks regarding the implementation in SCIP-Jack. First, by default preprocessing is fully used only for RSMTPs of dimension at most six within our solver. This approach is spawned by the problem specific structure in high dimensions, which proved to be distinctively recalcitrant to the reduction techniques employed by SCIP-Jack, leading to a poor performance of the presolving. Second, we do not expect this simple approach to be competitive with highly specialized solvers, such as *GeoSteiner* [WWZ00] in the cases $d = 2$ and $d = 3$. However, the motivation for our implementation was to provide solutions to RSMTP instances in dimensions $d \geq 4$, since for those there seem to be a lack of specialized solvers. Still, it is not practical to apply the grid transformation for large instances in high dimension, as the number of both vertices and edges increases exponentially with the number of dimensions. Apart

from the described modifications, a transformed RSMTP instance is handled equivalently to a usual SPG instance by SCIP-Jack.

An alternative SPG based solving method for the RSMTP is the full Steiner tree (FST) generation [Zac99]. The concatenation of FSTs, and thus the solving of the underlying RSMTP, can be reduced to an SPG, as done in [Pol04]. Thereby, the tested RSMTP instances could be solved significantly faster than by the well-known, specialized solver GeoSteiner.

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
Solids	5	5	32.0	17.7	–	–
Estein50	15	14	2.6	1408.3	107.0	0.3
Estein60	15	9	1.0	3641.8	12.9	0.3
Cancer	14	11	1.0	25.5	1.0	2.3

Table 11: Computational results for RSMTP instances.

The experimental results, summarized in Table 11, show that all of the Solids instances can be solved, four of them within a few seconds and at the root node. For the Estein50 class only one instance remains unsolved after two hours. Further, 13 of the instances can be solved at the root node. Regarding the related, but in terms of problem size larger, Estein60 test set, a notable decrease of the performance is evident: Only nine instances can be solved to optimality. However, the average gap remains the same. Furthermore, the number of branch-and-bound nodes of the unsolved problems is almost ten times smaller than for the Estein50 class.

Finally, the cancer class manifests the ability of SCIP-Jack to handle and solve instances in up to eight dimensions. Of the 14 instances 11 are solved at the root node. To the best of our knowledge, we are the first to solve those instances to optimality. Of the solved instances seven finish within 10 seconds and none takes longer than eight minutes. However, the instance 11_8D demonstrates the limits of the grid approach: The instance is transformed into a problem containing almost five million nodes and 65 million edges, a scale that is prohibitive for employing SCIP-Jack on a modest machine. As a result, SCIP-Jack quickly runs out of memory while solving this instance. Similarly, for the 12_8D problem, encompassing more than 12 million edges, SCIP-Jack stops short of computing a dual bound within the time limit, since it spends 98 percent of the time in the local heuristic (VQ).

3.4 The Prize Collecting Steiner Tree Problem

In contrast to the classical Steiner tree problem, the required tree for the **prize-collecting Steiner tree problem** (**PCSTP**) needs only to span a (possibly empty) subset of the terminals. However, a non-negative penalty is charged for each terminal not contained in the tree. Hence, the objective is to find a tree of minimum weight, given by both the sum of its edge costs and the penalties of all terminals not spanned by the tree. For a profound discussion on the PCSTP that details real-world applications and moreover introduces a specialized solver see [Lju04].

A formal definition of the problem is stated as: Given an undirected graph $G = (V, E)$, edge-weights $c : E \to \mathbb{Q}_+$, and node-weights $p : V \to \mathbb{Q}_{\geq 0}$, a tree $S = (V_S, E_S)$ in G is required such that

$$C(S) := \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \tag{46}$$

is minimized.

Before discussing the prize-collecting Steiner tree problem, we introduce a variation, the **rooted prize-collecting Steiner tree problem** (**RPCSTP**), that incorporates the additional condition that one distinguished node r, denoted as the **root**, must be part of every feasible solution to the problem. Hereinafter we assume $p_r = 0$.

3.4.1 Transformation

An RPCSTP can be transformed into an SAP as follows:

Transformation 3 (RPCSTP to SAP)**.**
Input: *An RPCSTP* $P = (V, E, p, r)$
Output: *An SAP* $P' = (V', A', T', c', r')$

1. *Set* $V' := V$, $A' := \{(v, w) \in V' \times V' \mid \{v, w\} \in E\}$, $r' := r$ *and* $c' : A' \to \mathbb{Q}_{\geq 0}$ *with* $c'_a = c_{\{v,w\}}$ *for* $a = (v, w) \in A'$.

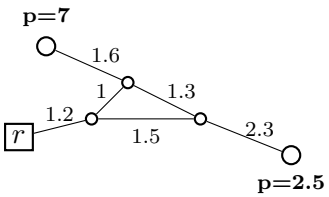
2. *Denote the set of all* $v \in V$ *with* $p_v > 0$ *by* $T = \{t_1, ..., t_s\}$. *For each node* $t_i \in T$ *a new node* t'_i *and an arc* $a = (t_i, t'_i)$ *with* $c'_a = 0$ *is added to* V' *and* A' *respectively.*

3. *Add arcs* (r', t'_i) *for each* $i \in \{1, ..., s\}$ *and set their respective weight to* p_{t_i}.

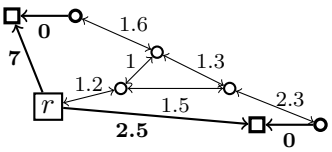
4. *Define the set of terminals* $T' := \{t'_1, ..., t'_s\} \cup \{r\}$.

5. ***Return*** (V', A', T', c', r').

A graphic depiction of the transformation is provided in Figure 8: On the left hand side an RPCSTP instance with a root r and two vertices of positive weight is illustrated. The associated SAP can be found on the right hand side. Thereupon, a (single) solution to both the RPCSTP and the associated SAP is visualised in Figure 9.

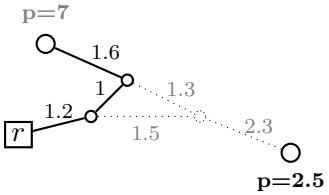


(a) RPCSTP instance

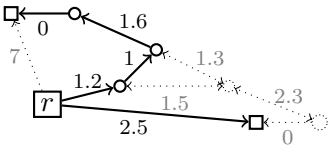


(b) transformed SAP instance (modifications in **bold**)

Figure 8: Illustration of an RPCSTP instance with root r (left) and the equivalent SAP obtained by Transformation 3 (right). Terminals are drawn as squares and Steiner vertices as circles (with those of positive weight enlarged).



(a) RPCSTP solution



(b) SAP solution

Figure 9: Illustration of a solution to the RPCSTP instance of Figure 8 (left) and the equivalent solution to the corresponding SAP (right).

Having performed Transformation 3, one ends up with an SAP P' that is endowed with exactly two incoming arcs, $(t_i, {t_i}')$ and (r', t'), for each terminal t'_i. To allow a bijection between the respective solution sets of P and P', each solution $S' = (V'_{S'}, A'_{S'}) \in P'$ that contains t_i should also contain $(t_i, {t_i}')$, more succinctly:

$$\forall i \in \{1, ..., s\} : t_i \in V'_{S'} \Longrightarrow (t_i, {t_i}') \in A'_{S'}. \tag{47}$$

Condition (47) is satisfied by all optimal solutions to P' and one can easily modify each feasible solution accordingly, concomitantly improving its objective value: Let $S' = (V'_{S'}, A'_{S'})$ be a solution to P' such that there is a $t_i \in V'_{S'}$ with $(t_i, {t_i}') \notin A'_{S'}$. Since ${t_i}'$ is a terminal of in-degree two in the SAP P', it can be forthwith inferred that $(r, {t_i}') \in A'_{S'}$. By construction, see

Transformation 3, it holds that $c_{(t_i,t_i')} = 0$ and $c_{(r,t_i')} > 0$. Thus, removing (r, t_i') and simultaneously inserting (t_i, t_i') one obtains a new tree $\tilde{S}'$ from S' that is of smaller cost than its predecessor. The latter also implies that S' is not optimal.

As another note on condition (47) we can readily infer that it implies first:

$$(t_i, t_i') \in A'_{S'} \Leftrightarrow t_i \in V_S, \tag{48}$$

since t_i is obviously incident to (t_i, t_i') and second

$$(r', t_i') \in A'_{S'} \Leftrightarrow t_i \notin V_S, \tag{49}$$

since t_i', being a terminal, needs to be of in-degree exactly one in every feasible solution.

Lemma 23 (RPCSTP to SAP)**.** *Let $P' = (V', A', T', c')$ be an SAP obtained from an RPCSTP $P = (V, E, c, p)$ by applying Transformation 3. Denote by $\mathcal{S}$ and $\mathcal{S}'$ the set of solutions to P and P', satisfying condition* (47)*, respectively. $\mathcal{S}'$ can be mapped bijectively onto $\mathcal{S}$ by assigning each $(V'_{S'}, A'_{S'}) \in \mathcal{S}'$ a $(V_S, E_S) \in \mathcal{S}$ satisfying*

$$V_S := \{v \in V \mid v \in V'_{S'}\} \tag{50}$$
$$E_S := \{\{v, w\} \in E \mid (v, w) \in A'_{S'} \text{ or } (w, v) \in A'_{S'}\}. \tag{51}$$

The mapping preserves the objective value.

Proof. To acknowledge that (50) and (51) constitute a mapping $\mathcal{S}' \to \mathcal{S}$ let $T = \{t_1, ..., t_s\}$ and $T' := \{t'_1, ..., t'_s\}$ as defined in Transformation 3. By definition, each solution to P' consists of directed paths from the root to all terminals. Those paths either contain the arc (t_i, t_i') or consist of the single arc (r', t_i'). Utilizing these observations, one can dichotomize the mapping described by (50) and (51) into two parts: First, all arcs (r', t_i') and (t_i, t_i') are removed, resulting in a remaining solution consisting of directed paths from r' to a subset of T. Second, all arcs are replaced by their corresponding edges, yielding a solution to the original RPCSTP P.

To prove the one-to-one correspondence, additionally let $S = (V_S, E_S) \in \mathcal{S}$.

Surjective. To validate that (50) and (51) represent a surjection, a solution $S' = (V'_{S'}, A'_{S'}) \in \mathcal{S}'$ is constructed that is mapped to S. Initially, set $V'_{S'} := V_S$ and $A'_{S'} := \emptyset$. Analogously to the proof of Lemma 15, add for each edge in E_S an arc to $A'_{S'}$ in such a way that finally there is for each $v' \in V'_{S'}$ a directed path from r' to v'. Thereafter, for each $i \in \{1, ...s\}$ set $a_i := (t_i, t_i')$ if $t_i \in V_S$ and $a_i := (r', t_i')$ otherwise and add a_i to $A'_{S'}$. Consequently, $S' := (V'_{S'}, A'_{S'})$ is a solution to P' and by applying (50) and (51), we obtain S.

Injective. To demonstrate that S' is the only solution to P' mapped to S, define the set of all arcs of P' corresponding to the edges of P as $A := \{(v,w) \in A' \mid \{v,w\} \in E\}$ and accordingly $A_{S'} := A'_{S'} \cap A$. Note that all arcs incident to terminals are designated by V_S as demonstrated subsequently to (47) (and epitomized in (48) and (49)). This implies that $A'_{S'}$ is already determined by $A_{S'}$. Now let $\tilde{S}' = (V'_{\tilde{S}'}, A'_{\tilde{S}'}) \in \mathcal{S}'$, $\tilde{S}' \neq S'$. Consequently, there is at least one arc $(v,w) \in A_{\tilde{S}'}$ such that $(v,w) \notin A_{S'}$ and $(w,v) \notin A_{S'}$ and therefore is $\tilde{S}'$ not mapped to S.
Finally, using the above notation one observes that:

$$\sum_{a \in A'_{S'}} c'_a = \sum_{a \in A_{S'}} c'_a + \sum_{a \in A'_{S'} \setminus A_{S'}} c'_a = \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v, \tag{52}$$

so the costs of S' and S are equal. □

Transformation 3 can be extended to cover the PCSTP in two steps: First, an artificial root node r' is added and the transformation is performed. Second, arcs (r', t_i) with cost zero are added. They connect the artificial root with all terminals of the original problem and allow to choose a root for the solution. To this end, only one of these arcs can be part of a feasible solution. This condition is enforced by the following constraint:

$$\sum_{a \in \delta^+(r'), c'_a = 0} y_a = 1. \tag{53}$$

Furthermore, to allow a bijection between the original and the transformed problem, for all t_i included in a solution the arc (r', t_i) with the smallest index i is required to be part of the solution. This condition can be expressed by using the following class of constraints:

$$\sum_{a \in \delta^-(t_j)} y_a + y_{(r',t_i)} \leq 1 \quad i = 1, ..., s;\ j = 1, ..., i-1. \tag{54}$$

An SAP that incorporates the conditions (47), (53) and (54) is henceforth referred to as **root constrained Steiner arborescence problem** (**rcSAP**). The constraints (53) and (54) can be incorporated into the cut-formulation (Formulation 5) without further alterations and each solution can be modified in order to meet condition (47). Although additional $\frac{s(s-1)}{2}$ constraints have to be introduced to satisfy (54), the solving time is considerably reduced, since concomitantly a plethora of symmetric solutions is excluded. It should not go unnoticed that an approach to eliminate symmetric solutions in the context of the PCSTP (although for a different formulation) was already applied in [LWP+06].

Transformation 4 (PCSTP to rcSAP)**.**
Input: *A PCSTP* $P = (V, E, c, p)$
Output: *An rcSAP* $P' = (V', A', T', c', r')$

1. *Add a vertex v_0 to V and set $r := v_0$.*
2. *Apply Transformation 3 to obtain $P' = (V', A', T', c', r')$.*
3. *Add arcs $a = (r', t_i)$ with $c'_a := 0$ for each $t_i \in T$.*
4. *Add constraints (53) and (54).*
5. ***Return*** (V', A', T', c', r').

Figure 10 illustrates Transformation 4 for a simple example instance. Also for the PCSTP a bijection between the original and the transformed solution space can be obtained:

Lemma 24 (PCSTP to rcSAP)**.** *Let $P = (V, E, c, p)$ be a PCSTP and $P' = (V', A', T', c', r')$ the corresponding rcSAP obtained by applying Transformation 4. Denote by $\mathcal{S}$ and $\mathcal{S}'$ the sets of solutions to P and P' respectively. Each solution $(V'_{S'}, A'_{S'}) \in \mathcal{S}'$ can be mapped to a solution $(V_S, E_S) \in \mathcal{S}$ defined by:*

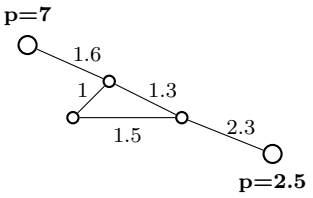
$$V_S := \{v \in V : \ v \in V'_{S'}\}, \tag{55}$$
$$E_S := \{\{v, w\} \in E : \ (v, w) \in A'_{S'} \text{ or } (w, v) \in A'_{S'}\}, \tag{56}$$

establishing a bijection. The objective value is preserved by the mapping.

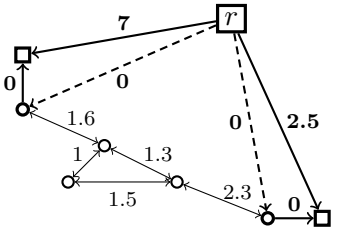
Proof. Likewise to the proof of Lemma 23, one observes that (55) and (56) constitute a mapping $\mathcal{S}' \to \mathcal{S}$. Let $S = (V_S, E_S) \in \mathcal{S}$ and $T = \{t_1, ..., t_s\}$ defined as in Transformation 4.
Surjective. To demonstrate that the mapping induced by (55) and (56) is a surjection, a solution $S' = (V'_{S'}, A'_{S'}) \in \mathcal{S}'$ mapped to S is constructed: Initially, define $V'_{S'} := V_S$, $A'_{S'} := \{(r, t_{i_0})\}$, with $i_0 := \min\{i \mid t_i \in V'_{S'}\}$. Thereby it is ensured that the constraints (54) are satisfied. Condition (53) is likewise met, since no additional arcs from the root to non-terminal vertices will be added. Next, extend $A'_{S'}$ analogously to the proof of Lemma 23. The so constructed $S' := (V'_{S'}, A'_{S'})$ is a solution to P' and by applying (55) and (56) S is obtained.
Injective. Parallel to the proof of Lemma 23 it can be shown that for a solution $\tilde{S}' \neq S'$ to P' there must be at least one arc $(v, w) \in A_{\tilde{S}'}$ such that $(v, w) \notin A_{S'}$ and $(w, v) \notin A_{S'}$ with A defined as in the proof of Lemma 23. Therefore, it follows that $\tilde{S}'$ is not mapped to S.

The equality of the solution values of S and S' can be validated by (52). □

When the amount of terminals is high even after the presolving, enforcing condition (54) by adding the corresponding constraints (which are of quadratic number) might become prohibitive. However, dropping them still allows for a bijection between the solutions to the original and transformed



(a) PCSTP instance



(b) transformed rcSAP instance (modifications in **bold**, terminals drawn as squares)

Figure 10: Illustration of a PCSTP instance and the associated rcSAP obtained by Transformation 4 (right). Of the dashed arcs only one can be selected.

problem. This bijection is rendered possible by the following equivalence relation:

Let $\mathcal{S}'$ be the set of all solutions to an rcSAP P' obtained from a PCSTP P by applying Transformation 4. Thereupon, $S'_1, S'_2 \in \mathcal{S}'$ are defined to be equivalent if and only if the corresponding solutions S_1 and S_2 to P are identical. Consequently, there is a one-to-one correspondence between the quotient space of $\mathcal{S}'$ and the set of solutions to P

3.4.2 Reductions

For the PCSTP several reduction techniques have been proposed in the literature. However, they are neither as sophisticated nor as exhaustive as their counterparts for the SPG. We strive to lessen this gap with a series of novel reduction techniques, introduced (along with several known ones) in this section. Customarily, it will be presupposed that a PCSTP $P_{PC} = (V, E, c, p)$ with a set of terminals $T := \{v \in V \mid p_v > 0\} \neq \emptyset$ is given. Furthermore, we postulate that when contracting an edge $\{v_i, v_j\} \in E$ with exactly one endpoint being a terminal, the edge is contracted into this terminal, if not specified otherwise.

Although the reduction techniques presented in this section are formulated for the PCSTP, it should be noted that all of them can be easily applied to the RPCSTP: Consider an RPCSTP instance (V, E, c, p, r). By setting $p_r := c(E) + p(V)$ it can be transformed to an equivalent PCSTP. While setting the new prize of the root to $p(V)$ would be sufficient to obtain an equivalent (unrooted) prize-collecting instance, the chosen prize aids to satisfy more reduction conditions, as will become evident in the following.

Basic Reductions

Once again, the straightforward reduction techniques are very reminiscent of those for the SPG:

Non-terminal of degree one ***(NTD_0)****: A non-terminal of degree zero can be deleted.*

Non-terminal of degree one ***(NTD_1)****: A non-terminal of degree one and its incident edge can be deleted.*

Non-terminal of degree two ***(NTD_2)****: A non-terminal v_i of degree two and its incident edges $\{v_i, v_j\}$, $\{v_i, v_k\}$ can be substituted by a single edge $\{v_j, v_k\}$ with $c_{jk} := c_{ij} + c_{ik}$. In the case of two parallel edges, one of lowest cost is retained.*

Terminal of degree zero*: (**TD_0**) If there are terminals $t_i, t_j \in T$, such that $\delta(t_i) = \emptyset$ and $p_{t_i} \leq p_{t_j}$, then t_i can be deleted from V (and p_{t_i} has to be added to the solution value of each feasible solution to P_{PC}).*

Proof. Since the only solution that contains t_i (i.e. $\{t_i\}$) is of higher or equal cost than the feasible solution $\{t_j\}$, there is at least one optimal solution that does not contain t_i. □

Terminal of degree one ***(TD_1)****: If there are terminals $t_i, t_j \in T$, such that $\delta(t_i) = \{e_l\}$, $p_{t_i} \leq p_{t_j}$ and additionally*

a) $p_{t_i} \leq c_{e_l}$, then t_i and e_l can be discarded;

b) $p_{t_i} > c_{e_l}$, then t_i and $e_l = \{t_i, v_k\}$ can be discarded, concurrently setting $p_{v_k} := p_{v_k} + p_{t_i} - c_{e_l}$.

In case a) p_{t_i} and in case b) c_{e_l} has to be added to the objective value of each feasible solution to P_{PC} to obtain the value of the corresponding original solution.

Proof. Since $p_{t_i} \leq p_{t'}$, $\{t_i\}$ cannot be the unique optimal solution.
a) Each solution which contains e_l (and t_i) can be reduced to a solution of not higher cost by removing e_l as well as t_i. Hence, there is at least one optimal solution containing neither e_l nor t_i. Since in the transformed problem, t_i cannot be part of any solution, p_{t_i} has to be added to the objective value.
b) Each solution containing v_k but not t_i can be extended to a solution of smaller cost by adding e_l and t_i. So each optimal solution is preserved by the operation. To demonstrate that the stated relation between the respective costs hold, denote by $P'_{PC} = (V', E', c', p')$ the problem obtained from P_{PC} by removing t_i and e_l and setting $p'_{v_k} := p_{v_k} + p_{t_i} - c_{e_l}$. Let $S' = (V'_{S'}, E'_{S'})$ be

a solution to P'_{PC} not containing v_k and denote the corresponding solution to P_{PC} by $S = (V_S, E_S)$. Thereupon:

$$\begin{aligned} C'(S') &= \sum_{e \in E'_{S'}} c'_e + \sum_{v \in V' \setminus V'_{S'}} p'_v \\ &= \sum_{e \in E'_{S'}} c'_e + \sum_{v \in V' \setminus (V'_{S'} \cup \{v_k\})} p'_v + p_{v_k} + p_{t_i} - c_{e_l} \\ &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v - c_{e_l} = C(S) - c_{e_l}. \end{aligned}$$

If, on the other hand, v_k is contained in S', then S comprises both e_l and t_i, hence:

$$\begin{aligned} C'(S') &= \sum_{e \in E'_{S'}} c'_e + \sum_{v \in V' \setminus V'_{S'}} p'_v \\ &= \sum_{e \in E_S} c_e - c_{e_l} + \sum_{v \in V \setminus V_S} p_v = C(S) - c_{e_l}. \end{aligned}$$

Hence, the relation $C'(S') = C(S) - c_{e_l}$ is established. □

***Terminal of degree two (TD$_2$)**: Let $t_i, t_j \in T$, such that $\delta(t_i) = \{\{t_i, v_k\}, \{t_i, v_l\}\}$ and $p_{t_i} \leq \min\{c_{\{v_k,t_i\}}, c_{\{t_i,v_l\}}, p_{t_j}\}$. If a new edge $\{v_k, v_l\}$ of cost $c_{\{t_i,v_k\}} + c_{\{t_i,v_l\}} - p_{t_i}$ is added, $\{v_k, t_i\}, \{v_l, t_i\}$ and t_i can be deleted. For a solution S' to the so transformed problem P'_{PC} and the corresponding solution S to the original problem the relation $C'(S') + p_{t_i} = C(S)$ holds.*

Proof. If t_i is of degree one in a solution, then both t_i and its incident edge in this solution can be removed to obtain a solution of smaller or equal cost. On the other hand, if t_i is of degree zero in a solution, then it is of equal or higher cost than the feasible solution $\{t_j\}$. Conclusively, if t_i is contained in an optimal solution, there is at least one optimal solution in which t_i is of degree two. If a solution S' to P'_{PC} contains the new edge $\{v_k, v_l\}$, the corresponding solution S to P_{PC} contains both original edges of, summed, cost $c_{\{t_i,v_k\}} + c_{\{t_i,v_l\}}$, hence $C'(S') + p_{t_i} = C(S)$. On the other hand, if $\{v_k, v_l\}$ is not contained in S', then p_{t_i} is not contained in S and therefore $C'(S') + p_{t_i} = C(S)$ holds as well. □

Both NTD$_1$ and NTD$_2$ were already proposed in [LR04]. However, to the best of our knowledge, the tests TD$_0$, TD$_1$ and TD$_2$ have not been described in the literature yet: TD$_1$ b) and TD$_2$ were suggested in [Uch06], but without the condition $\exists t_j : p_{t_j} \geq p_{t_i}$. The absence of the latter, or a comparable condition, renders both tests erroneous, since a unique optimal solution consisting only of t_i could then be discarded. We denote by **Degree-Test (DT)** the successive execution of those five tests. Since in each of them

only vertices of maximal degree two are checked, the worst-case complexity of DT is of $O(n)$, assuming that deleting or inserting an edge can be realized in $O(1)$.

Furthermore, noting that a PCSTP graph is not necessarily connected, we suggest the following simple test:

Unreachable non-terminal (UNT): *Each non-terminal vertex that cannot be connected by a path to any terminal can be removed along with all incident edges.*

We realize this test in linear time (with respect to the number of nodes and edges) by running a breadth-first search on the transformed graph obtained by Transformation 4 and deleting all vertices (and incident edges) that are not visited.

Alternative Based Reductions

The pivotal concept of the bottleneck Steiner distance which gives rise to strong reduction techniques for the SPG fails with the PCSTP in its original form. The reason is that a terminal need not be part of any optimal solution, so it cannot a priori be used as the endpoint of an alternative path. However, a redefinition of the bottleneck Steiner distance has been proposed in the context of the PCSTP [Uch06]:

Let $v_i, v_j \in V$ be two distinct vertices, $\mathcal{P}(v_i, v_j)$ the set of all simple paths between v_i and v_j and $P \in \mathcal{P}(v_i, v_j)$. Furthermore, denote by $P(v_k, v_l)$ the subpath between two vertices in P. The **local prize-collecting Steiner distance** of $P(v_k, v_l)$ is defined as

$$sd_{loc}(P(v_k, v_l)) = \sum_{e \in E[P(v_k, v_l)]} c_e - \sum_{v \in V[P(v_k, v_l)] \setminus \{v_k, v_l\}} p_v. \tag{57}$$

Built upon this definition, the **prize-collecting Steiner distance** of (the whole path) P is:

$$sd(P) = \max_{v_k, v_l \in V[P]} sd_{loc}(P(v_k, v_l)). \tag{58}$$

Finally, the **bottleneck prize-collecting Steiner distance** between v_i and v_k can be defined as

$$s_{pc}(v_i, v_k) = \min\{sd(P) \mid P \in \mathcal{P}(v_i, v_k)\}. \tag{59}$$

The two most salient tests spawned by the Steiner bottleneck distance in the context of the SPG find their equivalent for the PCSTP:

Lemma 25. *Every edge $\{v_i, v_j\} \in E$ with $c_{ij} > s_{pc}(v_i, v_j)$ can be discarded.*

Lemma 26. *A non-terminal vertex v_i is of degree at most two in a least one minimum Steiner tree if for each set Δ, with $|\Delta| \geq 3$, of vertices adjacent to v_i it holds that: the (summed) cost of all edges $\delta(v_i) \cap \delta(\Delta)$ is not less than the weight of a minimum spanning tree for the network $(\Delta, \Delta \times \Delta, s_{pc})$.*

Both Lemma 25 and Lemma 26 were formulated and proved in [Uch06]. In the same publication it was demonstrated that computing the bottleneck prize-collecting Steiner distance is NP-hard and the application of heuristics was suggested. However, no information was provided on how to actually proceed in order to obtain those distances. Thereupon, we propose two novel tests to calculate an upper bound on the bottleneck prize-collecting Steiner distance:

First, an adaptation of the SDC test, see Section 2.2.3 is employed. Only one addition to the original test is necessary: To eliminate an edge $\{v_i, v_j\}$, for an alternative path P between v_i and v_j containing exactly one interior terminal t_k, the inequality $C(P) - p_{t_k} \leq c_{\{v_i,v\}}$ needs to be satisfied as well. In the context of the (R)PCSTP we likewise denote this test by **SDC**.

For the second test some spadework is necessary. We wish to obtain for a given $\alpha \in \mathbb{N}$ to each terminal its α $\underline{d}$-nearest terminals. Unfortunately, Duin's nearest terminals algorithm utilized in Section 2.2 merely allows to compute the $\underline{d}$-nearest terminals $v_{i,1}, v_{i,2}, ...$ to a non-terminal v_i. Remedy is provided by the following, novel, lemma:

For ease of presentation, we subsequently assume for each $\{v_j, v_k\} \in \delta\big(N(t_i)\big)$ that $v_j \in N(t_i)$ holds.

Lemma 27. *Let $\alpha \in \mathbb{N}, 1 \leq \alpha < |T|$, $t_i \in T$ and assume that for each $v_j \in V \setminus T$ the $\alpha + 1$ $\underline{d}$-nearest terminals $v_{j,1}, v_{j,2}, ..., v_{j,\alpha+1}$ exist and their $\underline{d}$-distances to t_i are available. Thereupon, setting $t_i^{(0)} := t_i$, the remainder of the $\alpha+1$ $\underline{d}$-nearest terminals $t_i^{(1)}, ..., t_i^{(a)}$ to t_i can be computed as follows: Set $t_i^{(1)} := v_{k,1}$ with k such that:*

$$\underline{d}(t_i, v_{k,1}) = \min_{\{v_j,v_k\} \in \delta(N(t_i))} \{d(t_i, v_j) + c_{jk} + d(v_k, v_{k,1})\}. \tag{60}$$

Having computed $t_i^{(1)}, ..., t_i^{(r)}$ $(r < \alpha)$, we set $t_i^{(r+1)} := v_{k,l}$ with k, l such that:

$$\underline{d}(t_i, v_{k,l}) = \min_{\{v_j,v_k\} \in \delta(N(t_i))} \big\{d(t_i, v_j) + c_{jk} + \min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., r+2 : v_{k,l} \neq t_i^{(q)}, q = 0, ..., r\}\big\}. \tag{61}$$

Proof. First, note that both of the minima attained in (60) and (61) correspond to a path. Next, let $t_{min}^{(1)} \in T \setminus \{t_i\}$ such that $\underline{d}(t_i, t_{min}^{(1)})$ is minimal. The corresponding path between t_i and $t_{min}^{(1)}$ contains an edge $\{v_j, v_k\} \in \delta(N(t_i))$ and since the path is of minimum $\underline{d}$-length its cost is $d(t_i, v_j) +$

$c_{jk} + d(v_k, v_{k,1})$. Hence:

$$\min_{\{v_j,v_k\}\in\delta(N(t_i))}\{d(t_i, v_j) + c_{jk} + d(v_k, v_{k,1})\} \leq \underline{d}(t_i, t_{min}^{(1)}).$$

Furthermore, for $\{v_j, v_k\} \in \delta(N(t_i))$ let P be a path consisting of a shortest path between t_i and v_j, the edge $\{v_j, v_k\}$ and a shortest path between v_k and $v_{k,1}$. Since v_j lies in $N(t_i)$ and $v_{k,1}$ is a $\underline{d}$-nearest terminal to v_k, the path P contains no intermediary terminal vertices. Consequently it is of cost at least $\underline{d}(t_i, v_{k,1})$, so:

$$\min_{\{v_j,v_k\}\in\delta(N(t_i))}\{d(t_i, v_j) + c_{jk} + d(v_k, v_{k,1})\} \geq \underline{d}(t_i, v_{k,1}) \geq \underline{d}(t_i, t_{min}^{(1)}).$$

The validity of the claim for $r > 1$ can be demonstrated similarly: Denote by $t_{min}^{(r)}$ the r'th $\underline{d}$-nearest terminal to t_i, if such a terminal exists. Let $r \in \mathbb{N}, 1 < r < \alpha$ and assume that there is a $t_{min}^{(r+1)}$. Additionally, let P be a path corresponding to $\underline{d}(t_i, t_{min}^{(r+1)})$. This path contains an edge $\{v_j, v_k\} \in \delta(N(t_i))$ and can therefore be dissected into the subpath $P(t_i, v_j)$, the edge $\{v_j, v_k\}$ and the subpath $P(v_k, t_{min}^{(r+1)})$. The first subpath is of length $d(t_i, v_j)$, since otherwise it could be substituted by a shorter one. The second subpath is of length at least $\min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., r+2 : v_{k,l} \neq t_i^{(q)}, q = 0, ..., r, \}$, because this is the minimum length between v_k and a terminal other than $t_i^{(0)}, t_i^{(1)}, ..., t_i^{(r)}$ and without intermediary terminals. Note that v_k itself can be a terminal, so just scrutinizing all $v_{k,l}$ with $l \in \{1, ..., r+1\}$ is not sufficient, since these $r+1$ terminals could be identical to $t_i^{(0)}, t_i^{(1)}, ..., t_i^{(r)}$. This discussion implies:

$$\begin{aligned}\min_{\{v_j,v_k\}\in\delta(N(t_i))} &\big\{d(t_i, v_j) + c_{jk} \\ &+ \min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., r+2 : v_{k,l} \neq t_i^{(q)}, q = 0, 1, ..., r\}\big\} \\ &\leq \underline{d}(t_i, t_{min}^{(r+1)}).\end{aligned}$$

To see the converse, note that each path P associated with the set in (61) is composed of the components $P(t_i, v_j)$, $\{v_j, v_k\}$ and $P(v_k, v_{k,l})$, for some l such that $v_{k,l} \notin \{t_i^{(0)}, t_i^{(1)}, ..., t_i^{(r)}\}$. By construction none of these components contain intermediary terminals, thus the same holds for P. Hence, $C(P) \geq \underline{d}(t_i, v_{k,l})$ and therefore:

$$\begin{aligned}\min_{\{v_j,v_k\}\in\delta(N(t_i))} &\big\{d(t_i, v_j) + c_{jk} \\ &+ \min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., r+2 : v_{k,l} \neq t_i^{(q)}, q = 0, 1, ..., r\}\big\} \\ &\geq \underline{d}(t_i, t_{min}^{(r+1)}).\end{aligned}$$

Consequently, the lemma is established. □

Corollary 28. *Let $\alpha \in \mathbb{N}$, $0 < \alpha < |T|$. The α $\underline{d}$-nearest terminals to each terminal can be computed with worst-case complexity of $O(m + n \log n)$ if α is considered a constant.*

Proof. Compute to each non-terminal the $\alpha + 1$ $\underline{d}$-nearest terminals, in a total of $O(m + n \log n)$. Subsequently, traverse each Voronoi-boundary edge $\{v_i, v_j\} \in E$, with $t_k := base(v_i)$ and $t_l := base(v_j)$, and update the $\underline{d}$-nearest terminals $t_k^{(1)}$ and $t_l^{(1)}$ as well as the corresponding $\underline{d}$-distances according to (60). Thereafter, successively traverse all Voronoi-boundary edges $\{v_i, v_j\} \in E$ for $\alpha - 1$ additional times, updating in each iteration z both the values $t_k^{(z)}$ and $t_l^{(z)}$ and the corresponding $\underline{d}$-distances according to (61). In the z'th iteration for each edge at most z $\underline{d}$-nearest terminals have to be checked. As a is assumed to be a constant and $z \leq \alpha$ holds, the worst-case complexity for all iterations is of $O(m)$. In total this amounts to the overall bound of $O(m + n \log n)$. □

Having established Lemma 27, we are in a position to propose a modification of the SD test introduced in Section 2.2.3: As with the SPG, we compute the four $\underline{d}$-nearest terminals to each non-terminal v_i (or as many as exist). For each terminal t_i we compute, three $\underline{d}$-nearest terminals $t_i^{(1)}, t_i^{(2)}, t_i^{(3)}$. Additionally, we determine a further terminal $t_i^{(4)}$ that is not necessarily a fourth $\underline{d}$-nearest to t_i, but reasonably close empirically. The reason behind this procedure is that for an exact computation, the fifth $\underline{d}$-nearest nearest terminals $v_{j,5}$ to all Steiner vertices v_j are necessary. We proceed by choosing to each t_i a terminal $v_{k,l}$ such that:

$$\underline{d}(t_i, v_{k,l}) = \min_{\{v_j, v_k\} \in \delta(N(t_i))} \big\{ d(t_i, v_j) + c_{jk} + \min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., 4 : v_{k,l} \neq t_i^{(q)}, q = 0, 1, ..., 3\} \big\}.$$

For all (both non-terminal and terminal) vertices v_i we denote the set of the so obtained (up to four) close terminals by L_i. Next, we scrutinize each edge $e_k = \{v_i, v_j\} \in E$:

Mark each vertex of $t_q \in L_i$ satisfying $\underline{d}(v_i, t_q) < c_{ij}$. Thereafter, proceed for each $t_l \in L_j$ such that $\underline{d}(v_j, t_l) < c_{ij}$ as follows: If t_l has been marked and $\underline{d}(v_i, t_l) + \underline{d}(v_j, t_l) - p_{t_l} < c_{ij}$, delete e_k. Otherwise, if a $t_l^{(r)}, r = 1, ..., 4$ has been marked, we have found a path between v_i and v_j, containing exactly two intermediary terminals, namely t_l and $t_l^{(r)}$. Next, we examine whether the prize-collecting Steiner distance of this path is smaller than the cost of the edge $\{v_i, v_j\}$. For this purpose, we need to check whether the four inequalities

1. $\underline{d}(t_l, t_l^{(r)}) < c_{ij}$,
2. $\underline{d}(t_l, t_l^{(r)}) + \underline{d}(v_j, t_l) - p_{t_l} < c_{ij}$,

3. $\underline{d}(t_l, t_l^{(r)}) - \underline{d}(v_i, t_l^{(r)}) - p_{t_l^{(r)}} < c_{ij}$,

4. $\underline{d}(t_l, t_l^{(r)}) + \underline{d}(v_j, t_l) + \underline{d}(v_i, t_l^{(r)}) - p_{t_l} - p_{t_l^{(r)}} < c_{ij}$

are satisfied. If this is the case, $c_{ij} > s_{pc}(v_i, v_j)$ holds and we consequently remove e_k. The above procedure is reiterated vice versa, starting from v_j and marking all vertices in L_j, to detect additional $v_i - v_j$ paths containing exactly two intermediary vertices. We denote the described test by **SD**. Since the up to four close terminals are readily available, the foregoing examination of an edge can be performed in constant time. Therefore, this can be accomplished for all edges in $\Theta(m)$. Consequently, the SD test can be realized with worst-case complexity of $O(m + n \log n)$. With upper bounds on the bottleneck Steiner distances being available, we can now also extend the NTD_3 test (see Section 2.2.3) to cover the PCSTP, based upon Lemma 26.

Not only exclusion, but also inclusion tests are possible for the PCSTP: For both Lemma 6 and Lemma 9 introduced in Section 2.2.3 we propose an extension for the PCSTP, yielding a different but not less powerful result. A somewhat intricate theorem, which may at first glance appear too constraining to be of practical importance, sets the stage:

Theorem 29. *Let $t_i, t_j \in T$, $W \subset V$, with $t_i \in W$, $t_j \notin W$ and $|\delta(W)| \geq 2$. Further let $e_1 = \{v_1, v_1'\}, e_2 = \{v_2, v_2'\}$, with $v_1, v_2 \in W$, be two distinct edges in $\delta(W)$ such that $c_{e_1} \leq c_{e_2}$ and $c_{e_2} \leq c_{\tilde{e}}$ for all $\tilde{e} \in \delta(W) \setminus \{e_1\}$. Assume that the following three conditions hold:*

$$c_{e_2} \geq d(t_i, v_1) + c_{e_1} + d(v_1', t_j), \tag{62}$$

$$p_{t_j} \geq d(t_i, v_1) + c_{e_1} + d(v_1', t_j), \tag{63}$$

$$p_{t_i} \geq d(t_i, v_1) + c_{e_1}. \tag{64}$$

Thereupon, for each feasible solution S to P containing t_i, v_1, or v_1' (or a combination of them), there is a solution S' of lesser or equal cost containing both t_i and the edge $e_1 = \{v_1, v_1'\}$.

Proof. Let $S = (V_S, E_S)$ be feasible solution to P. Throughout this proof the subsequent procedure will be referred to as **cycle-pruning**:
Let $G' = (V', E')$ be a connected subgraph of G. Until G' is cycle-free (and therefore a tree), repeatedly select a cycle $C_{G'}$ and remove an arbitrary edge of $C_{G'}$ other than e_1 from G'. It should be noted that the prize-collecting cost $P(G') = \sum_{e \in E'} c_e + \sum_{v \in V \setminus V'} p_v$ of G' is not increased by this procedure, since only edges are removed, which are by definition of non-negative cost.

In the remainder of this proof three cases will be differentiated, the first one being $t_i \in V_S$, the second $v_1 \in V_S$ and the third $v_1' \in V_S$.

i) Suppose $t_i \in V_S$, but $\{v_1, v_1'\} \notin E_S$.
If $t_j \in V_S$, there is a unique path in S between t_i and t_j that contains an

edge $\{w, \overline{w}\}$, $w \in W$, $\overline{w} \notin W$ of cost at least c_{e_2}. Removing $\{w, \overline{w}\}$ one obtains a tree S_1 containing t_i and a tree S_2 containing t_j. By including a shortest path P_1 between t_i and v_1 as well as a shortest path P_2 between v_1' and t_j and including the edge $\{v_1, v_1'\}$, a new subgraph S' is obtained. However, S' is not necessarily a tree, since $P_1{}^\circ$ or $P_2{}^\circ$ may contain a vertex of S, resulting in a cycle. Nevertheless, this possible blemish can be remedied by cycle-pruning S'. Thereby S' becomes a tree and, containing V_S, it is of cost:

$$C(S') \leq C(S) - c_{e_2} + d(t_i, v_1) + c_{e_1} + d(v_1', t_j) \overset{(62)}{\leq} C(S).$$

In the complementary case $t_j \notin V_S$, construct a tree S' by once again including P_1 and P_2 and inserting the edge $\{v_1, v_1'\}$, followed by cyle-pruning. The tree S' is of cost:

$$C(S') \leq C(S) + d(t_i, v_1) + c_{e_1} + d(v_1', t_j) - p_{t_j} \overset{(63)}{\leq} C(S).$$

ii) Suppose $v_1 \in V_S$.
If $t_i \notin V_S$, adding a shortest path between t_i and S a tree S' of cost:

$$C(S') \leq C(S) - p_{t_i} + d(t_i, v_1) \overset{(64)}{\leq} C(S)$$

is obtained. Using this tree, case ii) can be reduced to case i).

iii) Suppose $v_1' \in V_S$. If $t_i \in V_S$ but $e_1 \notin E_S$, there is a unique path in S between v_1' and t_i that contains an edge $\{w, \overline{w}\}$, $w \in W$, $\overline{w} \notin W$ of cost at least c_{e_2}. The tree S' obtained from S by removing $\{w, \overline{w}\}$ and inserting e_1 as well as including a shortest path between v_1 and t_i (and if necessary performing cycle-pruning) is of cost:

$$C(S') \leq C(S) + c_{e_1} + d(v_1, t_i) - c_{e_2} \overset{(62)}{\leq} C(S).$$

If, on the other hand, $t_i \notin V_S$, adding a shortest path between v_1 and t_i and, if is not yet contained, the edge e_1, t_i is connected to S. After habitual cycle-pruning a tree S' of cost:

$$C(S') \leq C(S) + c_{e_1} + d(v_1, t_j) - p_{t_i} \overset{(64)}{\leq} C(S)$$

is obtained. □

Contrary to the NV and SL tests for the SPG, for the PCSTP we cannot assume that $\{v_1, v_1'\}$ is part of at least one minimum Steiner tree. However the following corollary allows to contract the edge nevertheless:

Corollary 30. *Assume that the premises of Theorem 29 are fulfilled and furthermore, that inequality* (64) *holds strictly or* $v_1 = t_i$ *is satisfied. Let* $P' = (V', E', c', p')$ *be the PCSTP obtained by contracting* $e_1 = \{v_1, v_1'\}$ *and setting* $p'_v := p_v$ *for* $v \in V' \setminus \{t_i\}$*. If both* $v_1 = t_i$ *and* $v_1' = t_j$*, we set* $p'(t_i) := p_{t_i} + p_{t_j} - c_{e_1}$*; otherwise we define* $p'(t_i) := p_{t_i} - c_{e_1}$*. Thereupon, for each optimal solution* S' *to* P' *the corresponding solution* S *to* P *is also optimal.*

Proof. Let $S' = (V'_{S'}, E'_{S'})$ be a solution to P' and S the corresponding solution to P. Additionally, assume that $e_1 = \{v_1, v_1'\}$ is contracted into v_1 (the opposite case is analogous).

First, suppose $v_1, t_i \notin V'_{S'}$. In this case S' and S consist of exactly the same edges and nodes, which implies that $\sum_{e \in E'_{S'}} c'_{e'} = \sum_{e \in E_S} c_e$. Recall that if both $v_1 = t_i$ and $v_1' = t_j$ hold, then $p'(t_i) = p_{t_i} + p_{t_j} - c_{e_1}$ and otherwise $p'(t_i) = p_{t_i} - c_{e_1}$. Thereupon it can be further inferred that $\sum_{v \notin V'_{S'}} p'_v = \sum_{v \notin V_S} p_v - c_{e_1}$. These deliberations amount to the equation:

$$C'(S') + c_{e_1} = C(S).$$

Second, if $v_1, t_i \in V'_{S'}$, then $\sum_{e \in E'_{S'}} c'_{e'} + c_{e_1} = \sum_{e \in E_S} c_e$ and $\sum_{v \notin V'_{S'}} p'_v = \sum_{v \notin V_S} p_v$, so likewise:

$$C'(S') + c_{e_1} = C(S).$$

Now assume that S' is an optimal solution to P'. We claim that only the two cases discussed above can occur: This assertion is certainly true if $t_i = v_1$, since this excludes additional cases. Otherwise, assume $t_i \in V'_{S'}$, but $v_1 \notin V'_{S'}$. Then $\sum_{e \in E'_{S'}} c'_{e'} = \sum_{e \in E_S} c_e$ and $\sum_{v \notin V'_{S'}} p'_v = \sum_{v \notin V_S} p_v$ so $C'(S') = C(S)$. Furthermore, according to Lemma 29 there would a solution $\tilde{S}$ of cost $C'(\tilde{S}') \leq C(S)$, containing t_i, v_1 and v_1'. Utilizing that the corresponding solution $\tilde{S}'$ to P' would be of cost $C(\tilde{S}) - c_e$, it would follow that:

$$C'(S') = C(S) \geq C(\tilde{S}) = C'(\tilde{S}') + c_e > C'(\tilde{S}'),$$

contradicting the inital assumption that S' is optimal.

Conversely, suppose $v_1 \in V'_{S'}$ but $t_i \notin V'_{S'}$. Due to the initial assumption $p_{t_i} > d(t_i, v_1) + c_{e_1}$, S' could be connected to t_i (in G') by a path of cost at most

$$d(v_1, t_i) < p_{t_i} - c_{e_1} = p'(t_i), \tag{65}$$

which would give rise to a solution of smaller cost.

Next, suppose that there was a solution $\tilde{S}$ to P such that $C(\tilde{S}) < C(S)$. According to Lemma 29, it may be assumed that $\tilde{S}$ contains either t_i, v_1,

and v_1' or none of them. In both cases, the corresponding solution $\tilde{S}'$ to P' would be of cost

$$C'(\tilde{S}') = C(\tilde{S}) - c_e < C(S) - c_e = C'(S'),$$

contradicting the earlier assumption that S' is optimal. Concludingly, S has to be an optimal solution to P. □

In the case of $|\delta(W)| \leq 1$, a corresponding result can be obtained:

Lemma 31. *Let $W \subset V$ and $t_i \in W$, $t_j \notin W$. If $\delta(W) = \emptyset$, $W \cap T = \{t_i\}$ and $p_{t_j} \geq p_{t_i}$, there is an optimal solution $S^\star = (V_{S^\star}, E_{S^\star})$ with $V_{S^\star} \cap W = \emptyset$. If $\delta(W) = \{e_1\} = \{v_1, v_1'\}$, $v_1 \in W$, and the conditions (63) and (64) of Lemma 29 hold, then for each feasible solution S to P containing t_i, v_1, or v_1' (or a combination of them), there is a solution S' of lesser or equal cost containing both t_i and the edge $e_1 = \{v_1, v_1'\}$.*

Proof. Let $\delta(W) = \emptyset$. The only optimal solution that might contain t_i is $\{t_i\}$, which is of equal or higher cost than the feasible solution $\{t_j\}$.

Let $\delta(W) = \{e_1\} = \{v_1, v_1'\}$, $v_1 \in W$, and assume that the conditions (63) and (64) of Lemma 29 hold. Differentiate between three cases:

First, let $t_i \in V_S$. If $t_j \notin V_S$, adding a shortest path between t_i and t_j if $e_1 \notin E_S$ or a shortest path between t_i and t_j otherwise, a subgraph S' is obtained. Analogously to the proof of Lemma 32, by applying cycle-pruning on S' the latter becomes a tree. It contains t_i and e_1 and is of cost:

$$C(S') \leq C(S) + d(t_i, v_1) + c_{e_1} + d(v_1', t_j) - p_{t_j} \overset{(63)}{\leq} C(S).$$

Second, let $v_i \in V_S$. If $t_i \in V_S$ but $e_1 \notin E_S$, then $t_j \notin V_S$. We can include a shortest path between t_i and t_j (containing e_1) and proceed as in the first case to obtain a tree S' of cost $C(S') \leq C(S)$. If $t_i \notin V_S$, we can include a shortest path between S and t_i, yielding a tree of S' of cost:

$$C(S') \leq C(S) + d(t_i, v_1) - p_{t_j} \overset{(63)}{<} C(S).$$

Third, let $v_i' \in V_S$. If $t_i \notin V_S$ we can include a shortest path between t_i and v_i' (containing e_1) and obtain, after cycle-pruning, a tree S' of cost:

$$C(S') \leq C(S) - p_{t_i} + c_{e_1} + d(t_i, v_1) \overset{(64)}{\leq} C(S). \tag{66}$$

On the other hand, if t_i is contained in S, so is e_1, since this is the only way to connect v_i' and t_i. □

With those rather general results at hand, we are now able to formulate extended forms of the NV and SL test for the PCSTP.

Lemma 32. *Let t_i be a terminal of degree at least two and let $\{t_i, v_i'\}$ and $\{t_i, v_i''\}$ be a shortest and a second shortest incident edge, respectively. Assume that there is a terminal $t_j \neq t_i$ such that:*

$$c_{\{t_i,v_i''\}} \geq c_{\{t_i,v_i'\}} + d(v_i', t_j), \tag{67}$$

$$p_{t_j} \geq c_{\{t_i,v_i'\}} + d(v_i', t_j), \tag{68}$$

$$p_{t_i} \geq c_{\{t_i,v_i'\}}. \tag{69}$$

Thereupon, for each feasible solution S to P containing t_i or v_i', there is a solution S' of lesser or equal cost containing $\{t_i, v_i'\}$.

Proof. Defining $W := \{t_i\}$, one readily recognizes the lemma as a special case of Theorem 29. □

Condition (67) of Lemma 32 can be verified analogously to the NV test introduced in Section 2.2 by using Lemma 6. Note that it can be realized in $O(m + n \log n)$ since with $d(v_i', t_j)$ available, the additional conditions (68) and (69) can be verified trivially by checking all incident edges of t_i. Since each edge is thereby checked at most twice, this procedure can be realized for the entire set of terminals in $\Theta(m)$. As with the SPG, the test can be extended based on the subsequent lemma.

Lemma 33. *Let t_i be a terminal of degree at least two and let $e_i' = \{t_i, v_i'\}$ be a shortest edge incident to t_i. Assume that there is a second edge $e_i'' = \{t_i, v_i''\}$ incident to t_i with $v_i'' \notin T$. If there is a terminal $t_j \neq t_i$ such that:*

$$c_e \geq c_{\{t_i,v_i'\}} + d(v_i', t_j) \quad \text{for all } e \in \delta(\{t_i, v_i''\}) \setminus \{e_i'\}, \tag{70}$$

$$p_{t_j} \geq c_{\{t_i,v_i'\}} + d(v_i', t_j), \tag{71}$$

$$p_{t_i} \geq c_{\{t_i,v_i'\}}, \tag{72}$$

then for each feasible solution S to P containing t_i or v_i', there is a solution S' of lesser or equal cost containing $\{t_i, v_i'\}$.

Proof. Defining $W := \{t_i, v_i''\}$, one readily verifies that the lemma is a special case of Theorem 29. □

We denote the test associated with Lemma 32 and Lemma 33 that contracts edges as described in Corollary 30 by **Nearest Vertex** (**NV**) test. Since Lemma 33 can be realized with additional costs of $O(m)$, NV is of $O(m + n \log n)$.

Theorem 29 can be further put to use to verify the following two lemmata, which in turn set the stage for a Voronoi based inclusion test:

Lemma 34. *Let t_i be a terminal and let $\{v_1, v_1'\}$ and $\{v_2, v_2'\}$ be a shortest and a second shortest Voronoi-boundary edge of $N(t_i)$ (satisfying $v_1, v_2 \in N(t_i)$ and $v_1', v_2' \notin N(t_i)$). Let $t_j := base(v_2')$ and assume:*

$$c_{\{v_2, v_2''\}} \geq d(t_i, v_1) + c_{\{v_1, v_1'\}} + d(v_1', t_j), \tag{73}$$

$$p_{t_j} \geq d(t_i, v_1) + c_{\{v_1, v_1'\}} + d(v_1', t_j), \tag{74}$$

$$p_{t_i} \geq c_{\{v_1, v_1'\}} + d(t_i, v_1). \tag{75}$$

Thereupon, for each feasible solution S to P containing t_i, v_1 or v_1' there is a solution S' of lesser or equal cost containing both $\{v_1, v_1'\}$ and t_i.

Proof. Defining $W := N(t_i)$, one readily verifies that the lemma is a special case of Theorem 29. □

Lemma 35. *Let t_i be a terminal and let $e_1' := \{v_1, v_1'\}$ be a shortest shortest Voronoi-boundary edge of $N(t_i)$. Assume that there is a second Voronoi-boundary edge of $N(t_i)$, namely $e_2' = \{v_2, v_2'\}$, with $v_2' \notin T \cup \{v_1'\}$. Further, let $t_j := base(v_2')$ and assume:*

$$c_e \geq d(t_i, v_1) + c_{e_1'} + d(v_1', t_j), \qquad \forall\ e \in \delta(\{t_i, v_i''\}) \setminus \{e_1'\}, \tag{76}$$

$$p_{t_j} \geq d(t_i, v_1) + c_{e_1'} + d(v_1', t_j), \tag{77}$$

$$p_{t_i} \geq d(t_i, v_1) + c_{e_1'}. \tag{78}$$

Then for each feasible solution S to P containing t_i, v_1 or v_1' there is a solution S' of lesser or equal cost containing both $\{v_1, v_1'\}$ and t_i.

Proof. Defining $W := N(t_i) \cup \{v_2'\}$, one readily verifies that the lemma is a special case of Theorem 29. □

We denote the test associated with the lemmata 34 and 35 that contracts edges as described in Corollary 30 by **Short Links** (**SL**). Additionally, this test checks whether the conditions of Lemma 31 are satisfied. SL can be realized in $O(m+n \log n)$: The computation of the Voronoi diagram requires $O(m+n \log n)$ [Meh88] and a shortest and second shortest Voronoi-boundary edge to all terminals can be computed by traversing all Voronoi region, in a total of $\Theta(m)$. The extension of the test affiliated with Lemma 35 is restricted to terminals t_i such that $|\delta(t_i)| \geq 2 * |\delta(v_i'')|$, bounding the additional costs to $O(m)$. Note, that the distances $d(v_1, t_i)$ and $d(v_1', t_j)$ are computed during the build up of the Voronoi regions.

Bound Based Reductions

All reduction techniques described in this subsection have, to the best of our knowledge, not been introduced in the literature so far. Preliminarily,

based on the *radius* introduced in Section 2.2 we define an equivalent for the (rooted) prize-collecting Steiner tree problem as follows:

$$pcradius(t_i) := \min\{radius(t_i), p_{t_i}\},\ t_i \in T. \tag{79}$$

This definition sets the stage for a series of lemmata and corollaries presented hereinafter that allow to eliminate vertices and edges analogously to the BND test introduced in Section 2.2. For ease of understanding, it is presupposed that all terminals are ordered such that $pcradius(t_1) \leq pcradius(t_2) \leq ... \leq pcradius(t_s)$.

Lemma 36. *Let $v_i \in V \setminus T$. If a minimum Steiner tree $S = (V_S, E_S)$ with $v_i \in V_S$ exists, then $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=1}^{s-2} pcradius(t_q)$ is a lower bound on the cost of S.*

Proof. Denote the (unique) path in S between v_i and a terminal $t_j \in V_S$ by P_j and the set of all such paths by $\mathcal{P}$. First, note that $|\mathcal{P}| \geq 2$, because otherwise, with $\mathcal{P} = \{P_l\}$, the tree $\{t_l\}$ would be of smaller cost than S, contradicting the initial assumption that the latter is of minimum cost. Second, two distinct paths in $\mathcal{P}$ can only have a subpath containing v_i in common, but no additional edges, since this would require a cycle in S. Additionally, there are at least two paths in $\mathcal{P}$ having only the vertex v_i in common: Otherwise, due to the precedent observation, all paths would have one edge $\{v_i, v_i'\}$ in common which could be discarded, yielding a tree of smaller cost than S.

Now, choose two distinct paths $P_k \in \mathcal{P}$ and $P_l \in \mathcal{P}$ such that their combined number of Voronoi-boundary edges is minimal and $V[P_k] \cap V[P_l] = \{v_i\}$ holds. Further, define $\mathcal{P}^- := \mathcal{P} \setminus \{P_k, P_l\}$. For all $P_r \in \mathcal{P}^-$, denote by P_r' the subpath of P_r between t_r and the first vertex not in $N(t_r)$. Suppose that P_k has an edge $e \in E_S$ in common with a P_r': Consequently, P_l cannot have any edge in common with P_r, because S is cycle-free. Furthermore, according to the prelusive observations, P_k and P_r have to contain a joint subpath including v_i and e. But in this case, P_k would need to contain at least one additional Voronoi-boundary edge (in order to be able to reach t_k, which is by definition not in $N(t_r)$). Therefore, and due to P_l and P_r being edge disjoint, P_r would have initially been selected instead of P_k.

Following the same line of argumentation, one validates that likewise P_l has no edge in common with any P_r'. Conclusively, the paths P_k, P_l and all P_r' are edge disjoint. Using their combined cost, a lower bound on the cost

of S can be obtained by:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_k) + C(P_l) + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=1}^{s-2} pcradius(t_q) + C(P_k) + C(P_l) \\ &\geq \sum_{q=1}^{s-2} pcradius(t_q) + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) \end{aligned}$$

where $C(P) := \sum_{e \in E[P]} c_e$. Ergo, the lemma is proven. □

A Steiner vertex v_i can be discarded if its associated lower bound, specified in Lemma 36, exceeds a known upper bound U of the underlying problem. If a solution S of cost U is available, v_i can eliminated in the case of equality of both bounds if additionally $v_i \notin V[S]$ is satisfied. Analogous tests are possible for all subsequent lemmata and corollaries in this subsection.

A similar approach can be applied to probe whether a terminal is part of any optimal solution. Recall that to each terminal t_i we denote by $t_i^{(1)} \neq t_i$ a terminal of shortest distance to t_i.

Corollary 37. *Let $t_i \in T$ and assume that a minimum Steiner tree $S = (V_S, E_S)$ other than $\{t_i\}$ exists such that $t_i \in V_S$. Additionally, let $\phi : \{1, .., s\} \to \{1, .., s\}$ be a bijection, such that $\phi(i) = 1$ and all $t_{\phi(j)}$ are ordered such that the $pcradius(t_{\phi(j)})$ values are non-decreasing in $j = 2, ..., s$. In this case, $\underline{d}(t_i, t_i^{(1)}) + \sum_{q=2}^{s-1} pcradius(t_{\phi(q)})$ is a lower bound on the cost of S.*

Proof. For each terminal $t_j \in (V_S \cap T) \setminus \{t_i\}$ denote by P_j the (unique) path in S between t_i and t_j. Further, denote the collection of all such paths by $\mathcal{P}$. Let $P_k \in \mathcal{P}$ be a path with a minimal number of Voronoi-boundary edges. Let $\mathcal{P}^- := \mathcal{P} \setminus \{P_k\}$. If $\mathcal{P}^-$ is non-empty, denote for each $P_r \in \mathcal{P}^-$ the subpath of P_r between t_r and the first vertex not in $N(t_r)$ by P'_r.

Using the same argumentation as in the proof to Lemma 36, one validates that P_k and all P'_r are edge disjoint. Therefore, the cost of S can be bounded

from below as follows:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_k) + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=2}^{s-1} pcradius(t_{\phi(q)}) + C(P_k) \\ &\geq \sum_{q=2}^{s-1} pcradius(t_{\phi(q)}) + \underline{d}(t_i, t_i^{(1)}), \end{aligned}$$

with the last line yielding the desired result. □

If a terminal cannot be eliminated by means of the conditions of the antecedent corollary, another approach based on the following lemma can be attempted:

Lemma 38. *Let $t_i \in T$ and assume that a minimum Steiner tree $S = (V_S, E_S)$ exists such that t_i is of degree at least two in S. Additionally, let $\phi : \{1, .., s\} \to \{1, .., s\}$ be a bijection, such that $\phi(i) = 1$ and all $t_{\phi(j)}$ are ordered such that the $pcradius(t_{\phi(j)})$ values are non-decreasing in $j = 2, ..., s$. In this case, $\underline{d}(t_i, t_i^{(1)}) + \underline{d}(t_i, t_i^{(2)}) + \sum_{q=2}^{s-2} pcradius(t_{\phi(q)})$ is a lower bound on the cost of S.*

Proof. Define $\mathcal{P}$ accordingly to the proof of Corollary 37. Additionally, denote by $P_k, P_l \in \mathcal{P}$ two edge disjoint paths such that their combined number of Voronoi-boundary edges is minimal and define $\mathcal{P}^- := \mathcal{P} \setminus \{P_k, P_l\}$. Define for each $P_r \in \mathcal{P}^-$ the subpath of P_r between t_r and the first vertex not in $N(t_r)$ by P'_r. Proceeding as in the proof to Lemma 36 it can be verified that P_k, P_l and all P'_r are edge disjoint. Hence:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_k) + C(P_l) + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=2}^{s-2} pcradius(t_{\phi(q)}) + C(P_k) + C(P_l) \\ &\geq \sum_{q=2}^{s-2} pcradius(t_{\phi(q)}) + \underline{d}(t_i, t_i^{(1)}) + \underline{d}(t_i, t_i^{(2)}). \end{aligned}$$

Thereby the claim is established. □

The last result can be utilized for a simple reduction test that checks whether a terminal that has been proven to be of degree at most one in at least one minimum Steiner tree can be discarded:

Lemma 39. *Let $t_i \in T$ and assume $p_{t_i} \leq c_e$ for all $e \in \delta(t_i)$. If t_i is not contained or of degree one in at least one minimum Steiner tree, there exists a minimum Steiner tree not containing t_i.*

Proof. Suppose that there is a minimum Steiner tree such that t_i is of degree one in S (otherwise the claim is already established). Deleting t_i and its incident edge in S, one obtains a new minimum Steiner tree. □

If the premises of Corollary 37 or Lemma 39 are fulfilled, t_i and all incident edges can be deleted. To obtain the original solution value, p_{t_i} needs to be added to the objective value of each feasible solution (analogously to the TD_0 test).

Lemma 40. *Let $\{v_i, v_j\} \in E$. If there is a minimum Steiner tree $S = (V_S, E_S)$ such that $\{v_i, v_j\} \in E_S$, then L defined by*

$$L := c_{\{v_i,v_j\}} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{s-2} pcradius(t_q) \tag{80}$$

if $base(v_i) \neq base(v_j)$ and

$$L := c_{\{v_i,v_j\}} + \min\left\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1})\right\} + \sum_{q=1}^{s-2} pcradius(t_q) \tag{81}$$

otherwise, is a lower bound on the cost of S.

Proof. Initially, define $T_S := V_S \cap T$ and denote the set of all paths in S between v_i and vertices in T_S reachable without using the edge $\{v_i, v_j\}$ by $\mathcal{P}_{v_i}$, and analogously the set of all paths in S between v_j and the remaining vertices in T_S by $\mathcal{P}_{v_j}$. Additionally, set $\mathcal{P} := \mathcal{P}_{v_i} \cup \mathcal{P}_{v_j}$. Note, that single vertices are considered a path in this proof. Therefore, both $\mathcal{P}_{v_i}$ and $\mathcal{P}_{v_j}$ are non-empty, since otherwise the edge $\{v_i, v_j\}$ could be removed from S to obtain a Steiner tree of smaller cost.

As in the proof to Lemma 11 one easily verifies that: first, a path in $\mathcal{P}_{v_i}$ cannot have edges in common with any path in $\mathcal{P}_{v_j}$ (and vice versa); second, two distinct paths that are both in either $\mathcal{P}_{v_i}$ or $\mathcal{P}_{v_j}$ can only have a subpath containing v_i or v_j respectively in common, but no additional edges.

Having set the stage, we choose two paths $P_k \in \mathcal{P}_{v_i}$ and $P_l \in \mathcal{P}_{v_j}$ such that their combined number of Voronoi-boundary edges is minimal. Further, define $\mathcal{P}^- := \mathcal{P} \setminus \{P_k, P_l\}$. For $P_r \in \mathcal{P}$, denote its subpath between t_r

and the first vertex not in $N(t_r)$ by P'_r. Following precisely the line of argumentation utilized in the proof to Lemma 11, one validates that the paths P_k, P_l and all P'_r are edge disjoint. Likewise, it can be demonstrated, that in the case of $base(v_i) \neq base(v_j)$ the cost of P_k plus the cost of P_l is at least $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1})$. With these results at hand one readily verifies that:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_k) + C(P_l) + c_{\{v_i, v_j\}} + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=1}^{s-2} pcradius(t_q) + C(P_k) + C(P_l) + c_{\{v_i, v_j\}} \\ &\geq \sum_{q=1}^{s-2} pcradius(t_q) + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + c_{\{v_i, v_j\}} \end{aligned}$$

Likewise in the case of $base(v_i) = base(v_j)$ the combined cost of P_k and P_l is at least $\min\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}\}$. Consequently:

$$\begin{aligned} C(S) &\geq \sum_{q=1}^{s-2} pcradius(t_q) + C(P_k) + C(P_l) + c_{\{v_i, v_j\}} \\ &\geq \sum_{q=1}^{s-2} pcradius(t_q) + \min\left\{ \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}) \right\} \\ &\quad + c_{\{v_i, v_j\}}. \end{aligned}$$

So overall, L as defined in (81) is a lower bound on the weight of S. □

Lemma 41. *Let $v_i \in V \setminus T, \delta(v_i) \geq 3$. If there exists a minimum Steiner tree S such that v_i is of degree at least three in S, then $\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \underline{d}(v_i, v_{i,3}) + \sum_{q=1}^{s-3} pcradius(t_q)$ is a lower bound on the weight of S.*

Proof. Initially, define $\mathcal{P}$ as in the proof to Lemma 36. One readily observes that there are at least three edge disjoint paths in $\mathcal{P}$: There are at least three distinct edges $e_{i,1}, e_{i,2}, e_{i,3} \in E_S$ incident to v_i. For $z = 1, 2, 3$ there exists a path $P_z \in \mathcal{P}$ containing $e_{i,z}$, since otherwise the edge $e_{i,z}$ could be discarded to obtain a tree S' containing the same terminals as S and being therefore of smaller cost. Since S is by definition cycle-free, the three paths cannot have a vertex other than v_i in common.

Having verified their existence, let $P_j, P_k, P_l \in \mathcal{P}$ be three edge disjoint paths in S such that they jointly contain a minimal number of Voronoi-boundary edges. For all remaining $P_r \in \mathcal{P}$, define P'_r as the subpath between

t_r and the first vertex not in $N(t_r)$. Analogously to the proof of Lemma 36, one validates that P_j, P_k, P_l and all P'_r are edge disjoint. Therefore, with $\mathcal{P}^- := \mathcal{P} \setminus \{P_j, P_k, P_l\}$:

$$\begin{aligned} C(S) &= \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_j) + C(P_k) + C(P_l) + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=1}^{s-3} pcradius(t_q) + C(P_j) + C(P_k) + C(P_l) \\ &\geq \sum_{q=1}^{s-3} pcradius(t_q) + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \underline{d}(v_i, v_{i,3}), \end{aligned}$$

which proves the lemma. □

This lemma can be likewise modified to handle terminals. Recall that for a terminal $t_i \in T$ we denote by $t_i^{(2)}$ and $t_i^{(3)}$ the terminals being second and third nearest to t_i with respect to $\underline{d}$.

Corollary 42. *Let $t_i \in T, \delta(t_i) \geq 3$ and assume the existence a minimum Steiner tree S such that v_i is of degree at least three in S. Furthermore, let $\phi : \{1, .., s\} \to \{1, .., s\}$ be a bijection, such that $\phi(i) = 1$ and all $t_{\phi(j)}$ are ordered such that the $pcradius(t_{\phi(j)})$ values are non-decreasing in $j = 2, ..., s$. Thereupon, $\underline{d}(t_i, t_i^{(1)}) + \underline{d}(t_i, t_i^{(2)}) + \underline{d}(t_i, t_i^{(3)}) + \sum_{q=2}^{s-3} pcradius(t_{\phi(q)})$ is a lower bound on the weight of S.*

Proof. For each terminal $t_j \in (V_S \cap T) \setminus \{t_i\}$ denote by P_j the (unique) path in S between t_i and t_j, and the collection of all such paths by $\mathcal{P}$. As in the proof to Lemma 41 it can be verified that there are at least three edge disjoint paths in P. Once again, choose three such paths $P_j, P_k, P_l \in P$, each with a minimal number of Voronoi-boundary edges. Denoting $\mathcal{P}^- := \mathcal{P} \setminus \{P_j, P_k, P_l\}$ and defining for all $P_r \in P^-$, P'_r as the subpath between t_r and the first vertex not in $N(t_r)$, one analogously yields that P_j, P_k, P_l and all P'_r are edge disjoint. Therefore, the cost of S can be bounded from

below as follows:

$$\begin{aligned} C(S) &= \sum_{e \in E} c_e + \sum_{v \in V \setminus V_S} p_v \\ &\geq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_j) + C(P_k) + C(P_l) + \sum_{v \in V \setminus V_S} p_v \\ &\geq \sum_{q=2}^{s-3} pcradius(t_{\phi(q)}) + C(P_j) + C(P_k) + C(P_l) \\ &\geq \sum_{q=2}^{s-3} pcradius(t_{\phi(q)}) + \underline{d}(t_i, t_i^{(1)}) + \underline{d}(t_i, t_i^{(2)}) + \underline{d}(t_i, t_i^{(3)}). \end{aligned}$$

□

Unfortunately, it appears to be difficult to translate Corollary 42 to a reduction test, only a special case is covered here:

Lemma 43. *Let $t_i \in T, \delta(t_i) = 3$ and assume that there is no minimum Steiner tree S such that t_i is of degree three in S. Let $e_i^{(1)}, e_i^{(2)}, e_i^{(3)}$ be the three incident edges to t_i with $c_{e_i^{(1)}} \leq c_{e_i^{(2)}} \leq c_{e_i^{(3)}}$. Further, assume that $p_{t_i} \leq c_{e_i^{(1)}}$ and $p_{t_i} \leq p_{t_j}$ for a $t_j \in T$. Additionally, let $v_i^{(1)}, v_i^{(2)}, v_i^{(3)} \in V$ such that $e_i^{(k)} = \{t_i, v_i^{(k)}\}$ for $k = 1, 2, 3$. Construct a new PCSTP instance $P'_{PC} = (V', E', c', p')$ from P_{PC} as follows:*

1. *Initially, set $V' := V$, $E' := E$ and remove t_i and $\delta(t_i)$ from V' and E' respectively. For all remaining $e \in E'$ and $v \in V'$ set $c'_e := c_e$ and $p'_v := p_v$.*
2. *For $k = 1, 2, 3$: Define edges $e'_k := \{v_i^{(k)}, v_i^{((k \bmod 3)+1)}\}$ of cost $c'_{e'_k} := c_{e_i^{(k)}} + c_{e_i^{((k \bmod 3)+1)}} - p_{t_i}$. If there is no edge $\{v_i^{(k)}, v_i^{((k \bmod 3)+1)}\} \in E'$, add e'_k to E'; if there is such an edge being of higher cost than e'_k, replace it by e'_k. Otherwise discard e'_k.*

Thereupon, each solution S' to P'_{PC} can be transformed to a solution S to P_{PC} by replacing each e'_k ($k = 1, 2, 3$) contained in S' by the edges $e_i^{(k)}$ and $e_i^{((k \bmod 3)+1)}$. If S is optimal, S is optimal as well and $C'(S') + p_{t_i} = C(S)$ holds.

Proof. Let S' be an optimal solution to P'_{PC}. The subgraph S yielded from S' by the described transformation is indeed a solution to P_{PC}: Since S is connected and edges are substituted by paths, the same holds for S'. Furthermore, for the same reason, if there was a cycle in S', there would be one in S as well.

Note that any solution $\tilde{S}$ to P_{PC} such that t_i is either not contained or of degree two, can be reduced to a solution $\tilde{S}'$ to P'_{PC} by replacing the two edges $e_i^{(k)}$ and $(e_i^{((k \bmod 3)+1)}$, if contained in $\tilde{S}$, by the edge e'_k. Subsequently, distinguish three cases:

First, if S' does not contain any of the newly inserted edges, its cost can be calculated by:

$$\begin{aligned} C'(S') &= \sum_{e \in E'} c'_e + \sum_{v \notin V'_{S'}} p'_v \\ &= \sum_{e \in E} c_e + \sum_{v \notin V_S} p_v - p_{t_i} \\ &= C(S) - p_{t_i}. \end{aligned}$$

Second, if S' contains exactly one newly inserted edge $c'_{e'_k}$, of cost $c_{e_i^{(k)}} + c_{e_i^{((k \bmod 3)+1)}} - p_{t_i}$, S contains the edges $e_i^{(k)}$ and $e_i^{((k \bmod 3)+1)}$ instead and likewise the vertex t_i. Therefore, the cost of S' can be stated as:

$$\begin{aligned} C'(S') &= \sum_{e \in E'} c'_e + \sum_{v \notin V'_{S'}} p'_v \\ &= \sum_{e \in E} c_e - p_{t_i} + \sum_{v \notin V_S} p_v \\ &= C(S). \end{aligned}$$

Third, suppose S' contained two of the newly inserted edges $c'_{e'_k}$ and $c'_{e'_l}$. Their combined cost would be $c_{e_i^{(1)}} + c_{e_i^{(2)}} + c_{e_i^{(3)}} + c_{e_i^{(l)}} - 2p_{t_i}$ for some $l \in \{1, ..., 3\}$. Moreover, t_i would be of degree three in S and since S could consequently not be optimal, there would be a solution $\tilde{S}$ to P with $C(\tilde{S}) > C(S)$ and t_i either not contained or of degree two. Hence:

$$\begin{aligned} C'(S') &= \sum_{e \in E \cap E'_{S'}} c'_e + c_{e_i^{(1)}} + c_{e_i^{(2)}} + c_{e_i^{(3)}} + c_{e_i^{(l)}} - 2p_{t_i} + \sum_{v \notin V'_{S'}} p'_v \\ &= \sum_{e \in E} c_e + c_{e_i^{(l)}} - 2p_{t_i} + \sum_{v \notin V_S} p_v \\ &\geq \sum_{e \in E} c_e - p_{t_i} + \sum_{v \notin V_S} p_v \\ &= C(S) - p_{t_i} \\ &> C(\tilde{S}) - p_{t_i} \\ &= C'(\tilde{S}'), \end{aligned}$$

contradicting the premise that S' is of minimum cost. Finally, note that there is no fourth case, since S' is by definition cycle-free and can therefore not contain all three newly inserted edges.

It remains to be demonstrated that S is of minimum cost: Suppose that this was not true. Consequently there would be a solution $\tilde{S}$ to P_{PC} of cost $C(\tilde{S}) < C(S)$ such that t_i was either not contained or of degree two. Denoting the corresponding solution to P' by $\tilde{S}'$, it could be inferred that:

$$C'(S') = C(S) - p_{t_i} > C(\tilde{S}) - p_{t_i} = C'(\tilde{S}').$$

Hence, S is indeed of minimum cost. □

The test associated with the introduced bound-based reduction approaches is denoted by **Bound** (**BND**) test. It works with an upper bound computed by the constructive heuristic that will be introduced in Section 3.4.3. The worst-case complexity, which would otherwise be of $O(m + n \log n)$, is dominated by the heuristic.

Ordering

The underlying principles for the disposition of the reduction methods within an overall loop are the same as for the SPG, see Section 2.2.5. Likewise, each method block other than the easy DT tests is disabled in a successive iterations as soon as less than 0.1 percent of all edges could be eliminated. The condition of truncation for the whole loop remains the same. Note that the ordering reported in Figure 11 is used for both the rooted and unrooted prize-collecting Steiner tree problem.

3.4.3 Heuristics

For the RPCSTP the one-phase RSP heuristic introduced in Section 2.3.1 can be employed on the corresponding SAP obtained from Transformation 3, using outgoing arcs instead of edges. Considering the PCSTP, we suggest an adaptation of one-phase RSPH: Given a PCSTP consider the rcSAP obtained by Transformation 4. Instead of commencing from different vertices, the starting point is invariably the (artificial) root. In each run all arcs between the root and non-terminals (denoted by (r', t) in Transformation 4) are temporarily removed, except for one. A tree is then computed on this new graph, using the same process as the original RSP heuristic, but only considering outgoing arcs. Instead of starting from a new terminal, the arc to remain in the graph is varied. Analogously to the use of RSPH for the SPG, in the course of the solving process the arc costs are modified according to the values of the best available LP solution.

The VQ heuristic requires an adaptation for both the RPCSTP and the PCSTP: All terminals are temporarily removed from the (transformed) graph and VQ is executed with all t_i, as defined in Transformation 3, marked as key vertices. Finally, the pruning has to be slightly adjusted such that (47), and for the PCSTP also (54), is satisfied by each solution.

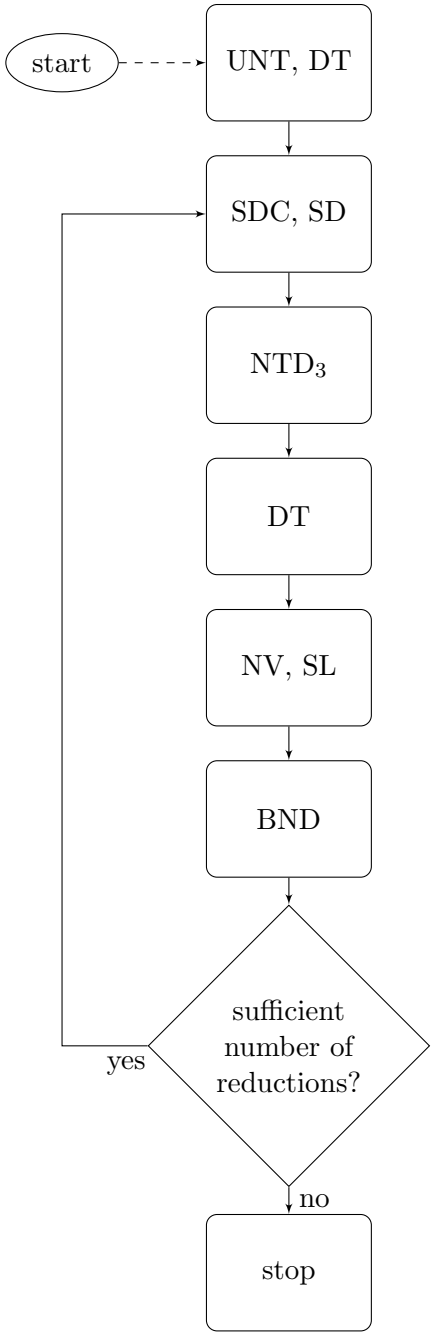


Figure 11: Illustration of the (R)PCSTP reduction package.

3.4.4 Implementation and Computational Results

Although the formulations underlying the rooted and unrooted PCSTP differ, we still consider them as related enough to justify a joint computational evaluation. The three PCSTP and the two RPCSTP test sets are outlined in Table 12.

Name	Instances	$\|V\|$	Status	Description
JMP	34	100-400	solved	Sparse instances of varying structure, introduced in [JMP00].
CRR	80	500-1000	solved	Sparse instances with up to $25,000$ edges, based on the C and D test sets of the SteinLib [Lju04].
PUCNU	18	1200-4096	unsolved	Hard instances derived from the PUC set, with $c \equiv 1$, $p \in \{1, 2\}$. From the 11th DIMACS Challenge.
Cologne1	14	741-751	solved	RPCSTP instances derived from the design of fiber optic networks for German cities [Lju04].
Cologne2	15	1801-1810	solved	

Table 12: Classes of (R)PCSTP instances.

For the JMP class all instances are solved without branching in less than three seconds, with an average of 0.6 seconds as illustrated in Table 13. Likewise, all instances of the CRR test set are solved in between 0.04 and 6.9 seconds. For neither of the two sets a branch-and-bound search is required. The third test set considered for the evaluation is the PUCNU class, derived from the PUC test set, see Section 2.6. As SCIP-Jack is already unable to solve many of the original instances, similar results are observed for the respective PCSTP versions. Of all instances ten are solved to optimality, with three of them requiring branching. The remaining eight instances evince optimality gaps in the range of 0.6 to 2.7 percent at termination.

Table 13: Computational results for PCSTP instances

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
JMP	34	34	1.0	0.6	–	–
CRR	80	80	1.0	1.0	–	–
PUCNU	18	10	3.9	43.0	54.5	1.8

Results on the RPCSTP sets are reported in Table 14. All instances are solved at the root node, in an average of 2.6 seconds for Cologne1 and 20.3 seconds for Cologne2. Each instance collected in the Cologne1 test set, is solved in less than eight seconds, the, larger, instances from Cologne2 require 0.6 up to 54.2 seconds.

Class	Instances	Solved	Optimal ∅ Nodes	Optimal ∅ Time [s]
Cologne1	14	14	1.0	2.6
Cologne2	15	15	1.0	20.3

Table 14: Computational results for RPCSTP instances.

Impact of Reduction Techniques

The results illustrated in Table 15 bespeak the strength of the employed reduction techniques. On average the number of edges is reduced by more than 50 percent and even for the hard PUCNU test set, whose SPG predecessor allowed for less than one percent of reduction in Section 2.6.1, more than 15 percent of the edges could be removed. Furthermore, it should be noted that the Cologne1 and Cologne2 test sets are already reduced by the techniques described in [Lju04], which renders the results even more notable. Of the Cologne1 instances four are already solved during presolving, while for the Cologne2 set this holds only for one instance. Of the PCSTP instances, 10 can be solved by reduction methods alone. On each test set, the execution of the reduction techniques requires less than one second (with respect to the shifted geometric mean).

Class	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]
JMP	57.1	51.7	0.1
CRR	34.8	22.6	0.4
PUCNU	96.5	84.7	0.6
Cologne1	36.8	37.0	0.3
Cologne2	59.0	59.5	0.9

Table 15: Computational results of the (R)PCSTP reduction package.

Furthermore, results on the leverage of the individual reduction components within the overall package are provided: Table 16 displays the results of iteratively excluding each reduction method from the whole package (analogously to the procedure in Section 2.6.1). As opposed to the results formerly presented in this thesis, the simple reduction techniques (UNT and DT) for the (R)PCSPG are overall not the most effective ones. Nevertheless, a considerable impact on the CRR instances and also on the JMP class is manifest. Interestingly enough, on Cologne2 the simple reductions lessen the overall strength of the reduction package, which means they hinder the elimination of further edges by the succeeding methods. The SD and SDC tests display a strong, robust performance, most notably for the PUCNU test set on which they can credit the majority of reductions. On the downside, however, one observes a high execution time, occupying more than 50 percent of the overall time on all five test classes. The effect of both the NTD_3 and the NV, SL

methods is somewhat less pronounced, but at the same time they reveal a higher efficiency, even improving the overall execution time on several tests sets, for example both the RPCSTP classes. Finally, the bound-based reduction strategy (BND) is notably strong on classes containing instances of few terminals, i.e. CRR, Cologne1 and Cologne2. Concomitantly, the total run time of the reduction package is improved on CRR as well as Cologne1.

	Class	UNT, DT	SDC, SD	NTD_3	NV,SL	BND
Edges	JMP	**+30.0**	**+57.8**	**+33.8**	**+9.1**	0
	CRR	**+118.5**	**+89.8**	**+16.2**	**+29.6**	**+72.6**
	PUCNU	**+4.2**	**+16.8**	0	0	0
	Cologne1	**+3.1**	**+27.5**	**+3.3**	-3.0	**+107.9**
	Cologne2	-0.3	**+20.5**	**+11.8**	**+0.9**	**+46.7**
Time	JMP	-9.2	-51.9	-14.0	**+8.9**	**+1.8**
	CRR	**+1.4**	-50.9	-16.8	-0.8	**+18.2**
	PUCNU	-10.7	-55.5	-0.8	-1.1	-1.6
	Cologne1	-4.3	-68.1	**+14.9**	**+6.9**	**+27.6**
	Cologne2	-18.8	-80.8	**+2.0**	**+2.6**	-13.9

Table 16: Evaluation of the (R)PCSTP preprocessing components. Each column reports the results of the reduction package without the specified method. The values denote the percentual change with respect to the default settings (see Table 15).

Having discussed our reduction approach for the PCSTP in depth, we furthermore aspire to classify our experimental results with respect to related publications. The most effective reduction techniques published in the literature are from [Uch06], clearly surpassing previous results [KLM+04, Lju04, LR04]. The author performed computational experiments on the CRR instances, which can be subdivided into a C and D test set. In both Table 17 and Table 18 a detailed comparison is provided between the results reported in [Uch06] and those of our PCSTP reduction package. Unfortunately, no run times are provided in [Uch06]. However, the fact that so called *expansions* of reduction tests were used, which can be exceedingly time-consuming [Pol04], suggests that these computations require far more time than our approach. No methods of this type are deployed in this thesis, since the purpose of the reduction approach is a more subordinate one, namely to serve as a component for fast exact solving.

Nevertheless, our reduction package still achieves notably better results on both test sets, eliminating almost one third more edges and vertices compared to [Uch06]. We ascribe this mostly to the BND test that manifests an impressive efficiency on the instances with few terminals. An example is the D11-A instance, for which all 5000 edges can be eliminated, over 3000 of them by the BND test. Indeed, on the instances of small $|T|$, the differences between our package and the results reported in [Uch06] become most pronounced, which is again epitomized in the D11-A instance: While

our reduction package already solves the problem, the approach described in [Uch06] leaves 977 vertices and 3971 edges untouched.

	Original			Reduced		Reduced [Uch06]	
Name	Nodes	Edges	Terminals	Nodes	Edges	Nodes	Edges
C01-A	500	625	5	1	0	105	190
C01-B	500	625	5	8	11	49	77
C02-A	500	625	10	1	0	82	148
C02-B	500	625	10	70	128	71	125
C03-A	500	625	83	138	244	113	190
C03-B	500	625	83	80	146	79	121
C04-A	500	625	125	122	218	72	119
C04-B	500	625	125	97	175	71	113
C05-A	500	625	250	56	112	7	9
C05-B	500	625	250	9	14	1	0
C06-A	500	1000	5	1	0	346	792
C06-B	500	1000	5	60	126	344	778
C07-A	500	1000	10	1	0	353	806
C07-B	500	1000	10	276	638	342	769
C08-A	500	1000	83	318	720	251	531
C08-B	500	1000	83	256	555	217	410
C09-A	500	1000	125	330	744	279	577
C09-B	500	1000	125	283	609	232	440
C10-A	500	1000	250	215	508	103	166
C10-B	500	1000	250	37	59	100	156
C11-A	500	2500	5	1	0	485	1801
C11-B	500	2500	5	155	378	480	1667
C12-A	500	2500	10	191	458	453	1495
C12-B	500	2500	10	389	1194	441	1358
C13-A	500	2500	83	400	1152	343	799
C13-B	500	2500	83	370	1010	317	704
C14-A	500	2500	125	316	816	190	365
C14-B	500	2500	125	265	656	179	330
C15-A	500	2500	250	167	412	1	0
C15-B	500	2500	250	1	0	1	0
C16-A	500	12500	5	65	144	499	2714
C16-B	500	12500	5	65	144	499	2714
C17-A	500	12500	10	353	1147	494	2295
C17-B	500	12500	10	176	459	494	2295
C18-A	500	12500	83	434	1553	374	1002
C18-B	500	12500	83	434	1551	374	997
C19-A	500	12500	125	283	733	246	589
C19-B	500	12500	125	285	735	249	592
C20-A	500	12500	250	1	0	1	0
C20-B	500	12500	250	1	0	1	0
Avg. ratio				33.6%	21.5%	46.7%	29.1%

Table 17: Reductions on C instances.

	Original			Reduced		Reduced [Uch06]	
Name	Nodes	Edges	Terminals	Nodes	Edges	Nodes	Edges
D01-A	1000	1250	5	1	0	223	422
D01-B	1000	1250	5	133	261	223	416
D02-A	1000	1250	10	1	0	238	450
D02-B	1000	1250	10	217	406	232	423
D03-A	1000	1250	167	230	422	114	194
D03-B	1000	1250	167	131	244	129	202
D04-A	1000	1250	250	242	446	171	297
D04-B	1000	1250	250	63	108	50	70
D05-A	1000	1250	500	179	355	84	125
D05-B	1000	1250	500	31	51	12	17
D06-A	1000	2000	5	1	0	740	1697
D06-B	1000	2000	5	378	969	736	1682
D07-A	1000	2000	10	1	0	721	1664
D07-B	1000	2000	10	473	1119	702	1602
D08-A	1000	2000	167	718	1621	673	1489
D08-B	1000	2000	167	628	1393	561	1142
D09-A	1000	2000	250	664	1521	580	1260
D09-B	1000	2000	250	492	1131	439	841
D10-A	1000	2000	500	463	1173	235	425
D10-B	1000	2000	500	131	309	36	56
D11-A	1000	5000	5	1	0	977	3971
D11-B	1000	5000	5	178	441	972	3740
D12-A	1000	5000	10	361	916	960	3300
D12-B	1000	5000	10	350	871	942	3040
D13-A	1000	5000	167	853	2531	708	1713
D13-B	1000	5000	167	788	2221	694	1631
D14-A	1000	5000	250	683	1898	571	1238
D14-B	1000	5000	250	607	1603	512	1062
D15-A	1000	5000	500	358	961	139	217
D15-B	1000	5000	500	8	12	4	5
D16-A	1000	25000	5	126	330	1000	6735
D16-B	1000	25000	5	126	330	1000	6725
D17-A	1000	25000	10	857	4148	999	6330
D17-B	1000	25000	10	637	2297	999	6330
D18-A	1000	25000	167	907	3706	812	2314
D18-B	1000	25000	167	907	3691	806	2276
D19-A	1000	25000	250	755	2383	686	1895
D19-B	1000	25000	250	747	2338	680	1870
D20-A	1000	25000	500	1	0	1	0
D20-B	1000	25000	500	1	0	1	0
Avg. ratio				36.1%	23.8%	50.9%	33.4%

Table 18: Reductions on D instances.

Impact of Heuristics

To measure the impact of the primal heuristics, we proceed as in Section 2.6 and deploy them in accumulative stages. As with the SPG, one observes a steady decline of the average gap when stepping up the employed heuristics. This ranges from 72.81 percent when using no Steiner heuristics to 1.84 percent for their full-fledged employment; the sharpest descent can be observed when enabling the RSP heuristic, which reduces the average gap by 69.4 percent. Furthermore, the results presented in Table 19 display that also the execution time is considerably reduced when the heuristics come into play.

However, one notes that excluding the average gap the differences between the accumulative stages of employed heuristics are diverse: On the JMP, CRR and Cologne2 class the execution times increase with the deployment of VQ as compared to using RSPH alone. On the other hand, by using both RSPH and VQ an additional PUC instance can be solved. Comparing the full-fledged heuristic package with RSPH, one sees that the latter prevails on the JMP as well as on the Cologne2 class, albeit with a low margin. On a related note, the additional instance solved by RSPH/VQ is also solved using all heuristics when the time limit is prolonged by five percent.

Summing up, the deployment of problem specific heuristics has been demonstrated to notably help improve the execution time and, to a far greater extent, the average gap for both the rooted and unrooted prize-collecting Steiner tree problem. While with respect to the overall execution time the impact of additionally employing the VQ and RC heuristic on top of RSPH is a minor, and partially even negative, one, the results on the, hard, PUC instances show a notable reduction of the average gap.

	Name	Instances	Solved	∅ Gap [%]	∅ Time [s]
none	JMP	34	34	–	1.6
	CRR	80	80	–	1.8
	PUCNU	18	11	71.81	726.5
	Cologne1	14	14	–	3.5
	Cologne2	15	15	–	34.5
RSPH	JMP	34	34	–	0.5
	CRR	80	80	–	1.0
	PUCNU	18	10	2.41	480.0
	Cologne1	14	14	–	3.2
	Cologne2	15	15	–	18.1
+VQ	JMP	34	34	–	0.9
	CRR	80	80	–	1.3
	PUCNU	18	11	2.00	459.9
	Cologne1	14	14	–	2.5
	Cologne2	15	15	–	19.6
+RCH	JMP	34	34	–	0.6
	CRR	80	80	–	1.0
	PUCNU	18	10	1.84	460.8
	Cologne1	14	14	–	2.6
	Cologne2	15	15	–	20.3

Table 19: Computational results of employing the Steiner problem heuristics in accumulative stages. The gap is given as the arithmetic mean among the unsolved problems, the time as the shifted geometric mean among all problems.

3.5 The Maximum Weight Connected Subgraph Problem

At first glance, the **maximum-weight connected subgraph problem** (**MWCSP**) bears little resemblance to the Steiner problems introduced so far: Given an undirected graph (V, E) with (possibly negative) node weights p, the objective is to find a tree that maximizes the sum of its node weights. However, it is possible to transform this problem into a prize-collecting Steiner tree problem. One transformation is given in [DKR+08]. In this thesis an alternative transformation is presented that leads to a significant reduction in the number of terminals for the resulting problem, as compared to the approach described in [DKR+08]. For our model the reduced amount of terminals leads to a substantial decrease of constraints (of the form (54)), as will become evident in this section.

In the following it is assumed that at least one vertex is assigned a negative and one a positive weight. In the case of only non-negative vertex-weights, the problem reduces to finding a spanning tree; in the case of only non-positive vertex-weights, the empty set constitutes an optimal solution. For a given solution S to an MWCSP we denote its weight by $C(S) := \sum_{v \in V[S]} p_v$.

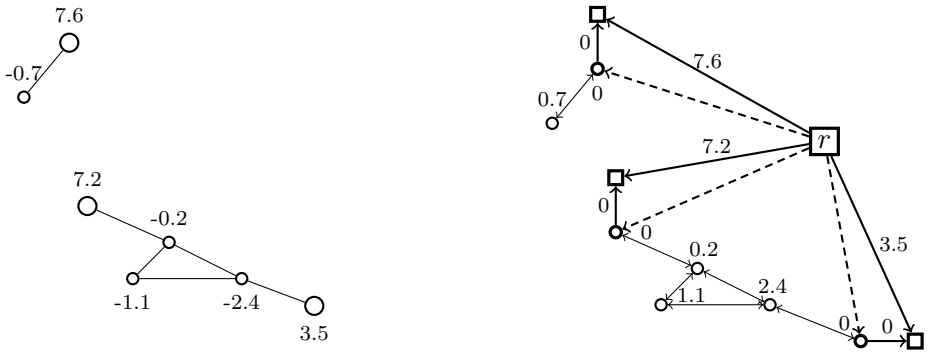
3.5.1 Transformation

Transformation 5 (MWCSP to rcSAP)**.**
Input: *An MWCSP* $P = (V, E, p)$
Output: *An rcSAP* $P'' = (V'', A'', T'', c'', r'')$

1. *Set* $V' := V$, $A' := \{(v, w) \in V' \times V' \mid \{v, w\} \in E\}$.
2. *Set* $c' : A' \to \mathbb{Q}_{\geq 0}$ *such that for* $a = (v, w) \in A'$*:*
$$c'_a = \begin{cases} -p_w, & \text{if } p_w < 0 \\ 0, & \text{otherwise} \end{cases}$$
3. *Set* $p' : V' \to \mathbb{Q}_{\geq 0}$ *such that for* $v \in V'$*:*
$$p'_v = \begin{cases} p_v, & \text{if } p_v > 0 \\ 0, & \text{otherwise} \end{cases}$$
4. *Perform Transformation 4 to* (V', A', c', p')*, slightly changed in such a way, that in step 2 instead of constructing a new arc set,* A' *is being used. The resulting rcSAP gives us* $P'' = (V'', A'', T'', c'', r'')$.

An illustration of Transformation 5 on an MWCS instance consisting of two connected components can be found in Figure 12.

A bijection between the solution spaces of the MWCS and the rcSAP obtained by applying Transformation 5 is established in the following lemma:



(a) MWCSP instance

(b) transformed rcSAP instance (modifications in **bold**, terminals drawn as squares)

Figure 12: Illustration of an MWCSP instance (left) and the associated rcSAP obtained by Transformation 5 (right). Of the dashed arcs only one can be selected. Note that for the MWCSP only vertex weights are given, while, in contrast, the rcSAP is endowed solely with arc costs.

Lemma 44 (MWCSP to rcSAP)**.** *Let $P = (V, E, p)$ be an MWCSP and $P'' = (V'', A'', T'', c'', r'')$ an rcSAP obtained from P by Transformation 5. Thereupon, the set of solutions to P'' can be bijectively mapped to the set of solutions to P: Assign each solution $S'' = (V''_{S''}, A''_{S''})$ to P'' the solution $S = (V_S, E_S)$ to P defined by:*

$$V_S := \{v \in V : \ v \in V''_{S''}\}, \tag{82}$$

$$E_S := \{\{v, w\} \in E : \ (v, w) \in A''_{S''} \text{ or } (w, v) \in A''_{S''}\}. \tag{83}$$

Furthermore, for the objective value $C(S)$ of S and the objective value $C''(S'')$ of S'' the following equality holds:

$$C(S) = \sum_{v \in V : p_v > 0} p_v - C''(S''). \tag{84}$$

Proof. The one-to-one correspondence between the sets of solutions to P and P'' can be seen analogously to the proof of Lemma 24.

To prove (84), let $S = (V_S, E_S)$ be a solution to P and $S'' = (V''_{S''}, A''_{S''})$ the corresponding solution to P'', obtained by applying (82) and (83). Further, define $A := \{(v, w) \in A'' : \ \{v, w\} \in E\}$. First, one observes that for each $v \in S$ such that $p_v \leq 0$ there is exactly one incoming arc $a \in A_{S''}$, so:

$$\sum_{v \in V_S : p_v \leq 0} p_v = - \sum_{a \in A_{S''}} c''_a. \tag{85}$$

Second:

$$\sum_{v \in V_S : p_v > 0} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{v \in V \setminus V_S : p_v > 0} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{a \in A''_{S''} \setminus A_{S''}} c''_a. \tag{86}$$

Finally, adding (85) and (86) the equation:

$$\sum_{v \in V_S} p_v = \sum_{v \in V : p_v > 0} p_v - \sum_{a \in A''_{S''}} c''_a \tag{87}$$

is obtained, which coincides with (84). □

Since after presolving the proportion of positive weight vertices is small, as will become evident in Section 3.5.2, Transformation 5 results in problems with significantly less terminals compared to the transformation described in [DKR+08]. However, also for all real-world DIMACS instances most of the vertex weights are non-positive. The differences between the two transformations with respect to the number of terminals for these instances are presented in Table 20.

3.5.2 Reductions

As to reduction techniques, the MWCSP with only a few methods having been published [AB14, EK14] falls far short of reaching the same scope as the SPG. This deficiency provided an incentive for us to develop several new techniques which will be subsequently introduced along with some known ones. As before, proves will only be provided for reduction techniques not published in the literature yet. Throughout this section it will be presupposed that an MWCS instance $P_{MW} = (V, E, p)$ is given. The reader is reminded that, antipodally to all other problems discussed in this thesis, in the case of the MWCSP the objective value is to be maximised. Furthermore, it should be kept in mind that at least one vertex is assumed to be of positive and one of negative cost.

Instance	Transformation 5	Transformation from [DKR+08]
drosophila001	71	5226
drosophila005	194	5226
drosophila0075	250	5226
HCMV	55	3863
lymphoma	67	2034
metabol_expr_mice_1	150	3523
metabol_expr_mice_2	85	3514
metabol_expr_mice_3	114	2853

Table 20: Number of terminals after transforming.

Basic Reductions

The first of the following two simple tests was already suggested in a less general form in [EK14] (namely only for negative vertices of degree zero):

***Vertex of degree zero (VD$_0$)**: A vertex v_i of degree zero can be discarded if there is a vertex v_{max} such that both $p_{v_{max}} \geq p_{v_i}$ and $v_{max} \neq v_i$.*

Proof. The only solution v_i might be contained in is $\{v_i\}$, which is of equal or less cost than the feasible solution $\{v_{max}\}$. □

***Vertex of degree one (VD$_1$)**: If there is vertex v_i of degree one with incident edge $e_k = \{v_i, v_j\}$, a vertex v_{max} satisfying both $p_{v_{max}} \geq p_{v_i}$ and $v_{max} \neq v_i$ and it moreover holds that*

a) $p_{v_i} \leq 0$, then v_i and e_k can be discarded;

b) $p_{v_i} > 0$, then v_i and e_k can be discarded while adding p_{v_i} to the weight of v_j.

Proof. Since $p_{v_i} \leq p_{v_{max}}$ and $v_{max} \neq v_i$ hold, $\{v_i\}$ cannot be the unique optimal solution.

a) Each solution that contains e_k (and v_i) can be reduced to solution of not higher cost by removing e_k as well as v_i. Hence, there is at least one optimal solution containing neither e_k nor v_i.

b) If v_j is contained in an optimal solution, v_i is as well, since $p_{v_i} > 0$. Hence, at least one optimal solution is preserved by the reduction described in part b) of the test. □

The next two reduction tests can be found in both [AB14] and [EK14].

***Non-negative incident vertices (NNIV)**: An edge $\{v_i, v_j\}$ such that $p_{v_i} \geq 0$, $p_{v_j} \geq 0$ can be contracted if the weight of the remaining vertex is set to $p_{v_i} + p_{v_j}$.*

***Non-positive chain (NPC)**: A path P between two vertices v_i, v_j, containing only non-positive interior vertices of degree two and fulfilling $|V[P]| > 3$ can be replaced by a node v' with $p_{v'} := \sum_{v \in V[P^\circ]} p_v$ and the two edges $\{v', v_i\}, \{v', v_j\}$.*

The joint execution of the four forgoing tests, in the specified order, will be hereinafter referred to as **Basic Test** (**BT**).

Recalling that connectivity is not postulated for the MWCSP, one readily devises the following test:

***Unreachable non-positive vertex (UNPV)**: Each non-positive vertex v_i that is not connected to any positive vertex can be deleted along with all incident edges.*

Proof. Since there is at least one vertex of positive cost, v_i cannot be contained in any optimal solution. □

We realize this test by running a breadth-first search from the (artificial) root on the transformed graph yielded by Transformation 5 and discarding all vertices that have not been visited in the course of the search.

Alternative Based Reductions

For the avoidance of doubt, we point out that apart from the subsequent test all MWCSP reduction methods introduced hereinafter have been newly devised for this thesis. The subsequent reduction test was tentatively suggested in [EK14], to be later generalized in [AB14]. Only the latter version is stated here:

Adjacent Neighbourhood Subset (ANS) [5] *Let v_i and v_j be two distinct vertices. If $p_{v_i} \leq 0$, $p_{v_i} \leq p_{v_j}$ and*

$$\big\{v \in V \setminus \{v_j\} \mid \{v_i, v\} \in E\big\} \subset \big\{v \in V \mid \{v_j, v\} \in E\big\},$$

then v_i and all incident edges can be removed.

However, even this revised version can be notably generalized:

Lemma 45. *Let $v_i \in V$ and $W \subset V \setminus \{v_i\}$, $W \neq \emptyset$ such that $(W, E[W])$ is connected and $\sum_{w \in W : p_w < 0} p_w \geq p_{v_i}$ holds. If:*

$$\big\{v \in V \setminus W \mid \{v_i, v\} \in E\big\} \subset \big\{v \in V \setminus W \mid \{w, v\} \in E, w \in W\big\}, \tag{88}$$

then there is at least one optimal solution that does not contain v_i (or any of its incident edges).

Proof. Suppose that there is an optimal solution $S = (V_S, E_S)$ containing v_i. From $\sum_{w \in W : p_w < 0} p_w \geq p_{v_i}$ it can be inferred that $p_{v_i} \leq 0$. Therefore, it can be assumed that v_i is of degree at least two in S (if v_i is of degree one in S, it can be simply discarded without deteriorating the objective value). Let $\Delta_S \subset V_S$ be the vertices adjacent to v_i in S. Removing v_i and all incident edges from S and inserting all vertices in $W \setminus S$ as well as all edges between W and $\Delta_S \cup W$, one obtains a connected subgraph $S' = (V_{S'}, E_{S'})$ such that:

$$C(S') = \sum_{v \in V_{S'}} p_v \geq \sum_{\substack{w \in W \\ p_w < 0}} p_w + C(S) - p_{v_i} \geq C(S). \tag{89}$$

[5] The reduction test was originally named "Neighbourhood Subset", abbreviated "NS", test in [AB14], but owned to rather obvious historical considerations we have decided to substitute it.

However, S' is not necessarily a tree, but this issue can be remedied without changing the weight of S' by repeatedly removing an edge on a cycle as long as there exists any. Due to (89) the resulting tree is of cost at least $C(S)$ and therefore optimal. Furthermore, it does not contain v_i, so the statement of the lemma is established. □

By processing for each node v_i only a limited amount of neighbours that are moreover of degree not higher than a given constant, the ANS test can be performed in $\Theta(m)$. Affiliated to Lemma 45 we implemented the following reduction test: For each vertex v_j of negative weight we consider as W the union of v_j and a limited amount of non-negative weight vertices adjacent to v_j. By again considering only neighbours v_i to v_j of bounded degree, this test can likewise be implemented in $\Theta(m)$. In the following, this test is referred to as **Connected Neighbourhood Subset (CNS)**. Although it dominates the ANS test in its unbounded version, this is not true for the actually implemented test if the bound on the number of vertices to be processed differs. So we use both the restricted ANS and the restricted CNS test, but with a higher upper bound on the admissible vertex degrees for the less expensive ANS test.

The perhaps most intuitive, but nonetheless effective, of the reduction tests introduced hereinafter is based on the following lemma:

Lemma 46. *Let $e_k = \{v_i, v_j\} \in E$. If there is a path P between v_i and v_j such that $e_k \notin E[P]$ and $p_{v_p} \geq 0$ for all $v_p \in V[P^\circ]$, then there is at least one optimal solution that does not contain e_k.*

Proof. Suppose that there is a solution S to P_{MW} containing e_k. By removing e_k from S one obtains two trees S_1, S_2 such that $v_i \in S_1$, $v_j \in S_2$. Since P connects v_i and v_j, there are vertices $v_r \in V[S_1] \cap V[P]$, $v_s \in V[S_2] \cap V[P]$ such that $P(v_r, v_s)$ does not include any additional vertices of S_1 or S_2. Reconnecting S_1 and S_2 by $P(v_r, v_s)$ one yields a new tree $\tilde{S}$ of weight at least $C(S)$.

These considerations prove the lemma, since they imply that to each optimal solution that contains e_k, an optimal solution not containing e_k can be constructed. □

We suggest a reduction test based on Lemma 46, designed as follows: By first executing the NNIV test, each path containing only interior vertices of non-negative weight is reduced to a path consisting of three vertices (with an interior vertex of non-negative weight). Thereafter, for each edge $e_k = \{v_i, v_j\}$ we mark all non-negative adjacent vertices of v_i, except for v_j. If no vertex of non-negative cost has been marked, we can already progress to the next edge. Otherwise, we traverse all non-negative neighbours of v_j. If in the course of the second search a marked vertex is visited, we eliminate e_k. By bounding the number of adjacent vertices to be visited from v_i and

v_j by a constant, the test can be performed for all edges in a total of $\Theta(m)$. We denote this test **Non-Negative Paths (NNP)**.

The development of the next reduction strategy was spawned by the observation that after having performed the NPC test, each non-positive chain is reduced to a single non-positive vertex of degree two. Naturally, one aspires to dispose of those as well. To set the stage for such a reduction method we define a function on two vertices, evocative of the bottleneck Steiner distance: Let $v_i, v_j \in V$ be two distinct vertices and $\mathcal{P}(v_i, v_j)$ the set of all simple paths between v_i and v_j. Further, for $P \in \mathcal{P}(v_i, v_j)$ let $P(x, y)$ be the subpath between two vertices x, y in P. In the context of the MWCS we define the **interior cost** of such a subpath as

$$C^\circ(P(x,y)) := \sum_{v \in V[P(x,y)] \setminus \{x,y\}} p_v. \tag{90}$$

Furthermore, we define the **vertex weight bottleneck length** of P as:

$$l_{vw}(P) := \min_{x,y \in V[P]} C^\circ(P(x,y)). \tag{91}$$

Given two disjoint trees that can be connected by a subpath of a given path P, by $l_{vw}(P)$ a lower bound on the additional vertex weights is provided.

The two preceding definitions pave the way for the **vertex weight bottleneck distance** between v_i and v_j, which we define as

$$d_{vw}(v_i, v_j) = \max\{l_{vw}(P) \mid P \in \mathcal{P}(v_i, v_j)\}. \tag{92}$$

With this new definition at hand, a lemma bearing some commonalities with the Steiner distance based exclusion tests can be formulated.

Lemma 47. *Let $v_i \in V$ such that $p_{v_i} \leq 0$ and $|\delta(v_i)| = 2$ with incident edges $\{v_i, v_j\}$ and $\{v_i, v_k\}$. If*

$$d_{vw}(v_j, v_k) > p_{v_i} \tag{93}$$

holds, then there is at least one optimal solution not containing v_i.

Proof. Suppose that there is an optimal solution S that contains v_i. In this case, due to v_i being of non-positive weight, we can assume that both incident edges of v_i are likewise part of the solution. Otherwise, v_i could simply be removed, to yield another optimal solution. Removing v_i as well as its incident edges from S, we obtain two subtrees S_1 and S_2. Let P be a path between v_j and v_k such that $l_{vw}(P) = d_{vw}(v_j, v_k)$. Additionally, let $P(v_1', v_2')$ be a subpath of P between $v_1' \in V[S_1]$ and $v_2' \in V[S_2]$ such that no

additional vertices of either S_1 or S_2 are contained. By including $P(v_1', v_2')$ a new tree S' is obtained which is of cost:

$$\begin{aligned} C(S') &= C(S) + C^\circ(P(v_1', v_2')) - p_{v_i} \\ &\geq C(S) + l_{vw}(P) - p_{v_i} \\ &= C(S) + d_{vw}(v_j, v_k) - p_{v_i} \\ &\overset{(93)}{>} C(S). \end{aligned}$$

This proves the lemma. □

We suggest the following reduction test to utilize Lemma 47: First perform step 1 and step 2 of Transformation 5 to obtain a directed graph $D' = (V', A')$ with, non-negative, arc costs c'. Following, a modified version of the SDC test can be used on D' to find alternative paths containing at most one interior vertex of positive weight: Let $v_i \in V$ be non-positive and of degree two with adjacent vertices v_j and v_k. Similar to the original SDC test, a limited execution of Dijkstra's algorithm is performed first from v_j and then from v_k, ignoring v_i and its incident edges. For each vertex being processed during Dijkstra's algorithm, only outgoing arcs are considered and the computation does not proceed from vertices of positive weights. If directed paths $\vec{P}_k'$ and $\vec{P}_j'$ from v_k and v_j respectively to a vertex $v_l \in V$ have been found, denote by P_{jk} the corresponding undirected path in G between v_j and v_k (over v_l) and distinguish two cases (note that we denote by p the vertex weights in the original graph G):

1. if $p_{v_l} > 0$, then $l_{vw}(P_{jk}) = \min\{-C'(\vec{P}_j'), -C'(\vec{P}_k'), -C'(\vec{P}_j') - C'(\vec{P}_k') + p_{v_l}\}$;
2. if $p_{v_l} \leq 0$, then $l_{vw}(P_{jk}) = -C'(\vec{P}_j') - C'(\vec{P}_k') - p_{v_l}$.

In the first case, we consider the interior costs of the two subpaths between each endpoint of P_{jk} and the intermediary vertex of positive weight as well as the interior cost of the whole path. The minimum among those three is equal to the vertex weight bottleneck length of P_{jk}. In the second case, P_{jk} does not contain any intermediary vertex of positive weight, so $l_{vw}(P_{jk}) = C^\circ(P_{jk})$ holds.

If $l_{vw}(P_{jk}) \geq p_{v_i}$, the vertex v_i and its incident edges are deleted. We denote this procedure by **Non-Positive Vertex of Degree two** (**$\mathbf{NPV_2}$**) test.

The concept of the vertex weight distance not only gives rise to Lemma 47 and the affiliated NPV_2 test, but furthermore leads the way to a far more general and powerful result:

Lemma 48. *Let $v_i \in V$ such that $p_{v_i} \leq 0$ and denote by Δ the set of all vertices adjacent to v_i. Furthermore, for $\Delta' \subset \Delta$ denote by $K_{\Delta'} :=$*

$(\Delta', \Delta' \times \Delta', d_{vw})$ the complete graph on Δ', with edge weight $d_{vw}(v_j, v_k)$ for an edge $\{v_j, v_k\} \in \Delta' \times \Delta'$. If for each $\Delta' \subset \Delta$ of cardinality at least two it holds that the weight of a maximum spanning tree on $K_{\Delta'}$ is greater than p_{v_i}, then there is at least one optimal solution that does not contain v_i (or any of its incident edges).

Proof. It can be readily verified that in the cases $|\Delta| = 0, 1, 2$ the claim has already been established by the proofs to VD_0, VD_1 and Lemma 47 respectively. So suppose $|\Delta| > 2$ and let $S = (V_S, E_S)$ be a tree such that v_i is of degree $k > 0$ in S. If $k < 2$, the claim can be established analogously to the hereinbefore referred to proofs. Otherwise, denote by $\Delta' = \{v'_1, ..., v'_k\} \subset V_S$ the vertices incident to v_i in S. With the premises of the lemma being satisfied, there is maximum spanning tree $T_{\Delta'}$ on $K_{\Delta'}$ of weight greater than p_{v_i}. The solution S can be segmented into k trees $S_1, .., S_k$ such that $v_j \in S_j$ for $j = 1, ..., k$, by removing v_i and $\delta(v_i)$. In the following we will iteratively rejoin these k trees to obtain a new tree not containing v_i and being of no lesser weight than S.

First, one observes that each edge of $T_{\Delta'}$ corresponds to a path in G between two vertices of Δ'. Let P_{rs} with $r, s \in \{1, ..., k\}$ be such a path connecting v'_r and v'_s. Since the edge between v'_r and v'_s is a spanning tree for the subset $\{v'_r, v'_s\} \subset \Delta$, it holds that $d_{vw}(v'_r, v'_s) > p_{v_i}$. Consequently, v_i is not contained in P_{rs}. Analogously to the proof of Lemma 47, S_r and S_s can be connected to a tree S'_1 by a subset of P_{rs}, which is of cost at least $d_{vw}(v'_r, v'_s)$. Thereupon, S'_1 can be linked in the same way to a tree S_l, $k \in \{1, ..., k\} \setminus \{r, s\}$, bringing forth a tree S'_2 that once more does not contain v_i. Reiterating in this manner, one can rejoin S'_2 with all not yet incorporated trees S_q, finally yielding a tree S'_{k-1}. This tree does not contain v_i and is of cost:

$$C(S'_{k-1}) \geq \sum_{l=\{1,...,k\}} C(S_l) + \left(\sum_{\{v_q, v_l\} \in E[T_{\Delta'}]} d_{vw}(v_q, v_l) \right) - p_{v_i} \geq C(S)$$

Consequently, the claim is established. □

Corollary 49. *Let $v_i \in V$ such that $p_{v_i} \leq 0$ and denote by Δ the set of all vertices adjacent to v_i. Denote the vertex weight bottleneck distances on the graph $G^- := (V^-, E^-)$ with $V^- := V \setminus \{v_i\}, E^- := E[V^-]$ by d^-_{vw}. Further, for $\Delta' \subset \Delta$ define $K_{\Delta'} := (\Delta', \Delta' \times \Delta', d^-_{vw})$. If for each $\Delta' \subset \Delta$ of cardinality at least two it holds that the weight of a maximum spanning tree on $K_{\Delta'}$ is not less than p_{v_i}, there is at least one optimal solution that does not contain v_i (or any of its incident edges).*

Proof. The proposition can be verified largely in line with the proof to Lemma 48. As the sole amendment, to demonstrate that for joining the subtrees $S_1, ..., S_k$ the vertex v_i can be disregarded one simply uses the fact that all paths corresponding to d^-_{vw} do not include v_i by definition. □

We use reduction tests based on the antecedent corollary only for non-positive vertices of degree three, four and five and refer to the corresponding methods as **Non-Positive Vertex of Degree k** ($\mathbf{NPV_k}$) test ($k \in \{3, 4, 5\}$). In line with the proceeding for the bottleneck based tests to both the SPG and the PCSTP, the vertex weight distances are not pre-computed and are furthermore substituted by ad-hoc calculated lower bounds to each pair of vertices in Δ. To this end, the procedure deployed in the NPV_2 test is utilized. If a vertex has been verified to satisfy the premises of Corollary 49, it is removed along with all incident edges.

Bound Based Reductions

Throughout this subsection, the connected components of G are denoted by $G_1, ..., G_l$. Utilizing the UNPV test we can assume that to each component $G_i = (V_i, E_i)$ there is at least one vertex $v_j \in V_i$ such that $p_{v_j} > 0$. For two distinct vertices $v_i, v_j \in V$ denote by $\mathcal{P}_{ij}$ the set of all simple paths between v_i and v_j only containing non-positive interior vertices. We define the **non-positive vertices distance** from v_i to v_j as:

$$d_{npv}(v_i, v_j) := \begin{cases} \max_{P \in \mathcal{P}_{ij}} \left\{ \sum_{v \in V[P]} p_v \right\} & \text{if } \mathcal{P}_{ij} \neq \emptyset \\ -\infty & \text{otherwise.} \end{cases} \tag{94}$$

Note that this definition includes the weights of the endpoints of a path, in contrast for instance to the interior cost C°.

Furthermore, denote by $T = \{t_1, ..., t_s\}$ the set of all vertices in G of positive weight and by $T_r = \{t_{\varphi(1)}, ..., t_{\varphi(s_r)}\}$ the corresponding, non-empty, set of positive vertices of a connected component G_r. Ensuing, we denote a partition $\{N(t) \mid t \in T\}$ of V such that

$$v_i \in N(t) \Rightarrow d_{npv}(v_i, t) \geq d_{npv}(v_i, t'), \ \forall t' \in T \tag{95}$$

by **maximum-weight Voronoi diagram**, in the context of the MWCS simply called Vornonoi diagram.

For computing such a Voronoi diagram we associate with each vertex $v_i \in V$ the following three values (which are also used for customary Voronoi diagrams [Meh88]):

- **base**($\mathbf{v_i}$): the positive vertex $t_j \in T$ such that $v_i \in N(t_j)$;
- **pred**($\mathbf{v_i}$): the predecessor of v_i on the path from $base(v_i)$ to v_i, set to -1 if $p_{v_i} > 0$;
- **dist**($\mathbf{v_i}$): the non-positive vertices distance $d_{npv}(base(v_i), v_i)$.

Those values can be computed by the subsequent algorithm:

Algorithm 1 (Maximum Weight Voronoi Diagram)**.**
Input: *An MWCS instance* (V, E, p)
Output: *Values base, pred and dist to each vertex of* G

1. *Initialize a priority queue* Q *of capacity* $|V|$*. Further, apply step 1 and step 2 of Transformation 5 to obtain a directed graph* $D' = (V', A')$ *with non-negative arc costs* c'*.*

2. *For all* $v_i \in V'$ *initialize values* $dist'$*,* $base'$ *and* $pred'$ *as follows: If* $p_{v_i} \leq 0$ *set* $dist'(v_i) = \infty$*,* $base'(v_i) = -1$*,* $pred(v_i) = -1$ *and mark* v_i *as unvisited, otherwise set* $dist'(v_i) = -p_{v_i}$*,* $base'(v_i) = v_i$*,* $pred'(v_i) = -1$ *mark* v_i *as visited and insert it into* Q*.*

3. *If* Q *is empty, goto step 5, otherwise remove a vertex of minimal* $dist'$ *value from* Q*, denote it by* v_{min} *and mark it as visited.*

4. *For each unvisited neighbour* $v_{min}^{(i)}$ *of* v_{min}*: If* $dist'(v_{min}^{(i)}) > dist'(v_{min}) + p(v_{min}^{(i)})$*, set* $dist'(v_{min}^{(i)}) := dist'(v_{min}) + p(v_{min}^{(i)})$*,* $base'(v_{min}^{(i)}) = base'(v_{min})$*,* $pred'(v_{min}^{(i)}) = v_{min}$ *and add* $v_{min}^{(i)}$ *to* Q *if it is not already contained. Goto step 3.*

5. *For all* $v_i \in V'$ *set* $dist(v_i) := -dist'(v_i)$*,* $pred(v_i) := pred'(v_i)$ *and* $base(v_i) := base'(v_i)$*.*

Lemma 50. *Algorithm 1 is guaranteed to terminate and the values returned by it form a Voronoi diagram.*

Proof. It can be readily verified that the algorithm terminates after a finite number of steps, since in each iteration one vertex is removed from Q and no vertex is inserted more than once. However, its correctness cannot be demonstrated quite as easily:

We conduct the proof by induction on the number of vertices that have been removed from Q. Preliminarily, the following invariant hypothesis is postulated, denoting each vertex that has been removed from the queue Q by **processed**: For each processed vertex v_i it holds that $-dist'(v_i) = \max_{t_j \in T} d_{npv}(t_j, v_i)$. For each not yet processed vertex v_i it holds that $-dist'(v_i)$ is the maximum non-positive vertices distance on processed nodes only, starting in a (processed) $t_j \in T$; if no such path exists $dist'(v_i) = \infty$ holds. In both cases, a corresponding path with the value $-dist(v_i)$ (if finite) can be obtained by iteratively moving along the *pred* values starting with $pred(v_i)$. This path starts in $base(v_i)$.

The base case is tantamount to exactly one, positive weight, vertex being processed and hence the hypothesis holds.

Assume the hypothesis for $n-1$ processed vertices. Next, choose a vertex v_i such that the $dist'$ value is minimal among all non-processed vertices. Suppose $-dist'(v_i) < d_{npv}(t_j, v_i)$ for a $t_j \in T$. Let P be a path between t_j

and v_i of vertex weight $d_{npv}(t_j, v_i)$. Starting from t_j let v_k be the first vertex in P not processed yet. If $v_k = v_i$, this would contradict the hypothesis for unprocessed vertices. Otherwise, since all vertices on P except for t_j are of non-positive weight, it can be inferred that $d_{npv}(t_j, v_k) \geq d_{npv}(t_j, v_i)$. Hence, v_k would have been selected instead of v_k, contradicting the initial assumption. Consequently, $-dist'(v_i) = \max_{t_j \in T} d_{npv}(t_j, v_i)$ is satisfied. Moreover, as the *dist*, *pred* and *base* values of v_i endure unaltered, a path corresponding to the value $-dist(v_i)$ and ending in $base(v_i)$ can still be obtained by means of the *pred* values.

Finally, when step 4 of the algorithm has been performed, it is still true that for each unprocessed vertex v_r the value $-dist'(v_r)$ is the maximal non-positive vertices distance on processed nodes starting in a $t_j \in T$: If there was a path of higher weight (in G) not containing v_i, the vertex v_r would have been selected previously and there cannot be a path of higher weight containing v_i, since it would have been updated while processing v_i. Furthermore, since by the hypothesis a path corresponding to the $-dist(v_i)$ value and starting in $base(v_i)$ can be obtained by the *pred* values, the according property holds true for each unprocessed v_r updated while processing v_i.

To conclude the demonstration, it remains to be verified that after the termination of the algorithm all vertices of G are processed: Suppose that this is not true and let be v_i be a non-processed vertex in a connected component G_k. Further, let $t_j \in T_k$ and P a path from t_j to v_i. The terminal vertex is processed, since it is inserted into Q in step 2 of the algorithm. Let v_0' be the first unprocessed vertex in P. However, v_0' would have been inserted into Q (and subsequently processed) while the predecessor of v_0' in P was being processed. □

Still, some definitions are necessary to conclusively pave the way for the MWCS bound-based reduction techniques: For a $v_i \in V \setminus T$ denote by $v_{i,1}$, $v_{i,2}$ two positive vertices such that $d_{npv}(v_i, v_{i,1}) \geq d_{npv}(v_i, v_{i,2})$ and $d_{npv}(v_i, v_{i,2}) \geq d_{npv}(v_i, t_j)$ for all $t_j \in T \setminus \{v_{i,1}\}$. If only one terminal is reachable from v_i, set $v_{i,2} := -1$. The $v_{i,1}$ values can be computed by Algorithm 1, the $v_{i,2}$ values by a simple adaption of Duin's nearest terminals algorithm, see Section 1.1.

Based on the preceding discussion, for $t_i \in T$ we define:

$$\mathbf{mwradius(t_i)} := \max\big\{0, \max\{d_{npv}(t_i, v_j) \mid \{v_j, v_k\} \in E : v_j \in N(t_i), v_k \notin N(t_i)\}\big\} \quad (96)$$

Once again succumbing to the convenience of a lucid notation, it will be supposed that for each $i = 1, ..., r$, all $t_j \in T_i$ are mapped by φ in such a manner that $mwradius(t_{\varphi(t_1)}) \leq mwradius(t_{\varphi(t_2)}) \leq ... \leq mwradius(t_{\varphi(t_{s_r})})$.

Lemma 51. *Let $v_i \in V$ with $p_{v_i} \leq 0$ and assume that an optimal solution $S = (V_S, E_S)$ to P_{MW} exists such that $v_i \in V_S$. Let G_r be the connected*

component containing v_s. The value

$$d_{npv}(v_i, v_{i,1}) + d_{npv}(v_i, v_{i,2}) - p_{v_i} + \sum_{l=3}^{s_r} mwradius(\varphi(t_l)) \tag{97}$$

is an upper bound on the weight of S.

Proof. Denote the (unique) path in S between v_i and a $t_j \in T \cap V[G_r]$ by P_j and the set of all such paths by $\mathcal{P}$.

Analogously to the proof of Lemma 36, two distinct paths $P_k \in \mathcal{P}$ and $P_l \in \mathcal{P}$ can be selected such that they jointly contain a minimal number of Voronoi-boundary edges and $V[P_k] \cap V[P_l] = \{v_i\}$. Likewise, define $\mathcal{P}^- := \mathcal{P} \setminus \{P_k, P_l\}$. For all $P_u \in \mathcal{P}^-$, denote by P'_u the subpath of P_u starting in t_u and ending in the last vertex still in $N(t_u)$ (so that P'_u lies completely in $N(t_u)$).

Suppose that P_k has a vertex $v_q \in V_S$ in common with a P'_u. As a consequence, P_l cannot have any vertex but v_i in common with P_u, because this would require a cycle in S. For the same reason, P_k and P_u contain a joint subpath between v_i and v_q. However, the existence of such a subpath implies that P_k includes at least one additional Voronoi-boundary edge (in order to be able to reach t_k which is by definition not in $N(t_u)$). Therefore, P_u would have initially been selected instead of P_k.

Following the same line of argumentation, one validates that likewise P_l cannot share a vertex with any P'_u. Conclusively, the paths P_k, P_l and all P'_u are vertex disjoint except for v_i which is contained in both P_k and P_l. Using their combined weight, one obtains an upper bound on the cost of S as follows:

$$\begin{aligned}
C(S) &= \sum_{v \in V_S} p_v \\
&\leq \left(\sum_{P_u \in \mathcal{P}^-} C(P'_u) \right) + C(P_k) + C(P_l) - p_{v_i} \\
&\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) + C(P_k) + C(P_l) - p_{v_i} \\
&\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) + d_{npv}(v_i, v_{i,1}) + d_{npv}(v_i, v_{i,2}) - p_{v_i}.
\end{aligned}$$

Thereupon, the claim is established. □

Lemma 52. *Let $\{v_i, v_j\} \in E$ and let G_r be the connected component containing $\{v_i, v_j\}$. If there is an optimal solution $S = (V_S, E_S)$ containing $\{v_i, v_j\}$, then U defined by*

$$U := d_{npv}(v_i, v_{i,1}) + d_{npv}(v_j, v_{j,1}) + \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) \tag{98}$$

if $base(v_i) \neq base(v_j)$ *and*

$$U := \max\left\{d_{npv}(v_i, v_{i,1}) + d_{npv}(v_j, v_{j,2}), d_{npv}(v_i, v_{i,2}) + d_{npv}(v_j, v_{j,1}\right\} + \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) \quad (99)$$

otherwise, is an upper bound on the weight of S.

Proof. Define $T_S := V_S \cap T$ and denote the set of all paths in S between v_i and vertices in T_S reachable without using the edge $\{v_i, v_j\}$ by $\mathcal{P}_{v_i}$, and analogously the set of all paths in S between v_j and the remaining vertices in T_S by $\mathcal{P}_{v_j}$. Additionally, set $\mathcal{P} := \mathcal{P}_{v_i} \cap \mathcal{P}_{v_j}$. In line with the proof to Lemma 40, both $\mathcal{P}_{v_i}$ and $\mathcal{P}_{v_j}$ are non-empty. Likewise, choose two paths $P_k \in \mathcal{P}_{v_i}$ and $P_l \in \mathcal{P}_{v_j}$ such that each of them contains a minimal number Voronoi-boundary edges. Further, let $\mathcal{P}^- := \mathcal{P} \setminus \{P_k, P_l\}$ and for $P_r \in \mathcal{P}^-$ define $P'_r \in \mathcal{P}$ as the longest subpath of P_r starting from v_r and lying completely in $N(v_r)$. Analogously to the proof of Lemma 51, one validates that the paths P_k, P_l and all P'_r are vertex disjoint. In the case of $base(v_i) \neq base(v_j)$ the weight of P_k plus the weight of P_l is at most $d_{npv}(v_i, v_{i,1}) + d_{npv}(v_j, v_{j,1})$. With these results at hand one readily verifies that:

$$\begin{aligned} C(S) &= \sum_{v \in V_S} p_v \\ &\leq \left(\sum_{P_r \in \mathcal{P}^-} C(P'_r) \right) + C(P_k) + C(P_l) \\ &\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) + C(P_k) + C(P_l) \\ &\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) + d_{npv}(v_i, v_{i,1}) + d_{npv}(v_j, v_{j,1}). \end{aligned}$$

In the case of $t_b := base(v_i) = base(v_j)$ the combined weight of P_k and P_l is at most $\min\left\{d_{npv}(v_i, v_{i,1}) + d_{npv}(v_j, v_{j,2}), d_{npv}(v_i, v_{i,2}) + d_{npv}(v_j, v_{j,1})\right\}$, since by definition: $d_{npv}(t_b, v_i) = d_{npv}(v_i, v_{i,1})$ and $d_{npv}(t_b, v_j) = d_{npv}(v_j, t_{j,1}$. Consequently:

$$\begin{aligned} C(S) &\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) + C(P_k) + C(P_l) \\ &\leq \sum_{q=3}^{s_r} mwradius(t_{\varphi(q)}) \\ &\quad + \min\{d_{npv}(v_i, t_{i,1}) + d_{npv}(v_j, t_{j,2}), d_{npv}(v_i, v_{i,2}) + d(v_j, v_{j,1}\} \end{aligned}$$

Consquently, U as defined in (81) is an upper bound on the weight of S. □

We apply a **Bound** (**BND**) reduction test in line with heretofore approaches for bound-based reduction tests; i.e. by first computing an upper bound by means of a constructive, shortest path based primal heuristic and then attempting to eliminate vertices and edges.

In addition to the methods described formulated in this section, further reduction tests can be conceived, which have, owed to limited time, neither been particularised nor implemented for this thesis:

First, the bound-based reduction tests could be naturally extended to cover vertices of positive cost, as patterned by the bound-based lemmata to the PCSTP.

Second, one might utilize the bound-based tests not only to eliminate edges, but moreover to dispose of non-positive chains that could not be eliminated by the NPV_2 test: This approach would result in a test similar to the one affiliated with Lemma 52, but instead of the endpoints v_i and v_j of an edge, one would consider the endpoints of a non-positive chain. Moreover, a bound-based test comparable to the one associated with Lemma 13 in Section 2.2.4 could be used to prove that a vertex of both non-positive weight and small degree is of degree at most two in at least one optimal solution. This vertex could then possibly be eliminated by a simplified version of the NPV_k test, only considering "maximum spanning trees" for subsets of cardinality two.

Finally, it can be observed that the maximum-weight Voronoi decomposition, is potentially very unbalanced, e.g. in the case of one or two vertices being of comparatively high weight, most vertices might be assigned to the corresponding one or two Voronoi regions. Hence, a different decomposition leading to small $\sum_{q=3}^{s_r} mwradius(t_{\varphi(q)})$ values seems to be preferable.

Ordering

The settings for the reiteration of the reduction methods within the overall loop as well as for the truncation of the latter are as heretofore, see for example Section 2.2.5.

Naturally, empirically less expensive tests are performed first. In this way, the BND test, which requires the computation of an upper bound, is performed at the end of the loop. Moreover, all other tests are performed prior to the execution of the NPV_k test, in order to reduce the degree of vertices. The final ordering is illustrated in Figure 13.

3.5.3 Implementation and Computational Results

The computational settings of SCIP-Jack are identical to those of the PCSTP, except for the use of VQ, as the latter cannot so easily be adapted to handle anti-parallel arcs of different weight and is therefore disabled. Instead, we have implemented a simple improvement heuristic that attempts

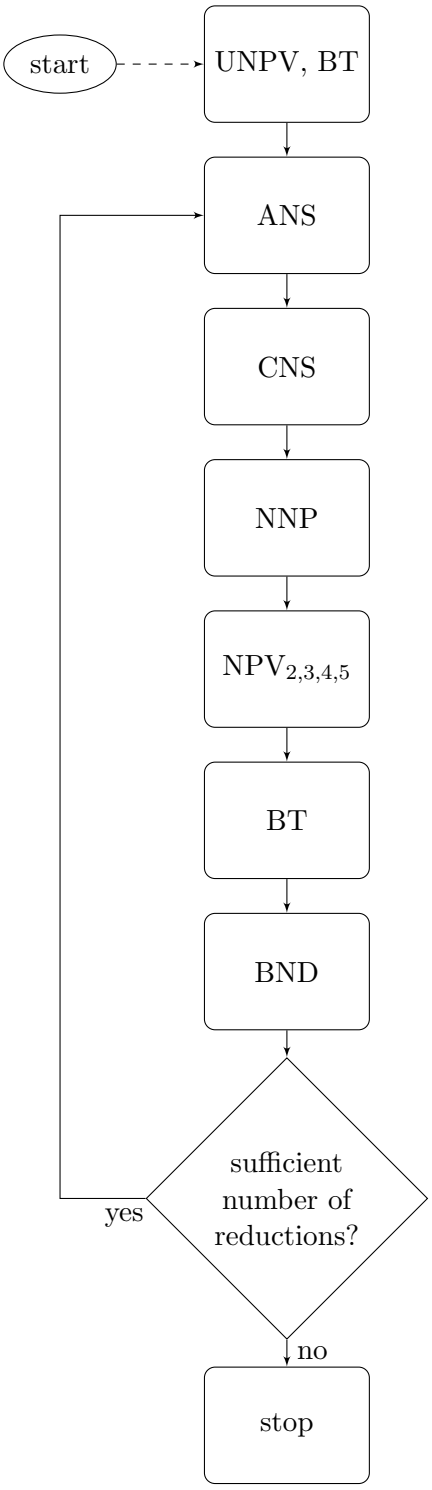


Figure 13: Illustration of the MWCSP reduction package.

to connect vertices of positive weight to the current Steiner tree using only one edge.

Name	Instances	$\|V\|$	Status	Description
ACTMOD	8	2034-5226	solved	Real-world instances derived from integrative biological network analysis [DKR+08].
JMPALMK	72	500-1500	solved	Euclidean, randomly generated instances introduced in [AMLM13].

Table 21: Classes of MWCSP instances.

For the computational experiments merely two classes of instances were available which vary widely in their scope, with the JMPALMK set containing nine times as many instances as the ACTMOD class, see Table 21.

Of the JMPALMK test set all but two instances are solved to optimality within one second, more than 90 percent of them even in less than 0.4 seconds. For the ACTMOD class more time is required, but SCIP-Jack still manages to solve all of them within 20 seconds. Neither the JMPALMK nor the ACTMOD instances require branching. The summarized results are provided in Table 22.

			Optimal	
Class	Instances	Solved	∅ Nodes	∅ Time [s]
ACTMOD	8	8	1.0	5.6
JMPALMK	72	72	1.0	0.3

Table 22: Computational results for MWCS instances.

Since almost all instances of the JMPALMK are solved during preprocessing, as will be demonstrated subsequently, and the ACTMOD instances are solved very fast, we refrain from presenting detailed results on the impact of the primal heuristics, but merely mark that without problem specific primal heuristics the time required for solving the ACTMOD instances is almost two times as high on average. In line with the experimental results to the heretofore introduced Steiner problem variants, the recombination heuristic does not improve the run time of the, easy, MWCSP test instances; on the contrary, the execution time is prolonged by four percent on average.

Impact of Reduction Techniques

Within the MWCS section, a considerable weight has been given to developing efficient reduction techniques. The results illustrated in Table 23 prove their strength. Not only 67 of all 72 JMPALMK instances are solved during presolving, but furthermore on average the ACTMOD instances contain less than a third of their original vertices and edges after reduction. We ascribe the considerably stronger performance on the JMPALMK class to

the higher proportion of terminals, which fosters several reduction methods. For instance, adjacent non-negative vertices are forthwith contracted by the NNIV test and also the CNS test becomes empirically more effective on problems with a higher percentage of non-negative vertices.

Class	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]
ACTMOD	33.2	27.0	0.8
JMPALMK	0.3	0.1	0.2

Table 23: Computational results of the MWCSP reduction package. The percentage of remaining edges and vertices is stated with respect to the arithmetic mean, the time is given as a shifted geometric mean (with $s = 1$).

Additionally, Table 24 displays the computational results obtained from iteratively excluding each reduction method from the otherwise unaltered package. It can be seen that the simple reduction methods (BT) are once again the most effective. However, the contextually almost astronomical number of more than 25,000 percent for the JMPALMK set should be regarded against the backdrop of only 0.05 percent remaining edges on average. Furthermore, BT notably improves the run time of the overall reduction package on the JMPALMK class. However, the run time results for the JMPALMK class should be interpreted cautiously, given the exceedingly small times. Second place is the CNS test, resulting in 38.2 percent more remaining edges for the ACTMOD and 104.4 for the JMPALMK class when excluded. Additionally, the overall run time on JMPALMK is considerably improved by CNS, while it slightly deteriorates on the ACTMOD set. Both NNP and NPV_k demonstrate a solid impact on the overall reduction, although the latter exhibits the worst time performance of all reduction methods on the ACTMOD class. In the same way, excluding the BND test would notably improve the total run time on ACTMOD, while its effect is quite limited. Last and least are the ANS and UNPV tests that do not achieve a measurable amount of reductions. While the UNPV test at least allows to eliminate a few additional edges on ACTMOD and furthermore improves the run time on JMPALMK, the ANS tests does not bring any positive results. This is caused by CNS dominating ANS on all instances. Therefore, ANS is disabled for the computational comparisons in Section 4.2.

In terms of reduction strength the performance of our package on the ACTMOD instances is about twice as good as the best known results reported in the literature [AB14]. This is illustrated in detail in Table 25: It can be seen that our reduction strategy is superior on each instance for both the number of reduced vertices and edges. In terms of reduced edges the difference is most pronounced for the drosophila instances, while the highest variation of reduced vertices can be found for the, very sparse, metabol-exprmice problems.

	Class	UNPV	BT	ANS	CNS	NNP	NPV_k	BND
Edges	ACTMOD	**+0.0**	**+48.0**	0	**+38.2**	**+34.7**	**+11.5**	**+0.2**
	JMPALMK	0	**+25982.2**	0	**+104.4**	**+9.1**	**+82.4**	0
Time	ACTMOD	-11.8	-2.6	-8.0	-4.4	-11.6	-32.5	-26.6
	JMPALMK	**+6.2**	**+275.9**	-2.0	**+98.4**	-1.4	**+0.1**	-2.3

Table 24: Computational results of the MWCSP presolving. Each column reports the results of the reduction package without the specified method. The values denote the percentual change with respect to the default settings (see Table 23).

For the JMPALMK test set we have not been able to find individual results concerning reduction techniques. However, for the combined ACTMOD and JMPALMK instances the value of 16.8 percent remaining vertices and 5.6 percent remaining edges is reported in [EK14]. As against to 3.8 and 2.7 percent of remaining vertices and edges, respectively, by our approach. No run times for the preprocessing were reported by the authors, but at least theoretically their reduction package is of higher cost, being laden with a worst case complexity of $O(n(m + n \log n))$.

	Original		Reduced		Reduced [AB14]	
Name	Nodes	Edges	Nodes	Edges	Nodes	Edges
drosophila001	5226	93394	2404	21947	2857	45802
drosophila005	5226	93394	2095	12110	2802	43859
drosophila0075	5226	93394	1894	9839	2738	40017
HCMV	3863	29293	1406	10816	2659	21414
lymphoma	2034	7756	1045	4467	1461	6895
metabol-expr-mice1	3523	4345	704	1135	1813	2741
metabol-expr-mice2	3514	4332	706	1148	1808	2738
metabol-expr-mice3	2853	3335	442	727	1175	1796
Avg. ratio			33.2%	27.0%	55.7%	60.1%

Table 25: Reductions on the ACTMOD instances

3.6 The Degree Constrained Steiner Tree Problem

The **degree-constrained Steiner tree problem** (**DCSTP**) is an SPG with an additional constraint: Given a Steiner tree problem in graphs represented by (V, E, T, c) and a function $b : V \to \mathbb{N}$, the objective is to find a minimum Steiner tree S such that $\delta_S(v) \leq b(v)$ is satisfied for all $v \in V[S]$.

A comprehensive discussion of the DCSTP, including its applications in biology, can be found in [LMP14].

3.6.1 Primal Heuristics

We use a variation of RSPH, altered in such a way that while choosing a new (shortest) path to be added to the current tree it is checked: First, whether attaching this path would violate any degree constraints and second, whether after having added this path at least one additional edge could be added (or all terminals are spanned). If no such path can be found, a vertex of the tree is pseudo randomly chosen that allows to add at least one adjacent edge, and such an edge leading to a vertex of high degree and being of small cost is chosen.

3.6.2 Implementation and Computational Results

The DCSTP is implemented by just adding the additional degree constraints for each node as linear constraints to the customarily employed formulation (Formulation 5). Problem specific reduction techniques have not been developed, neither by us nor in any publication we are aware of.

Computational experiments are performed on the 20 instances in the TreeFam test set of the DIMACS challenge, derived from a computational biology application. The size of the, invariably complete (i.e. containing the maximum amount of edges), instances ranges from a problem comprised of 52 vertices, 1326 edges and 35 terminals to one of 832 vertices, 345696 edges and 407 terminals. Furthermore, the degree bound for each vertex is less than or equal to three. The results are presented in Table 26.

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
TreeFam	20	7	162.8	40.6	413.9	25.9

Table 26: Computational results for DCSTP instances.

SCIP-Jack solves five instances to optimality and proves the infeasibility of another two. However, SCIP-Jack stops short of solving the remaining 13 instances within the time limit. The optimality gaps among these problems range from 0.2 to 133.3 percent This result is attributed to the lack of both reduction techniques and a more refined constructive heuristic.

However, without using the constructive heuristic the average gap among the unsolved instances is more than twice as high.

3.7 The Group Steiner Tree Problem

The **group Steiner tree problem** (**GSTP**) is another generalization of the Steiner tree problem, originating from VLSI design [HRW92], where the concept of terminals as a set of vertices to be interconnected is extended to a set of vertex groups: Given an undirected graph $G = (V, E)$, edge costs $c : E \to \mathbb{Q}_+$ and a series of vertex subsets $T_1, ..., T_s \subset V$, $s \in \mathbb{N}$, a minimum cost tree spanning at least one vertex of each subset is required. By interpreting each terminal t as a subset $\{t\}$, every SPG can be considered as a GSTP. Therefore, the latter is likewise NP-hard.

3.7.1 Transformation

Not only can an SPG be considered a GSTP, but also it is possible to transform each GTSP instance $(V, E, T_1, .., T_s, c)$ to an SPG by means of the following scheme:

Transformation 6 (GSTP to SPG)**.**
Input: *A GSTP* $P = (V, E, T_1, ...T_s, c)$
Output: *An SPG* $P' = (V', E', T', c')$

1. *Set* $V' := V$, $E' := E$, $T' = \emptyset$, $c' := c$, $K := \sum_{e \in E} c_e + 1$.
2. *For* $i = 1, ..., s$ *add a new node* t'_i *to* V' *and* T' *and for all* $v_j \in T_i$ *add an edge* $e = \{t'_i, v_j\}$ *, with* $c'_e := K$.
3. ***Return*** (V', E', T', c').

Let $(V, E, T_1, ...T_s, c)$ be a GSTP and $P' = (V', A', T', c')$ an SPG obtained by applying Transformation 6 on P. A solution S' to P' can be reduced to a solution S to P by deleting all vertices and edges of S not in (V, E). The GSTP P can in this way be solved on the SPG P' as shown in [HRW92] and [Vos88].

This approach has already been deployed by [DVV04] to solve group Steiner tree problems and demonstrated to be competitive with specialized solvers at the time of publishing.

3.7.2 Implementation and Computational Results

For SCIP-JACK to solve a GSTP, first Transformation 6 is applied. The resulting problem is treated as a normal SPG, which is solved without any alteration.

Computational results for two test sets of unpublished group Steiner tree instances derived from a industry wire routing problem are presented in Table 28. SCIP-JACK solves all but two of the first test set, with run times ranging from 1.8 to 434.1 seconds. Five of the instances solved to

Name	Instances	$\|V\|$	Status	Description
GSTP1	8	349-1253	unsolved	Sparse instances derived from a problem in VLSI design.
GSTP2	10	838-3177	unsolved	

Table 27: Classes of GSTP instances.

optimality only require a single node, with the remaining instance solved using 614 nodes. Two instances of this set are terminated at the time limit of two hours, exhibiting optimality gaps of 1.1 and 4.2 percent, respectively. The performance on the second test set is notably weaker. Five instances are solved within the time limit (with one requiring seven branch-and-bound nodes), whereas all other instances terminate with 54.7 branch-and-bound nodes on average. The best optimality gap among the unsolved instances of GSTP2 is 0.8, the worst 2.9 percent.

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
GSTP1	8	6	11.6	39.2	178.2	2.7
GSTP2	10	5	2.0	3196.0	54.7	1.6

Table 28: Computational results for GSTP instances.

3.8 The Hop Constrained Directed Steiner Tree Problem

The **hop-constrained directed Steiner tree problem** (**HCDSTP**) can be considered as an SAP incorporating additional conditions [BDK14]: First, the arc costs are required to be positive. Second, the additional constraint that the number of selected arcs must not exceed a predetermined bound H, called **hop limit**, is incorporated. Finally, all terminals except the root are necessarily leaves in each feasible solution. The flow-balance cut formulation (Formulation 5) used by SCIP-Jack is easily extended to cover this variation by adding one extra linear inequality bounding the sum of all binary arc variables and removing all incoming arcs to each terminal other than the root.

Still, the hop limit brings significant implications for the preprocessing and the heuristics in its wake: Many of the reduction techniques remove or include edges from the graph if a less costly alternative can be found, regardless of whether this alternative includes a greater number of edges and possibly violates the hop constraint. Likewise, the so far employed heuristics do not consider the number of included edges, but merely their costs.

Real-world applications of the HCDSTP include the three dimensional placement of drones for multi-target surveillance, see [OKD+10].

3.8.1 Reductions

For the HCDSTP no reduction techniques that we are aware of have been published yet. Several approaches that are effective for other Steiner problems, like the alternative-based reductions, are rendered prohibitive by the hop constraint. Therefore, this section encompasses merely bound-based reduction techniques. Subsequently, it will be supposed that an HCDSTP $P_{HC} = (V, A, T, c, r, H)$ is given. As with the SAP, the representation $T^r := T \setminus \{r\}$ is used and $r = t_1$ is presupposed.

Bound Based Reductions

Hereinafter, we reuse the concept of the rooted inward Voronoi diagram, introduced in Section 3.1.1. Additionally, for each $i = 2, ..., s$ we define $c^i_{min} := \min_{a \in \delta^-(t_i)} c_a$.

Lemma 53. *Let $v_i \in V \setminus T$ and define $t_b := base(v_i)$. If there is an optimal solution $S = (V_S, A_S)$ to P_{HC} such that $v_i \in V_S$, then*

$$\vec{d}(r, v_i) + \vec{d}(v_i, t_b) + \sum_{q=2}^{s} c^i_{min} - \max_{q=2,...,n} c^q_{min} \tag{100}$$

is a lower bound on the weight of S.

Proof. Since S is an arborescence rooted in r, it contains a path P_1 from r to v_i that is of cost at least $\vec{d}(r, v_i)$. Additionally, at least one path P_2 from v_i to a terminal t_j other than the root is contained in S, since the latter has been assumed to be optimal. P_2 is of cost at least $\vec{d}(v_i, base(v_i))$ and due to S being cycle free, P_1 and P_2 are arc-disjoint. Since for all $t_k \in T \setminus \{r, t_j\}$ it holds that $\delta_S^+(t_k) = \emptyset$, none of their incoming arcs can be contained in P_1 or P_2. However, as all of them are contained in S, for each terminal in $T \setminus \{r, t_j\}$ one of its incoming arcs has to be part of S as well. Hence:

$$\begin{aligned} C(S) &\geq C(P_1) + C(P_2) + \sum_{q=2, q\neq t_j}^{s} c^q_{min} \\ &\geq \vec{d}(r, v_i) + \vec{d}(v_i, t_j) + \sum_{q=2, q\neq t_j}^{s} c^q_{min} \\ &\geq \vec{d}(r, v_i) + \vec{d}(v_i, base(v_i)) + \sum_{q=2}^{s} c^i_{min} - \max_{q=2,\dots,n} c^q_{min} \end{aligned}$$

The last term is tantamount to (100), so the lemma is established. □

Not only the rooted inward Voronoi diagram, but also the associated lemmata can be adopted from the SAP. To this end, assume the setting for the lemmata 20 and 21 in Section 3.1.1.

Lemma 54. *Let $v_i \in V \setminus T$. If there is an optimal solution $S = (V_S, A_S)$ such that $v_i \in V_S$, then*

$$\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=2}^{s-2} inradius(t_q), \tag{101}$$

where the $\underline{d}$-distances are affiliated with $\overline{G}_D$, is a lower bound on the weight of S.

Proof. Analogous to the proof of Lemma 21. □

Lemma 55. *Let $(v_i, v_j) \in A$. If there is an optimal solution $S = (V_S, A_S)$ such that $(v_i, v_j) \in A_S$, then L defined by*

$$L := c_{ij} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=2}^{s-2} inradius(t_q) \tag{102}$$

if $v_{i,1} \neq v_{j,1}$ and

$$L := \min\left\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1})\right\} + \sum_{q=2}^{s-2} inradius(t_q) \tag{103}$$

otherwise, is a lower bound on the weight of S. The $\underline{d}$-distances are stated with respect to $\overline{G}_D$.

Proof. Analogous to the proof of Lemma 21. □

We denote the reduction test combining the antecedent three lemmata by **Cost Bound (CBND)** test: First, a solution S is computed by the heuristic introduced in Section 3.8.2. Second, we traverse each vertex $v_i \in V$. If v_i is a non-terminal and the bound (100) or (101) is higher than $C(S)$ (or equal in the case of $v_i \notin V_S$), the vertex v_i can be deleted. Otherwise, we traverse each outgoing arc $\delta^+(v_i)$ and attempt to eliminate it in the same way using the bounds (102) and (103) respectively.

While the idiosycracy of the HCDSTP in the shape of the hop constraint impedes numerous reduction techniques, it concomitantly gives rise to tests of a novel type. In the remainder of this section, denote by $\vec{d}_1$ the directed distance function with respect to the arc costs $\mathbb{1}$. Consequently, $\vec{d}_1(v_i, v_j)$ is equal to the minimal number of arcs necessary to reach the vertex v_j from the vertex v_i. Further, let N_1 be a rooted inward Voronoi diagram on (V, A) with respect to d_1.

Lemma 56. *Let $v_i \in V \setminus T$ and $t_b \in T^r$ such that $v_i \in N_1(t_b)$. If*

$$\vec{d}_1(r, v_i) + \vec{d}_1(v_i, t_b) + |T| - 2 > H \tag{104}$$

is satisfied, v_i cannot be contained in any optimal solution.

Proof. Let $S = (V_S, A_S)$ be an optimal solution and suppose $v_i \in V_S$. Analogously to the proof of Lemma 53 it can be verified that S contains arc-disjoint paths P_1, from r to v_i and P_2, from v_i to a $t_j \in T^r$. Considering the definition of d_1 one readily acknowledges that at least $\vec{d}(r, v_i)$ arcs are contained in P_1 and at least $\vec{d}_1(v_i, t_j)$ arcs are contained in P_2. Furthermore, both P_1 and P_2 contain exactly one terminal, since to all terminal other than the root there are no outgoing arcs. Noting that to each terminal in $T \setminus \{r, t_j\}$ there is at least one arc included in S, one obtains that

$$\begin{aligned} |A_s| &\geq \vec{d}_1(r, v_i) + \vec{d}_1(v_i, t_j) + |T| - 2 \\ &\geq \vec{d}_1(r, v_i) + \vec{d}_1(v_i, t_b) + |T| - 2 \\ &\overset{(104)}{>} H. \end{aligned}$$

Hence it can be inferred that, violating the hop constraint, S is not a feasible solution. □

Furthermore, by exploiting the hop constraint property, both Lemma 54 and Lemma 55 can be reshaped. To this end, denote for $t_i \in T^r$ by $inradius_1(t_i)$ the $inradius(t_i)$ value with respect to N_1 and assume that

the $t_i \in T^r$ are ordered in such a way that the corresponding $inradius_1$ values are non-decreasing in i. Further, the $\underline{d}_1$-distances are stated with respect to $\overline{G}_D$ with edge costs $\mathbb{1}$.

Lemma 57. *Let $v_i \in V \setminus T$. If*

$$\underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_i, v_{i,2}) + \sum_{q=2}^{s-2} inradius_1(t_q) > H \tag{105}$$

is satisfied, v_i cannot be contained in any optimal solution.

Proof. Assume that (105) is satisfied for a $v_i \in V \setminus T$. Using d_1 as the distance function d, one can be infer from Lemma 20 that for an optimal solution $S = (V_S, A_S)$ containing v_i it would hold that

$$|A_S| = C(S) \geq \underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_i, v_{i,2}) + \sum_{q=2}^{s-2} inradius_1(t_q) \overset{(105)}{>} H.$$

Therefore, no such $S = (V_S, A_S)$ can exist. □

Lemma 58. *Let $(v_i, v_j) \in A$. If*

$$1 + \underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_j, v_{j,1}) + \sum_{q=2}^{s-2} inradius_1(t_q) > H \tag{106}$$

in the case of $v_{i,1} \neq v_{j,1}$ or

$$1 + \min\left\{\underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_j, v_{j,2}), \underline{d}_1(v_i, v_{i,2}) + \underline{d}_1(v_j, v_{j,1})\right\} + \sum_{q=2}^{s-2} inradius_1(t_q) > H \tag{107}$$

otherwise, (v_i, v_j) cannot be contained in any optimal solution.

Proof. Let $(v_i, v_j) \in A$. Suppose that (v_i, v_j) was contained in an optimal solution $S = (V_S, A_S)$. Using Lemma 21 this would imply in the case of $v_{i,1} \neq v_{j,1}$ that:

$$|A_S| = C(S) \geq 1 + \underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_j, v_{j,1}) + \sum_{q=2}^{s-2} inradius_1(t_q) \overset{(106)}{>} H$$

and in the case of $v_{i,1} = v_{j,1}$ that:

$$\begin{aligned} |A_S| &= C(S) \\ &\geq 1 + \min\left\{\underline{d}_1(v_i, v_{i,1}) + \underline{d}_1(v_j, v_{j,2}), \underline{d}_1(v_i, v_{i,2}) + \underline{d}_1(v_j, v_{j,1})\right\} \\ &\qquad + \sum_{q=2}^{s-2} inradius_1(t_q) \\ &\overset{(107)}{>} H. \end{aligned}$$

Hence, no such S can exist and the lemma is established. □

Associated to the antecedent three lemmata is the **Hop Bound (HBND)** test: Successively, we process each vertex $v_i \in V$: If v_i is a non-terminal and the bound (104) or (105) is greater than the hop bound H, the vertex v_i is deleted. Otherwise, each outgoing arc $a_j \in \delta^+(v_i)$ is traversed and if the bound (106) or (107) respectively is greater than H, the arc is deleted. It should be noted that a predecessor of HBND was already introduced in [GKM+15].

3.8.2 Primal Heuristics

Similar to the presolving techniques, the heretofore introduced Steiner problem heuristics do not take into account the hop limit. As such, any identified solution may not be feasible. Therefore, a simple variation of RSPH, see Section 2.3, is used for the HCDSTP: Each arc a, having original costs c_a, is assigned the new cost $c'_a := 1 + \lambda \frac{c_a}{c_{max}}$, with $\lambda \in \mathbb{Q}_+$ and $c_{max} := \max_{a \in A} c_a$. Initially λ is set to 3 but its value is decreased or increased after each iteration of the constructive heuristic, depending on whether the last computed solution exceeds or is below the hop limit, respectively. This modification to λ is performed relatively to the deviation of the number of edges from the hop limit.

3.8.3 Implementation and Computational Results

Name	Instances	$\|V\|$	Status	Description
Gr12	19	809	solved	Prevalently dense instances with $\|T\| = 10$. Derived from 3D placement of unmanned aerial vehicles [BDK14].
Gr14	20	3209	unsolved	
Gr16	21	12509	unsolved	

Table 29: Classes of HCDSTP instances.

The three different test sets considered for the computational experiments, see Table 29, contain all instances used in the evaluation of the DIMACS challenge.

Of the Gr12 set SCIP-Jack is able to solve all instances in less than 435 seconds, 16 of them within 13 seconds. On average, the instances require 13.4 seconds of run time and 6.7 nodes, as illustrated in Table 30. Only two instances of this test set were not solved at the root node. The performance on the Gr14 test set is notably diminished, only ten of the instances can be solved to optimality within two hours. All of the unsolved instances terminate with large optimality gaps, ranging from 3.5 % to 40.6 %, after 102.1 nodes on average.

Finally, more than half of the Gr16 instances were terminated due to insufficient memory. Therefore, we resorted for this test set to a different

machine, being endowed with Intel Xeon E5-2697 CPUs with 2.70 GHz and 128 GB RAM. Although this machine can boast more RAM than the machines of the cluster used for the other experimental results of this thesis (see Section 2.6), it is also notably slower.

The results for the Gr16 test set are remarkably worse than for the other two sets: None of the instances can solved to optimality, with the lowest gap being 26.2 percent; the hardest one exhibits a gap of even 191.3 percent. However, it should be taken into consideration that the number of arcs of the instances ranges from more than 1.6 to almost nine million, far more than in most other instances in this thesis. Furthermore, only on four instances a branch-and-bound search was performed; on all of them the number of branching nodes stays below 20. We put this down to the fact that compared to the Gr12 and Gr16 instances far more time is required for computations at the root node, such as LP-solving.

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
Gr12	19	19	6.7	13.4	–	–
Gr14	21	10	60.7	462.9	102.1	18.4
Gr16	20	0	–	–	2.1	81.6

Table 30: Computational results for HCDSTP instances.

Impact of Reduction Techniques

The efficiency of the HCDSTP reduction package on the classes Gr12, Gr14 and Gr16 is presented in Table 31. In terms of effectiveness, the results on all three test sets are comparable, with an average of 69 and 71 percent of vertices and edges, respectively. In contrast, the execution times differ considerably, ranging from less than 0.1 seconds on Gr12 to more than seven seconds on Gr16. Responsible for this behavior is the computation of a primal solution for the CBND test. Indeed, this test occupies more than 95 percent of the run time for the Gr16 set.

Class	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]
Gr12	69.2	72.1	0.0
Gr14	67.2	66.7	0.6
Gr16	69.7	73.8	7.6

Table 31: Computational results of the HCDSTP reduction package. The percentage of remaining edges and vertices is stated with respect to the arithmetic mean, the time is given as a shifted geometric mean (with $s = 1$).

4 Final Computational Results and Comparisons

Although several computational results have been presented in the course of this work, they have been primarily focused on evaluating the performance of separate components and have therefore not been collated with other exact solving results from the literature. In contrast, in the following SCIP-Jack is compared on several Steiner variants with the best problem specific solvers described in the literature. To this end, the heretofore computational environment is used. These settings are identical to the ones that were used to evaluate the solvers competing in the 11th DIMACS Challenge, which allows for an equitable comparison with the results obtained there. In this context it should be noted that the results described in this thesis are significantly better than those obtained by the previous version of SCIP-Jack that was submitted to the DIMACS Challenge. However, we do not provide comparisons between the current and just mentioned version of SCIP-Jack, but merely report the results yielded by the latest version. Besides in the DIMACS Challenge, results for a previous version of SCIP-Jack can be found in [GKM$^+$15].

4.1 The Steiner Tree Problem in Graphs

First of all, it must be noted that SCIP-Jack is outperformed on three of the four heretofore used SPG test sets (see Section 2.6) by the currently best known SPG solver [PVD14] (an improved version of the program described in [Pol04]). A further notably strong, but overall inferior, SPG solver is introduced in [PUW14]. The results given in [PVD14] were obtained on a single-threaded PC with an Intel Core i7 920 of 2.66 GHz, using a Linux 3.4.2-1.fc16.x86_64 operating system and CPLEX 12.6. Using this setting, the authors manage to solve all 20 E instances in less than four seconds to optimality, 19 of them in less than one second (as opposed to more than five seconds on average required by SCIP-Jack, with more than 300 seconds for the e18 instance). Furthermore, they solve all I320 and vienna-I instances to optimality in at most 2915 and 624 seconds, respectively. However, of the PUC set only 12 instances are solved within two hours, while SCIP-Jack manages to solve nine instances in the same time.

To provide a broader picture of the performance of SCIP-Jack on the SPG, which has been covered in most detail compared to the other Steiner problem variants in thesis, computational results on additional test sets from the SteinLib are provided in Table 32.

			Optimal		Timeout	
Class	Instances	Solved	∅ Nodes	∅ Time [s]	∅ Nodes	∅ Gap [%]
D	20	20	1.0	0.6	–	–
X	3	3	1.0	1.7	–	–
SP	8	5	1.0	0.0	8.1	0.8
I640	100	72	27.3	47.5	120.5	0.8
LIN	37	23	1.0	10.6	1.0	21.1

Table 32: Computational results for additional SPG instances.

As a comparison, in [PVD14] all D and X instances can be solved to optimality. Furthermore, 95 of the I640 instances and 32 of the LIN instances can be solved within two hours. However, on the SP class SCIP-Jack manages to obtain better optimality gaps for the two instances w13c29 and w23c23 that could not be solved in [PVD14] (using a time limit of 24 hours as opposed to two hours for our experiments).

Instance	$\|V\|$	$\|E\|$	$\|T\|$	Best	SCIP-Jack	Instance	$\|V\|$	$\|E\|$	$\|T\|$	Best	SCIP-Jack
bip42p	1200	3982	200	24657	24657*	cc3-5u	125	750	13	36	36*
bip42u	1200	3982	200	236	236*	cc5-3p	243	1215	27	7299	7299*
bip52p	2200	7997	200	24535	**24526**	cc5-3u	243	1215	27	71	71*
bip52u	2200	7997	200	234	234	cc6-2p	64	192	12	3271	3271*
bip62p	1200	10002	200	22870	**22843**	cc6-2u	64	192	12	32	32*
bip62u	1200	10002	200	220	**219**	cc6-3p	729	4368	76	20456	**20270***
bipa2p	3300	18073	300	35379	**35326**	cc6-3u	729	4368	76	197	197*
bipa2u	3300	18073	300	341	**338**	cc7-3p	2187	15308	222	57088	57117
bipe2p	550	5013	50	5616	5616*	cc7-3u	2187	15308	222	552	552
bipe2u	550	5013	50	54	54*	cc9-2p	512	2304	64	17296	**17199**
cc10-2p	1024	5120	135	35379	**35227**	cc9-2u	512	2304	64	167	167*
cc10-2u	1024	5120	135	342	343	hc10p	1024	5120	512	60494	**59797**
cc11-2p	2048	11263	244	63826	**63636**	hc10u	1024	5120	512	581	**575**
cc11-2u	2048	11263	244	614	618	hc11p	2048	11264	1024	119779	**119689**
cc12-2p	4096	24574	473	121106	122099	hc11u	2048	11264	1024	1154	**1151**
cc12-2u	4096	24574	473	1179	1184	hc12p	4096	24576	2048	236949	**236080**
cc3-10p	1000	13500	50	12860	**12837**	hc12u	4096	24576	2048	2275	**2262**
cc3-10u	1000	13500	50	125	126	hc6p	64	192	32	4003	4003*
cc3-11p	1331	19965	61	15609	15648	hc6u	64	192	32	39	39*
cc3-11u	1331	19965	61	153	153	hc7p	128	448	64	7905	7905*
cc3-12p	1728	28512	74	18838	18997	hc7u	128	448	64	77	77*
cc3-12u	1728	28512	74	186	187	hc8p	256	1024	128	15322	15322*
cc3-4p	64	288	8	2338	2338*	hc8u	256	1024	128	148	148*
cc3-4u	64	288	8	23	23*	hc9p	512	2304	256	30258	**30242**
cc3-5p	125	750	13	3661	3661*	hc9u	512	2304	256	292	292

Table 33: Primal bound improvements on the PUC instances.

4.1.1 Parallel Computing

In the following, computational results of the parallel approach introduced in Section 2.5 are presented. Although only results for the unsolved instances of the PUC test set are provided, it should be noted that the parallel version of SCIP-Jack is capable of handling all 11 variants covered in this thesis; indeed, SCIP-Jack was able to take first place not only in the single-, but also in the multi-thread RPCSTP category of the DIMACS Challenge. Furthermore, we once more want to emphasize that the parallel results presented in this subsection were obtained in corporation with Yuji Shinano and are already published in [GKM+15].

Owned to limited availability, different clusters and supercomputers were used for the parallel computations. The largest of these computations comprised up to 864 cores, but only involved eight instances (`bip52p, bip62u, bipa2p, bipa2u, cc11-2p, cc12-2p, cc3-12p, hc9p`). In contrast, all other computations were conducted with 192 or less cores. As a reference to the scalability of ParaSCIP, the largest computation previously performed was an 80,000 cores run on Titan at ORNL [SAB+15]. We expect SCIP-Jack to also run on such a large scale computing environment, although at this stage only computational experiments of comparatively small scale have been conducted.

Table 33 shows the results on the instances of the PUC test set. We

list the number of vertices, edges and terminals, as well as the best primal bound known at the beginning of the challenge (August 2014), and the primal solution value obtained by our experiments with the parallel version of SCIP-Jack, which employs the LP solver of CPLEX 12.6. Prior to the experiments performed with SCIP-Jack, 32 instances of the PUC test set remained unsolved. Three of these instances have been solved by SCIP-Jack to proven optimality. These instances are underlined and marked with an asterisk in Table 33. For a further 16 instances SCIP-Jack improved the best known solution. All instances for which the best known primal bound has been improved are marked in bold. Finally, all previously solved instances of the PUC test set have also been solved by SCIP-Jack to proven optimality; signified by an asterisk (without underline).

In summary, the heretofore reported results demonstrate an overall strong performance of the parallel version of SCIP-Jack in solving computationally difficult instances.

4.2 Variants of the Steiner Tree Problem

In the following, for several Steiner problem variants comparisons are drawn between the results presented in this thesis and the best known ones from the literature. For this purpose, results obtained in the course of the 11th DIMACS Challenge are used predominantly. However, also other results reported in the literature are provided. Once again, we want to point out that in our thesis an improved version of SCIP-Jack is used as compared to the one that competed in the DIMACS Challenge. Preliminarily, some remarks on the Steiner problem variants not covered hereinafter are necessary:

For the Steiner arborescence problem merely very easy instances were available, solvable in fractions of a second. Therefore, comparisons with other results from the literature do not seem reasonable.

For the node-weighted Steiner tree problem only two instances have been covered in this thesis (and published in the course of the DIMACS Challenge); unfortunately we could not find any publication reporting results for these instances.

The rectilinear Steiner minimum tree instances of dimension two can be solved by the geometric Steiner problem solver GeoSteiner, which was already mentioned in Section 3.3. As described in the same section, more advanced modifications would be necessary to render SCIP-Jack competitive with this solver. Nevertheless, we are not aware of any other solver being able to handle the high dimensional RMST instances that can be solved by SCIP-Jack as demonstrated in Section 3.3.2.

Finally, the group Steiner tree problem instances have not been made publicly available yet and can therefore not be used for computational comparisons.

To the best of our knowledge, for all Steiner problem variants covered hereinafter no results published in the literature exceed those achieved in the course of the DIMACS Challenge.

4.2.1 The Prize Collecting Steiner Tree Problem

Both the Cologne1 and Cologne2 class were used as test sets in the DIMACS Challenge. In the following two tables we compare our results with the best known other ones. These were achieved by the solver *Mozartballs* [FLL$^+$14]. The best results reported for the RPCSTP instances outside the DIMACS Challenge can be found in [LWP$^+$06]. Requiring more than 40,000 seconds of run time for several instances, these results are clearly inferior to the ones of the best two groups competing in the challenge.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Optimum	SCIP-Jack t [s]	Mozartballs t [s]
i101M1	758	12704	11	**109271.503**	0.2	1.0
i101M2	758	12704	11	**315925.31**	3.4	20.0
i101M3	758	12704	11	**355625.409**	5.0	23.0
i102M1	760	12730	12	**104065.801**	0.2	1.0
i102M2	760	12730	12	**352538.819**	6.4	26.0
i102M3	760	12730	12	**454365.927**	7.9	42.0
i103M1	764	12738	14	**139749.407**	0.2	1.0
i103M2	764	12738	14	**407834.228**	3.6	16.0
i103M3	764	12738	14	**456125.488**	5.8	29.0
i104M2	744	12598	4	**89920.8353**	0.3	6.0
i104M3	744	12598	4	**97148.789**	1.0	12.0
i105M1	744	12604	4	**26717.2025**	0.2	1.0
i105M2	744	12604	4	**100269.619**	2.1	8.0
i105M3	744	12604	4	**110351.163**	4.2	15.0

Table 34: Comparison on the Cologne1 test set between SCIP-Jack and Mozartballs.

The results presented in Table 34 demonstrate the superiority of SCIP-Jack on the Cologne1 instances: Each instance is solved at least thrice as fast as by Mozartballs. For the i104M2 instance the execution time of Mozartballs is even 20 times higher than that of SCIP-Jack.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Optimum	SCIP-Jack t [s]	Mozartballs t [s]
i201M2	1812	33522	10	**355467.684**	0.6	16.0
i201M3	1812	33522	10	**628833.614**	23.2	126.0
i201M4	1812	33522	10	**773398.303**	47.6	170.0
i202M2	1814	33520	11	**288946.832**	6.2	15.0
i202M3	1814	33520	11	**419184.159**	31.4	63.0
i202M4	1814	33520	11	**430034.264**	23.8	109.0
i203M2	1824	33584	16	**459894.776**	3.5	14.0
i203M3	1824	33584	16	**643062.02**	51.9	255.0
i203M4	1824	33584	16	**677733.067**	54.2	211.0
i204M2	1805	33454	5	**161700.545**	0.3	8.0
i204M3	1805	33454	5	**245287.203**	24.6	27.0
i204M4	1805	33454	5	**245287.203**	28.6	36.0
i205M2	1823	33640	14	**571031.415**	19.0	12.0
i205M3	1823	33640	14	**672403.143**	36.8	46.0
i205M4	1823	33640	14	**713973.623**	26.4	56.0

Table 35: Comparison on the Cologne2 test set between SCIP-Jack and Mozartballs.

The same pattern continues on the Cologne2 class as evidenced in Table 35: Although one instance, i205M2, is solved faster by Mozartballs, on the remainder of the test set SCIP-Jack clearly prevails.

For the unrooted prize-collecting Steiner tree problem, unfortunately, none of our test sets was fully covered in the DIMACS Challenge. However, from each of the three test sets four instances were included. For all CRR as well as for all JMP instances (C13-A, D19-B, D03-B, D20-A and

P400-3, P400-4, K400-7, K400-10) our solver prevails against all participating groups, being on average more than twice as fast as the respective best competitor. The results for the four included PUCNU instances are somewhat lesser: One instance, cc3-12nu, is solved to optimality by SCIP-Jack, while no other group manages to reduce the optimality gap to less than two percent. For another one, cc12-2nu, SCIP-Jack achieves the smallest optimality gap. Yet, the other two instances, bip52-nu and bip62-nu, are solved by Mozartballs to optimality, but not by SCIP-Jack (nor any other group).

4.2.2 The Maximum Weight Connected Subgraph Problem

In this subsection SCIP-Jack is compared with the best groups participating in the DIMACS Challenge in the category MWCS. For this purpose, in Table 36 the results of both SCIP-Jack and its nemesis Mozartballs (reemerging this time in an MWCS variant) are presented. Notably, Mozartballs was the fastest solver for each ACTMOD instance.

Instance	$\|V\|$	$\|A\|$	$\|T\|$	Optimum	SCIP-Jack t [s]	Mozartballs t [s]
drosophila001	5298	187214	72	**24.3855064**	16.2	21.6
drosophila005	5421	187952	195	**178.663952**	17.6	15.5
drosophila0075	5477	188288	251	**260.523557**	8.5	10.5
HCMV	3919	58916	56	**7.55431486**	5.4	2.2
lymphoma	2102	15914	68	**70.1663087**	1.9	0.4
metabol_expr_mice_1	3674	9590	151	**544.94837**	2.0	2.7
metabol_expr_mice_2	3600	9174	86	**241.077524**	0.9	1.9
metabol_expr_mice_3	2968	7354	115	**508.260877**	0.8	1.1

Table 36: Comparison on the ACTMOD test set between SCIP-Jack and the winner of the MWCS category of the 11th DIMACS Challenge.

It can be seen that SCIP-Jack is faster on five out of the eight ACTMOD instances. However, one might see the differences as being moderate, considering that all instances can be solved in less than half a minute.

We do not provide detailed information on the JMPALMK class since only a subset of it was part of the DIMACS Challenge. Notwithstanding, on 15 of the 20 JMPALMK instances tested in the course of the Challenge SCIP-Jack proves to be faster than Mozartballs, which is in turn faster on only two. Additionally, on 11 of the instances SCIP-Jack exceeds all participants of the Challenge, whereas it is outperformed on only four (by the solver *Heinz* described in [EK14]).

Summarizing, the results suggest that SCIP-Jack is superior or at least competitive to state-of-the-art solvers for the MWCSP on the available benchmark sets.

4.2.3 The Degree Constrained Steiner Tree Problem

All DCSTP instances described in Section 3.6.2 were used in the DIMACS Challenge. Furthermore, experimental results on these instances are reported in [LMP14], where a DCSTP specific solver is introduced. The computations were performed on a "compute server with 12-core Opteron processors and 64GB RAM" [LMP14]. If one excepts the differences in the machines used for the computations, our results are superior for all but one instance.

A comparison with the best results obtained in the DIMACS Challenge is provided in Table 37.

				SCIP-Jack		Mozartballs	
Instance	$\|V\|$	$\|E\|$	$\|T\|$	Gap %	t [s]	Gap %	t [s]
TF101057-t1	52	1326	35	0.0	0.0	0.0	0.3
TF101057-t3	52	1326	35	0.0	46.7	0.0	0.7
TF101125-t1	304	46056	155	0.0	4.6	0.0	14.0
TF101125-t3	304	46056	155	3.7	7200	0.0	3194.1
TF101202-t1	188	17578	72	2.2	7200	0.0	131.3
TF101202-t3	188	17578	72	0.6	7200	0.0	87.2
TF102003-t1	832	345696	407	102.0	7200	>100	7200
TF102003-t3	832	345696	407	7.6	7200	62.2	7200
TF105035-t1	237	27966	104	32.8	7200	0.0	407.4
TF105035-t3	237	27966	104	1.6	7200	0.0	1181.3
TF105272-t1	476	113050	223	113.1	7200	>100	7200
TF105272-t3	476	113050	223	8.2	7200	10.5	7200
TF105419-t1	55	1485	24	0.0	161.4	0.0	1.1
TF105419-t3	55	1485	24	0.0	5.9	0.0	2.6
TF105897-t1	314	49141	133	62.4	7200	0.0	3206.9
TF105897-t3	314	49141	133	2.5	7200	51.6	7200
TF106403-t1	119	7021	46	0.0	493.3	0.0	15.5
TF106403-t3	119	7021	46	0.0	64.4	0.0	71.3
TF106478-t1	130	8385	54	0.4	7200	0.0	20.5
TF106478-t3	130	8385	54	0.2	7200	0.0	37.5

Table 37. Comparison on the TreeFam test set between SCIP-Jack and Mozartballs.

The results display an overall stronger performance of Mozartballs: While SCIP-Jack yields smaller optimality gap on three instances (TF102003-t3, TF105272-t3 and TF105897-t3) and is faster one another three (TF101057-t1, TF101125-t1 and TF106403-t3), Mozartballs obtains smaller optimality gaps on eight and better time results on additional four problems.

Notwithstanding the better performance of Mozartballs, SCIP-Jack is by no means outranked and has also achieved better results than the specialized solver described in [LMP14]; against the backdrop of the limited efforts having been made for the DCSTP in this thesis, the comparative performance of SCIP-Jack certainly bears mentioning.

4.2.4 The Hop Constrained Directed Steiner Tree Problem

For the HCDSTP, besides SCIP-Jack only several versions of the specialized primal heuristic framework *Stephop*, see [BDK14], competed in the

DIMACS Challenge. Therefore, no detailed information is provided here. Nevertheless, it should not go unnoticed that for the 20 instances of the Gr12 and Gr14 sets being part of the Challenge, the upper bounds computed by SCIP-Jack within two hours are equal for three and better for 13 instances, as compared to the results achieved by the competing heuristics.

4.3 Using SoPlex

Having established a powerful Steiner problem solver able to handle a notable number of variants, we still perceive that a certain blemish lingers: Using the commercial LP solver CPLEX, we cannot equitably claim that SCIP-Jack is genuinely non-commercial and available in source code. Therefore, in this subsection results are provided for employing SoPlex (see Section 2.5) instead.

The results of using SoPlex 2.2.0 as the underlying LP solver of SCIP-Jack are presented in Table 38. In each line of the table aggregated results for the test set specified in the first column are provided. The second column denotes the Steiner problem type that the current test set belongs to. In the last three columns, the percentual change of the number of solved instance, the required branch-and-bound nodes and finally the execution time is stated, all with respect to the results obtained by using SCIP-Jack/CPLEX.

For the evaluation of SCIP-Jack/SoPlex we have singled out those six Steiner problem variants on which the main focus has been set throughout this thesis. The results reveal an overall deterioration of both the number of solved instances and, more conspicuously, the execution time. Although the run time is comparable on the, easy, SAP instances, the solving process on all other non-trivial test sets is prolonged. Most notable is the JMP test set, which requires more than two times the execution time it consumed when CPLEX was being used. The number of solved instances decreases on four test sets, increases for one (PUCNU) and remains unchanged for the rest. Interestingly, although two more instances can be solved on the PUCNU test set using SoPlex, the average run time is more than 40 percent higher. With respect to the branch-and-bound nodes the results are overall balanced.

Overall, the results demonstrate that although slower than SCIP-Jack combined with CPLEX, SCIP-Jack/SoPlex still constitutes a powerful, fast solver that allows to solve 443 out of 555 benchmark instances within two hours.

Class	Type	Solved Instances	Nodes	Time
E	SPG	–	+32.2	+30.2
I320		-3.2	-6.7	+16.2
vienna-I		-14.0	+46.0	+44.8
PUC		-11.1	-48.6	+10.9
Gene	SAP	–	–	+4.7
Gene2002		–	–	-9.9
JMP	PCSTP	–	–	+120.0
CRR		–	–	+32.0
PUCNU		+20.0	+41.8	+42.8
Cologne1	RPCSTP	–	–	+52.3
Cologne2		–	–	+63.7
JMPALMK	MWCSP	–	–	-26.9
ACTMOD		–	+21.4	+10.3
Gr12	HCDSTP	–	+50.1	+25.4
Gr14		-30.0	-83.2	+65.1

Table 38: Computational results of SCIP-Jack/SoPlex. In each line the percentual changes with respect to the results obtained by using CPLEX are reported.

5 Conclusions and Outlook

In this thesis an approach has been presented that allows to solve 11 different Steiner problems within a single framework, standing in stark contrast to the problem specific strategy of other state-of-the-art Steiner tree solvers. This achievement has been rendered possible by employing various, often novel, methods to transform the problems into a general form, which can be solved by a branch-and-cut based procedure using the academic MIP framework SCIP. Two advantages of employing this generic approach are immediately conspicuous:

First, the amount of problem specific code is notably reduced. While for the SPG around 20,000 lines of code (with more than 10 percent being comments) were needed, each additional problem variant required only about 500 lines on average. As a comparison, the pure SPG solver described in [Pol04] is reported to encompass "roughly thousand pages" of code, which amounts to about 60,000 lines of code taking the size of each page in [Pol04] as a reference.

Second, a number of general solution methods natively implemented in SCIP, e.g. cutting planes (being notably effective for several hard instances such as in the PUC class), are available and will be kept up-to-date automatically by the continuous improvements in the framework.

Moreover, not entirely in line with the overall generic approach, various novel reduction techniques have been introduced that surpass existing methods for several Steiner problem variants such as the maximum-weight connected subgraph problem. Likewise, several heuristics have been devised, although they can all be seen as variations of three basic versions, the RSP, VQ and RC heuristics. Albeit both the reduction techniques and, partly, the heuristics are problem specific, they easily assimilate within our general framework, also by virtue of the plugin-based structure of SCIP.

In this way, the combination of the generic branch-and-cut based approach and problem specific techniques has enabled us to achieve our initial goal to be able to handle a large variety of Steiner problem variants, but to allow at the same time for fast solving: The exact solver SCIP-Jack (the offspring of embedding the pure SPG solver Jack-III into SCIP) accompanying this work is capable of solving several problem classes, e.g. the (real-world) RPCSTP instances, faster than any other solver we are aware of. Furthermore, it should be marked that the opportunity to solve instances in a massively parallel distributed memory environment has been added at minimal cost.

Still, there surely is potential for further improving the performance of the solver. Concerning the Steiner tree problem in graphs, additional methods described in the literature, most notably in [Pol04], such as dual heuristics and additional reduction tests are expected to bear a high capability for improving the run time of SCIP-Jack.

Concerning the Steiner problem variants, there are many heuristic approaches that can be employed to specific variants. Likewise, we expect that the development of additional efficient reduction techniques would exert a notable impact on the execution time; some suggestions for such extensions have already been provided in the course of this thesis, e.g. to the maximum-weight connected subgraph problem. Using the plugin structure of SCIP, we hope to include some of these heuristics and reduction techniques in the future. Also, SCIP-Jack could be extended to encompass further variants described in the literature, such as the *hop-constrained minimum spanning tree problem* which can be reduced to a Steiner tree problem in graphs [GSU11].

Summarizing, pursuing our generic approach we have succeeded in incorporating a variety of novel and known techniques to a general-purpose Steiner problem framework. The accompanying solver is competitive or even superior to specialized state-of-the-art programs for several Steiner problem variants. Moreover, we deem it important to point out that, to the best of our knowledge, this has been the first time a powerful exact solver for non-geometric Steiner problems is available in source code to the scientific community: The SCIP Optimization Suite already contains a previous version of our solver and we furthermore plan to publish the current version of SCIP-Jack, presented in this thesis, as part of the next release of SCIP. We hope that the availability of such a device will foster the use of Steiner trees in modelling real-world phenomena as has already been the case for instance in genetics [JKC+00].

Ultimately, we have succeeded in creating a powerful exact solving framework of unprecedented versatility that can veritably be designated as a Steiner *class* solver.

References

[AB09] Tobias Achterberg and Timo Berthold. Hybrid Branching. In Willem Jan van Hoeve and John N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009*, volume 5547 of *Lecture Notes in Computer Science*, pages 309–311. Springer, May 2009.

[AB14] Ernst Althaus and Markus Blumenstock. Algorithms for the Maximum Weight Connected Subgraph and Prize-collecting Steiner Tree Problems. Unpublished manuscript at http://dimacs11.cs.princeton.edu/workshop.html, 2014.

[ABCC11] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, New Jersey, 2011.

[Ach07] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[Ach09] Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[AKM04] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33:42–54, 2004.

[AMLM13] Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel. *The maximum weight connected subgraph problem.*, pages 245–270. Berlin: Springer, 2013.

[Ane80] Yash P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10(2):167–178, 1980.

[BDK14] Oleg Burdakov, Patrick Doherty, and Jonas Kvarnström. Local Search for Hop-constrained Directed Steiner Tree Problem with Application to UAV-based Multi-target Surveillance. In Butenko, S., Pasiliao, E.L., Shylo, and V., editors, *Examining Robustness and Vulnerability of Networked Systems*, volume 37 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 26–50. IOS Press, 2014.

[Bea84] J. E. Beasley. An algorithm for the Steiner problem in graphs. *Networks*, 14(1):147–159, 1984.

[Bea89] J.E. Beasley. An SST-based algorithm for the Steiner problem in graphs. *Networks*, 19:1–16, 1989.

[Bea92] J.E. Beasley. A heuristic for Euclidean and rectilinear Steiner problems. *European Journal of Operational Research*, 58:284–292, 1992.

[BGG+12] Timo Berthold, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Thorsten Koch, and Yuji Shinano. Solving mixed integer linear and nonlinear problems using the SCIP Optimization Suite. Technical Report 12-27, ZIB, Takustr.7, 14195 Berlin, 2012.

[BGRS13] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner Tree Approximation via Iterative Randomized Rounding. *The Journal of the ACM*, 60(1):6, 2013.

[BGTZ14] M. Brazil, R. L. Graham, D. A. Thomas, and M. Zachariasen. On the History of the Euclidean Steiner Tree Problem. *Archive for History of Exact Sciences*, 68:327–354, 2014.

[BP87] Anantaram Balakrishnan and Nitin R. Patel. Problem reduction methods and a tree generation algorithm for the Steiner network problem. *Networks*, 17(1):65–85, 1987.

[BP89] Marshall W. Bern and Paul E. Plassmann. The Steiner Problem with Edge Lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

[BW05] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers.* Athena Scientific, 2005.

[CC08] Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008.

[CCC+98] Moses Charikar, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation Algorithms for Directed Steiner Problems. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 192–200, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[CGSW14] Krzysztof Ciebiera, Piotr Godlewski, Piotr Sankowski, and Piotr Wygocki. Approximation Algorithms for Steiner Tree Problems Based on Universal Solution Frameworks. *Computing Research Repository*, abs/1410.7534, 2014.

[CR94] Sunil Chopra and M. Rao. The Steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(2):209–229, 1994.

[CSHH+13] Salim Akhter Chowdhury, Stanley Shackney, Kerstin Heselmeyer-Haddad, Thomas Ried, Alejandro A. Schäffer, and Russell Schwartz. Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. *Bioinformatics*, 29(13):189–198, 2013.

[dAW02] Marcus Poggi de Aragao and Renato F. Werneck. On the Implementation of MST-based Heuristics for the Steiner Problem in Graphs. In *Proceedings of the 4th International Workshop on Algorithm Engineering and Experiments*, pages 1–15. Springer, 2002.

[DHK14] Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Philip N. Klein. Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs. *ACM Transactions on Algorithms*, 10(3):13:1–13:20, 2014.

[DKR+08] Marcus T. Dittrich, Gunnar W. Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. In *ISMB*, pages 223–231, 2008.

[Dui93] C. Duin. *Steiner Problems in Graphs.* PhD thesis, University of Amsterdam, 1993.

[DV89] C. W. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19(5):549–567, 1989.

[DVV04] C. W. Duin, A. Volgenant, and S. Voss. Solving group Steiner problems as Steiner problems. *European Journal of Operational Research*, 154(1):323–329, 2004.

[EK14] Mohammed El-Kebir and Gunnar W. Klau. Solving the Maximum-Weight Connected Subgraph Problem to Optimality. *Computing Research Repository*, abs/1409.5308, 2014.

[Ema10] Nahit Emanet. *The Rectilinear Steiner Tree Problem.* Lambert Academic Publishing, 2010.

[ES03] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003.

[Fis91] M. Fischetti. Facets of two Steiner arborescence polyhedra. *Mathematical Programming*, 51:401–419, 1991.

[FLL+14] Matteo Fischetti, Markus Leitner, Ivana Ljubic, Martin Luipersbeck, Michele Monaci, Max Resch, Domenico Salvagnin, and Markus Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. Unpublished manuscript at http://dimacs11.cs.princeton.edu/workshop.html, 2014.

[FT84] Michael L. Fredman and Robert Endre Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. In *FOCS*, pages 338–346. IEEE Computer Society, 1984.

[GHNP01] Clemens Gröpl, Stefan Hougardy, Till Nierhoff, and Hans Jürgen Prömel. Approximation algorithms for the Steiner tree problem in graphs. In *Steiner trees in industry*, volume 11 of *Combinatorial Optimization*, pages 235–279. Kluwer Acad. Publ., Dordrecht, 2001.

[GJ77] M.R. Garey and D.S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

[GK99] Sudipto Guha and Samir Khuller. Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets. *Information and Computation*, 150(1):57–74, 1999.

[GKM+15] Gerald Gamrath, Thorsten Koch, Stephen Maher, Daniel Rehfeldt, and Yuji Shinano. SCIP-Jack - A solver for STP and variants with parallelization extensions. Technical Report 15-27, ZIB, Takustr.7, 14195 Berlin, 2015.

[GM90] Martin Grötschel and Clyde L. Monma. Integer Polyhedra Arising from Certain Network Design Problems with Connectivity Constraints. *SIAM Journal on Discrete Mathematics*, 3(4):502–523, 1990.

[GM93] Michel X. Goemans and Young-Soo Myung. A catalog of Steiner tree formulations. *Networks*, 23(1):19–28, 1993.

[GSU11] Luis Gouveia, Luidi Simonetti, and Eduardo Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Mathematical Programming*, 128(1-2):123–148, 2011.

[GT88] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. *The Journal of the ACM*, 35(4):921–940, 1988.

[Hak71] S. Louis Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1(2):113–133, 1971.

[Han66] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal of Applied Mathematics*, 14(2):255–265, 1966.

[Hau] Karpinski Hauptmann. A Compendium on Steiner Tree Problems. `http://theory.cs.uni-bonn.de/info5/Steinerkompendium/`. Accessed: Sep. 18, 2015.

[Hel] Keld Helsgaun. Helsgaun's Lin-Kernighan heuristic. `http://www.akira.ruc.dk/~keld/research/LKH/`. Accessed: Sep. 18, 2015.

[Hel09] Keld Helsgaun. General k-opt submoves for the Lin-Kernighan TSP heuristic. *Mathematical Programming Computation*, 1(2-3):119–163, 2009.

[Hin08] Michalewicz Hingston, Barone, editor. *Design by Evolution*. Natural computing series. Springer, New York, 2008.

[HK03] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In Lawrence L. Larmore and Michel X. Goemans, editors, *STOC*, pages 585–594. ACM, 2003.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem*. Annals of Discrete Mathematics. Elsevier Science, 1992.

[JKC+00] J.J. Johnston, R.I. Kelley, T.O. Crawford, D.H. Morton, R. Agarwala, T. Koch, A.A. Schäffer, C.A. Francomano, and L.G. Biesecker. A novel nemaline myopathy in the Amish caused by a mutation in troponin T1. *American Journal of Human Genetics*, pages 814–821, October 2000.

[JMP00] David S. Johnson, Maria Minkoff, and Steven Phillips. The Prize Collecting Steiner Tree Problem: Theory and Practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 760–769, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KLM+04] Gunnar W. Klau, Ivana Ljubic, Andreas Moser, Petra Mutzel, Philipp Neuner, Ulrich Pferschy, Günther R. Raidl, and René

Weiskircher. Combining a Memetic Algorithm with Integer Programming to Solve the Prize-Collecting Steiner Tree Problem. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell, editors, *GECCO (1)*, volume 3102 of *Lecture Notes in Computer Science*, pages 1304–1315. Springer, 2004.

[KM98] Thorsten Koch and Alexander Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

[KMV01] Thorsten Koch, Alexander Martin, and Stefan Voß. SteinLib: An updated library on Steiner tree problems in graphs. In D.-Z. Du and X. Cheng, editors, *Steiner Trees in Industries*, pages 285–325. Kluwer, 2001.

[Koc95] Thorsten Koch. Jack-III: Ein Branch & Cut-Verfahren zur Lösung des gewichteten Steinerbaumproblems in Graphen. Master's thesis, Technische Universität Berlin, 1995.

[KV07] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007.

[Lju04] Ivana Ljubić. *Exact and Memetic Algorithms for Two Network Design Problems*. PhD thesis, Vienna University of Technology, 2004.

[LLL+14] M. Leitner, I. Ljubic, M. Luipersbeck, M. Prossegger, and M. Resch. New Real-world Instances for the Steiner Tree Problem in Graphs. Technical report, ISOR, Uni Wien, 2014.

[LMP14] Frauke Liers, Alexander Martin, and Susanne Pape. Steiner Trees with Degree Constraints: Structural Results and an Exact Solution Approach. Technical report, Department Mathematik, 2014.

[LR04] Abilio Lucena and Mauricio G. C. Resende. Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1-3):277–294, 2004.

[LWP+06] Ivana Ljubic, René Weiskircher, Ulrich Pferschy, Gunnar W. Klau, Petra Mutzel, and Matteo Fischetti. An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem. *Mathematical Programming*, 105(2-3):427–449, 2006.

[Meh88] Kurt Mehlhorn. A Faster Approximation Algorithm for the Steiner Problem in Graphs. *Information Processing Letters*, 27(3):125–128, 1988.

[Mit02] J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press, January 2002.

[Mit15] Hans Mittelmann. Benchmarks for optimization software, Accessed Sep. 18, 2015. `http://plato.asu.edu/bench.html`.

[MR01] Anna Moss and Yuval Rabani. Approximation algorithms for constrained for constrained node weighted Steiner tree problems. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *STOC*, pages 373–382. ACM, 2001.

[MW95] Thomas L. Magnanti and Laurence A. Wolsey. *Handbooks in Operations Research and Management Science*, volume Volume 7, chapter Chapter 9 Optimal trees, pages 503–615. Elsevier, 1995.

[OKD+10] Per-Magnus Olsson, Jonas Kvarnström, Patrick Doherty, Oleg Burdakov, and Kaj Holmberg. Generating UAV communication networks for monitoring and surveillance. In *In Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV*, 2010.

[PD01] Tobias Polzin and Siavash Vahdati Daneshmand. Improved Algorithms for the Steiner Problem in Networks. *Discrete Appl. Math.*, 112(1-3):263–300, September 2001.

[Pol04] Tobias Polzin. *Algorithms for the Steiner problem in networks.* PhD thesis, Saarland University, 2004.

[PS02] Hans-Jürgen Prömel and Angelika Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity.* Advanced Lectures in Mathematics. Vieweg, 2002.

[PUW14] Thomas Pajor, Eduardo Uchoa, and Renato F. Werneck. A Robust and Scalable Algorithm for the Steiner Problem in Graphs. *Computing Research Repository*, 2014.

[PVD14] Tobias Polzin and Siavash Vahdati-Daneshmand. The Steiner Tree Challenge: An updated Study. Unpublished manuscript at http://dimacs11.cs.princeton.edu/downloads.html, 2014.

[RdAR+01] I. Rosseti, M.P. de Aragão, C.C. Ribeiro, E. Uchoa, and R.F. Werneck. New benchmark instances for the Steiner problem in graphs. In *Extended Abstracts of the 4th Metaheuristics International Conference (MIC'2001)*, pages 557–561, Porto, 2001.

[Reh] Daniel Rehfeldt. SCIP-JACK documentation. `http://scip.zib.de/doc/applications/STP/`. Accessed: Sep. 18, 2015.

[RUW01] Celso C. Ribeiro, Eduardo Uchoa, and Renato F. Werneck. A Hybrid Grasp With Perturbations For The Steiner Problem In Graphs. *Informs Journal on Computing*, 14:200–2, 2001.

[SAB+12] Yuji Shinano, Tobias Achterberg, Timo Berthold, Stefan Heinz, and Thorsten Koch. ParaSCIP: a parallel extension of SCIP. In Christian Bischof, Heinz-Gerd Hegering, Wolfgang Nagel, and Gabriel Wittum, editors, *Competence in High Performance Computing 2010*, pages 135 – 148, 2012.

[SAB+15] Yuji Shinano, Tobias Achterberg, Timo Berthold, Stefan Heinz, Thorsten Koch, and Michael Winkler. Solving Open MIP Instances with ParaSCIP on Supercomputers using up to 80,000 Cores. Technical Report 15-53, ZIB, Takustr.7, 14195 Berlin, 2015.

[Seg87] Arie Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

[SHVW13] Yuji Shinano, Stefan Heinz, Stefan Vigerske, and Michael Winkler. FiberSCIP - a shared memory parallelization of SCIP. Technical Report 13-55, ZIB, Takustr.7, 14195 Berlin, 2013.

[Sny92] Timothy Law Snyder. On the Exact Location of Steiner Points in General Dimension. *SIAM Journal of Applied Mathematics*, 21(1):163–180, 1992.

[TA80] H. Takahashi and Mastsuyama A. An approximate solution for the Steiner problem in graphs. *Mathematica Japonicae*, 24:573 – 577, 1980.

[Tar79] Robert Endre Tarjan. Applications of Path Compression on Balanced Trees. *The Journal of the ACM*, 26(4):690–715, October 1979.

[Uch06] Eduardo Uchoa. Reduction Tests for the Prize-collecting Steiner Problem. *Oper. Res. Lett.*, 34(4):437–444, July 2006.

[UW10] Eduardo Uchoa and Renato Fonseca F. Werneck. Fast Local Search for Steiner Trees in Graphs. In Guy E. Blelloch and

Dan Halperin, editors, *ALENEX*, pages 1–10. SIAM Journal of Applied Mathematics, 2010.

[VD04] Siavash Vahdati-Daneshmand. *Algorithmic approaches to the Steiner problem in networks.* PhD thesis, University of Mannheim, 2004.

[Vos88] S. Voss. A survey on some generalizations of Steiner's problem. *1st Balkan Conference on Operational Research Proceedings*, 1:41–51, 1988.

[VW10] François Vanderbeck and Laurence A. Wolsey. Reformulation and Decomposition of Integer Programs. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming*, pages 431–502. Springer, 2010.

[Wik] Wikipedia. Skipjack tuna. `https://en.wikipedia.org/wiki/Skipjack_tuna`. Accessed: Sep. 18, 2015.

[Won84] R.T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

[Wun96] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus.* PhD thesis, Technische Universität Berlin, 1996.

[WWZ00] 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, 2000.

[Zac99] Martin Zachariasen. Rectilinear full Steiner tree generation. *Networks*, 33(2):125–143, 1999.

[ZR00] M. Zachariasen and A. Rohe. Rectilinear group Steiner trees and applications in VLSI design. Technical Report 00906, Institute for Discrete Mathematics, 2000.

A Zusammenfassung (German Summary)

Das Steinerbaumproblem in Graphen ist ein klassisches NP-schweres Optimierungsproblem. Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph mit Kantengewichten $c : E \to \mathbb{Q}_+$ und einer Menge $T \subset G$ von *Terminalen*. Gesucht ist ein Baum in G, der alle Terminale enthält und die Summe seiner Kantengewichte minimiert.

Ungeachtet der Prädominanz des klassischen Steinerbaumproblems in der Literatur findet sich dieses in praktischen Anwendungen meist in abgewandelter Form wieder. Obwohl diese Variationen untereinander eine starke Verwandtschaft aufweisen, sind moderne (exakte) Lösungsanansätze überwiegend problemspezifisch. Dem entgegengestellt wird in dieser Arbeit ein generisches Konzept eingeführt, das schließlich das exakte Lösen von 11 verschiedenen Steinerbaum Varianten mit Hilfe eines einzigen Programms ermöglicht. Dies wird realisiert durch die Transformation der verschiedenen Varianten in eine allgemeine Form. Im Zuge der Umsetzung dieses Ansatzes wurde der Steinerbaumlöser Jack-III in das MIP-Framework SCIP eingebettet und grundlegend erweitert. Neben den Transformationen liegt eine wesentliche Erweiterung in neu entwickelten Reduktionstechniken und Heuristiken für eine Vielzahl der behandelten Probleme.

Nach einer Einführung in Abschnitt 1 behandelt Abschnitt 2 einen Ansatz zum exakten Lösen des Steinerbaumproblems in Graphen, der gleichzeitig die Grundlage für die behandelten Variationen bildet. Im Wesentlichen lassen sich hier drei Komponenten unterscheiden:

- Reduktionen, die eine Reduzierung der Größe (Anzahl an Knoten, Kanten und Terminalen) des Problems ermöglichen.
- Primale Heuristiken, die das Berechnen guter zulässiger Lösungen erleichtern.
- Branch and Cut, welches das exakte Lösen eines Problems erlaubt.

In Abschnitt 3 wird der beschriebene Ansatz auf 10 Variationen des klassischen Steinerbaumproblems übertragen. Hierzu werden diese mit Hilfe von Transformationen oder zusätzlicher (linearer) Nebenbedingungen auf eine Form gebracht, die vom generischen Branch and Cut Ansatz gelöst werden kann. Im Zuge dessen wird weiterhin demonstriert, dass die neu entwickelten Reduktionstechniken existierenden Methoden überlegen sind, siehe etwa Abschnitt 3.5.

Abschließend erfolgt in Abschnitt 4 ein Vergleich des entwickelten Lösers SCIP-Jack auf mehreren Steinerbaum Varianten mit den jeweils besten bekannten problemspezifischen Programmen. Wenngleich SCIP-Jack für das klassischen Steinerbaumproblem in Graphen dem besten Löser auf den meisten Problemklassen unterlegen ist, erzielt er für mehrere Varianten,

am deutlichsten für das RPCSTP, bessere Ergebnisse als alle bekannten Löser. Dies ist umso bemerkenswerter vor dem Hintergrund der generischen Ausrichtung des Lösungsansatzes. Weiterhin wird eine Parallelisierung von SCIP-Jack vorgestellt, mit deren Hilfe drei Benchmark Instanzen erstmalig gelöst und die besten bekannten oberen Schranken für mehrere weitere verbessert werden konnten. Schließlich wird gezeigt, dass der akademische LP-Löser SoPlex anstelle des zuvor verwendeten kommerziellen CPLEX verwendet werden kann. Dies erlaubt das Lösen von Steinerbaumproblemen ohne Verwendung kommerzieller Programme.

B Notation and Abbreviations

B.1 General Mathematical Notation

Symbol	Definition
$\mathbb{N}$	the natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$
$\mathbb{Z}$	the integers $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$
$\mathbb{Z}_{\geq 0}$	the non-negative integers $\mathbb{Z}_{\geq 0} = \mathbb{N} \cup \{0\}$
$\mathbb{Q}$	the rational numbers
$\mathbb{Q}_{\geq 0}$	the non-negative rational numbers: $\mathbb{Q}_{\geq 0} = \{x \in \mathbb{Q} \mid x \geq 0\}$
$\mathbb{Q}_{+}$	the positive rational numbers: $\mathbb{Q}_{+} = \{x \in \mathbb{Q} \mid x > 0\}$
$\mathbb{R}$	the real numbers
$\mathbb{R}_{\geq 0}$	the non-negative real numbers: $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$
$\mathbb{R}_{+}$	the positive real numbers: $\mathbb{R}_{+} = \{x \in \mathbb{R} \mid x > 0\}$

B.2 Steiner Problem Variants

Abbreviation	Name
DCSTP	Degree-constrained Steiner tree problem
GSTP	Group Steiner tree problem
HCDSTP	Hop-constrained directed Steiner tree problem
MWCSP	Maximum-weight connected subgraph problem
NWSTP	Node-weighted Steiner tree problem
PCSTP	Prize-collecting Steiner tree problem
RPCSTP	Rooted PCSTP
RSMTP	Rectilinear Steiner minimum tree problem
OARSMTP	Obstacle-avoiding RSMTP
SAP	Steiner arborescence problem
SPG	Steiner tree problem in graphs

C Detailed Experimental Results

Hereinafter, detailed computational results on the instance sets discussed in the course of this thesis are provided.

The subsequent tables are consistently structured as follows: First, the name of the respective instance is stated. In the next three columns the number of vertices, arcs and terminals of the instance is itemised. In this regard, it should be noted that in the case of problem classes that are solved by using a graph transformation, only the number of vertices and arcs in the final model are reported. Following, the segment labelled "Presolved" provides the core values of the reduced problem along with the reduction time.

Finally, in the last segment the dual and primal bound or the optimal solution value is given, respectively. Moreover, the number of branch-and-bound nodes (N) and the total run time is listed. A time-out is signified by a "$>$" in front of the termination time. Furthermore, opposed to other publications, the reported final execution time includes the preprocessing. The time limit for each instance is two hours.

C.1 The Steiner Tree Problem in Graphs

Table 39. Detailed computational results for the **D** test set.

	Original			Presolved						
Instance	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
d01	1000	2500	5	128	502	5	0.1	**106**	1	0.2
d02	1000	2500	10	208	764	10	0.1	**220**	1	0.2
d03	1000	2500	167	28	96	17	0.0	**1565**	1	0.0
d04	1000	2500	250	22	62	14	0.0	**1935**	1	0.0
d05	1000	2500	500	8	24	6	0.0	**3250**	1	0.0
d06	1000	4000	5	195	894	5	0.2	**67**	1	0.4
d07	1000	4000	10	345	1494	10	0.6	**103**	1	0.8
d08	1000	4000	167	178	602	74	0.4	**1072**	1	0.8
d09	1000	4000	250	80	258	48	0.2	**1448**	1	0.2
d10	1000	4000	500	27	106	20	0.0	**2110**	1	0.1
d11	1000	10000	5	64	248	5	0.4	**29**	1	0.4
d12	1000	10000	10	314	1418	9	0.6	**42**	1	0.8
d13	1000	10000	167	28	90	18	0.8	**500**	1	0.8
d14	1000	10000	250	111	400	55	0.4	**667**	1	0.6
d15	1000	10000	500	11	34	8	0.1	**1116**	1	0.1
d16	1000	50000	5	123	644	5	0.4	**13**	1	0.4
d17	1000	50000	10	631	4442	9	0.5	**23**	1	1.2
d18	1000	50000	167	822	5102	94	0.5	**223**	1	2.7
d19	1000	50000	250	702	4010	96	0.5	**310**	1	1.0
d20	1000	50000	500	0	0	0	0.7	**537**	1	0.7

Table 40. Detailed computational results for the **E** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
e01	2500	6250	5	11	34	4	0.2	**111**	1	0.2
e02	2500	6250	10	221	874	9	0.3	**214**	1	0.5
e03	2500	6250	417	135	432	80	0.1	**4013**	1	0.1
e04	2500	6250	625	109	496	63	0.1	**5101**	1	0.1
e05	2500	6250	1250	11	32	8	0.1	**8128**	1	0.1
e06	2500	10000	5	293	1486	5	0.8	**73**	1	1.0
e07	2500	10000	10	1233	6182	10	1.2	**145**	1	2.3
e08	2500	10000	417	537	1966	198	0.5	**2640**	1	2.0
e09	2500	10000	625	214	722	124	0.3	**3604**	1	0.7
e10	2500	10000	1250	52	202	36	0.2	**5600**	1	0.2
e11	2500	25000	5	293	1474	5	0.7	**34**	1	0.8
e12	2500	25000	10	2455	19838	10	1.0	**67**	1	6.4
e13	2500	25000	417	1076	4500	251	1.2	**1280**	1	12.7
e14	2500	25000	625	282	1092	140	0.6	**1732**	1	1.0
e15	2500	25000	1250	24	72	16	0.3	**2784**	1	0.3
e16	2500	125000	5	74	306	5	0.9	**15**	1	0.9
e17	2500	125000	10	1920	16818	10	1.9	**25**	1	5.5
e18	2500	125000	417	2071	14138	245	1.6	**564**	45	388.7
e19	2500	125000	625	1200	5856	174	5.7	**758**	2	10.4
e20	2500	125000	1250	0	0	0	17.1	**1342**	1	17.1

Table 41. Detailed computational results for the **I320** test set.

Instance	Original			Presolved				Dual	Primal	Gap %	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]					
i320-001	320	960	8	89	398	8	0.0		**2672**		1	0.1
i320-002	320	960	8	114	480	8	0.0		**2847**		1	0.1
i320-003	320	960	8	162	636	8	0.1		**2972**		1	0.3
i320-004	320	960	8	87	388	8	0.1		**2905**		1	0.1
i320-005	320	960	8	146	612	8	0.0		**2991**		1	0.2
i320-011	320	3690	8	320	3686	8	0.1		**2053**		7	1.5
i320-012	320	3690	8	319	3690	8	0.1		**1997**		1	0.8
i320-013	320	3690	8	320	3690	8	0.1		**2072**		1	1.6
i320-014	320	3690	8	320	3690	8	0.1		**2061**		15	2.2
i320-015	320	3690	8	319	3688	8	0.1		**2059**		9	1.8
i320-021	320	102080	8	320	5006	8	1.2		**1553**		1	2.8
i320-022	320	102080	8	320	5008	8	1.2		**1565**		1	2.5
i320-023	320	102080	8	320	5006	8	1.2		**1549**		1	1.9
i320-024	320	102080	8	320	5006	8	1.4		**1553**		1	2.4
i320-025	320	102080	8	320	5006	8	1.2		**1550**		1	2.0
i320-031	320	1280	8	201	1032	8	0.0		**2673**		7	0.9
i320-032	320	1280	8	232	1116	8	0.1		**2770**		3	0.8
i320-033	320	1280	8	137	726	8	0.1		**2769**		1	0.2
i320-034	320	1280	8	184	900	8	0.1		**2521**		1	0.1
i320-035	320	1280	8	127	680	8	0.0		**2385**		1	0.1
i320-041	320	20416	8	320	19926	8	0.3		**1707**		1	11.7
i320-042	320	20416	8	320	19802	8	0.3		**1682**		1	7.2
i320-043	320	20416	8	319	17174	8	0.3		**1723**		39	9.4
i320-044	320	20416	8	320	18976	8	0.3		**1681**		1	9.2
i320-045	320	20416	8	320	19666	8	0.3		**1686**		1	4.4
i320-101	320	960	17	147	590	16	0.0		**5548**		1	0.1
i320-102	320	960	17	154	602	15	0.0		**5556**		1	0.4
i320-103	320	960	17	154	602	17	0.1		**6239**		1	0.1
i320-104	320	960	17	152	602	17	0.0		**5703**		1	0.2

cont. next page

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
i320-105	320	960	17	158	618	16	0.0	**5928**			1	0.6
i320-111	320	3690	17	320	3690	17	0.1	**4273**			19	5.2
i320-112	320	3690	17	320	3690	17	0.1	**4213**			125	16.3
i320-113	320	3690	17	320	3690	17	0.1	**4205**			57	10.0
i320-114	320	3690	17	320	3690	17	0.1	**4104**			13	4.2
i320-115	320	3690	17	319	3688	17	0.0	**4238**			5	2.5
i320-121	320	102080	17	320	101844	17	1.1	**3321**			1	73.7
i320-122	320	102080	17	320	101840	17	1.3	**3314**			1	75.9
i320-123	320	102080	17	320	101842	17	1.1	**3332**			1	84.3
i320-124	320	102080	17	320	101840	17	1.4	**3323**			1	89.0
i320-125	320	102080	17	320	101846	17	1.3	**3340**			1	82.7
i320-131	320	1280	17	250	1132	17	0.1	**5255**			1	1.3
i320-132	320	1280	17	249	1126	15	0.0	**5052**			1	0.9
i320-133	320	1280	17	240	1108	16	0.0	**5125**			1	1.4
i320-134	320	1280	17	241	1118	17	0.1	**5272**			1	0.5
i320-135	320	1280	17	254	1144	17	0.0	**5342**			11	2.2
i320-141	320	20416	17	320	20386	17	0.3	**3606**			139	121.9
i320-142	320	20416	17	320	20400	17	0.2	**3567**			9	24.1
i320-143	320	20416	17	320	20388	17	0.3	**3561**			13	29.3
i320-144	320	20416	17	320	20378	17	0.3	**3512**			1	12.0
i320-145	320	20416	17	320	20382	17	0.3	**3601**			121	105.0
i320-201	320	960	34	155	592	33	0.1	**10044**			1	0.3
i320-202	320	960	34	168	638	31	0.0	**11223**			1	1.5
i320-203	320	960	34	156	606	32	0.0	**10148**			1	0.2
i320-204	320	960	34	161	624	33	0.0	**10275**			1	0.7
i320-205	320	960	34	148	560	29	0.0	**10573**			1	0.1
i320-211	320	3690	34	320	3690	34	0.1	**8039**			110	42.9
i320-212	320	3690	34	320	3690	34	0.1	**8044**			107	35.7
i320-213	320	3690	34	320	3686	34	0.1	**7984**			82	56.9
i320-214	320	3690	34	319	3688	34	0.1	**8046**			281	150.5
i320-215	320	3690	34	319	3684	34	0.0	**8015**			752	236.7
i320-221	320	102080	34	320	101050	34	1.2	**6679**			13	340.4
i320-222	320	102080	34	320	101036	34	1.3	**6686**			13	389.0
i320-223	320	102080	34	320	101032	34	1.5	**6695**			39	719.7
i320-224	320	102080	34	320	101036	34	1.2	**6694**			25	494.6
i320-225	320	102080	34	320	101030	34	1.4	**6691**			27	578.5
i320-231	320	1280	34	243	1116	32	0.0	**9862**			3	3.4
i320-232	320	1280	34	245	1120	34	0.0	**9933**			15	6.2
i320-233	320	1280	34	245	1122	34	0.0	**9787**			1	0.8
i320-234	320	1280	34	242	1110	34	0.0	**9517**			1	2.2
i320-235	320	1280	34	249	1126	34	0.0	**9945**			1	2.2
i320-241	320	20416	34	320	20240	34	0.3	**7027**			279	766.8
i320-242	320	20416	34	320	20276	34	0.3	**7072**			527	932.7
i320-243	320	20416	34	320	20268	34	0.3	**7044**			426	974.0
i320-244	320	20416	34	320	20232	34	0.3	**7078**			479	791.5
i320-245	320	20416	34	320	20224	34	0.3	**7046**			167	449.7
i320-301	320	960	80	155	566	57	0.1	**23279**			1	1.2
i320-302	320	960	80	151	542	53	0.0	**23387**			1	1.0
i320-303	320	960	80	166	620	60	0.0	**22693**			1	1.6
i320-304	320	960	80	139	534	45	0.0	**23451**			1	1.4
i320-305	320	960	80	137	508	56	0.0	**22547**			3	1.2
i320-311	320	3690	80	320	3648	80	0.1	**17945**			5518	2894.0
i320-312	320	3690	80	320	3608	80	0.1	**18122**			8700	3694.0
i320-313	320	3690	80	320	3600	80	0.1	**17991**			4769	2243.2

cont. next page

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
i320-314	320	3690	80	320	3624	80	0.0	**18088**			18252	7015.6
i320-315	320	3690	80	320	3642	80	0.1	**17987**			5495	3049.8
i320-321	320	102080	80	320	95950	80	1.2	**15648**			57	2714.1
i320-322	320	102080	80	320	95962	80	1.1	**15646**			165	5353.1
i320-323	320	102080	80	320	95940	80	1.2	**15654**			89	3715.9
i320-324	320	102080	80	320	95974	80	1.2	15663.0782	15667	0.0	306	>7210.2
i320-325	320	102080	80	320	95960	80	1.4	**15649**			61	4297.7
i320-331	320	1280	80	251	1094	73	0.0	**21517**			125	20.3
i320-332	320	1280	80	248	1102	74	0.0	**21674**			9	3.5
i320-333	320	1280	80	259	1138	75	0.0	**21339**			20	5.5
i320-334	320	1280	80	255	1130	76	0.0	**21415**			1	2.5
i320-335	320	1280	80	254	1130	76	0.0	**21378**			25	8.7
i320-341	320	20416	80	320	19338	80	0.3	16219.9271	16296	0.5	2174	>7210.0
i320-342	320	20416	80	320	19352	80	0.3	**16228**			843	3203.7
i320-343	320	20416	80	320	19332	80	0.5	16236.7206	16281	0.3	2042	>7210.0
i320-344	320	20416	80	320	19304	80	0.5	16234.8508	16302	0.4	1724	>7210.0
i320-345	320	20416	80	320	19354	80	0.9	16220.03	16289	0.4	1831	>7210.0

Table 42. Detailed computational results for the **I640** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
i640-001	640	1920	9	277	1166	9	0.2	**4033**			1	0.6
i640-002	640	1920	9	170	798	9	0.1	**3588**			1	0.5
i640-003	640	1920	9	271	1158	9	0.2	**3438**			1	0.3
i640-004	640	1920	9	300	1230	9	0.2	**4000**			1	1.2
i640-005	640	1920	9	278	1230	9	0.2	**4006**			1	0.7
i640-011	640	8270	9	639	8270	9	0.2	**2392**			1	2.2
i640-012	640	8270	9	640	8270	9	0.3	**2465**			5	4.8
i640-013	640	8270	9	640	8270	9	0.3	**2399**			1	3.6
i640-014	640	8270	9	640	8270	9	0.3	**2171**			1	1.7
i640-015	640	8270	9	640	8270	9	0.2	**2347**			19	3.9
i640-021	640	408960	9	640	11374	9	6.6	**1749**			1	16.1
i640-022	640	408960	9	640	11374	9	6.0	**1756**			1	13.5
i640-023	640	408960	9	640	11374	9	6.1	**1754**			1	14.5
i640-024	640	408960	9	640	11374	9	6.9	**1751**			1	14.7
i640-025	640	408960	9	640	11374	9	7.0	**1745**			1	14.0
i640-031	640	2560	9	462	2228	9	0.2	**3278**			1	0.6
i640-032	640	2560	9	462	2224	9	0.1	**3187**			1	0.3
i640-033	640	2560	9	447	2236	9	0.1	**3260**			1	1.1
i640-034	640	2560	9	458	2224	9	0.1	**2953**			1	0.6
i640-035	640	2560	9	470	2198	9	0.1	**3292**			1	0.7
i640-041	640	81792	9	640	80302	9	1.4	**1897**			1	60.9
i640-042	640	81792	9	640	80570	9	1.6	**1934**			107	154.9
i640-043	640	81792	9	640	80142	9	1.9	**1931**			111	158.8
i640-044	640	81792	9	640	79876	9	2.0	**1938**			157	199.7
i640-045	640	81792	9	562	62656	9	1.8	**1866**			1	45.9
i640-101	640	1920	25	321	1262	25	0.1	**8764**			1	2.3
i640-102	640	1920	25	311	1230	25	0.1	**9109**			1	0.8
i640-103	640	1920	25	305	1230	24	0.1	**8819**			1	1.4
i640-104	640	1920	25	303	1222	23	0.1	**9040**			1	1.6
i640-105	640	1920	25	324	1270	25	0.1	**9623**			7	5.8
i640-111	640	8270	25	640	8270	25	0.1	**6167**			291	115.7
i640-112	640	8270	25	640	8270	25	0.1	**6304**			91	95.0

cont. next page

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
i640-113	640	8270	25	640	8270	25	0.1	**6249**			361	226.1
i640-114	640	8270	25	640	8270	25	0.2	**6308**			152	100.9
i640-115	640	8270	25	640	8270	25	0.2	**6217**			426	213.0
i640-121	640	408960	25	640	408416	25	5.5	**4906**			3	1441.3
i640-122	640	408960	25	640	408416	25	5.4	**4911**			7	1615.5
i640-123	640	408960	25	640	408414	25	5.4	**4913**			15	1592.6
i640-124	640	408960	25	640	408414	25	5.4	**4906**			13	1807.0
i640-125	640	408960	25	640	408422	25	5.3	**4920**			23	2008.0
i640-131	640	2560	25	481	2234	25	0.2	**8097**			1	3.2
i640-132	640	2560	25	480	2226	24	0.1	**8154**			5	4.1
i640-133	640	2560	25	482	2236	25	0.1	**8021**			1	2.5
i640-134	640	2560	25	485	2244	25	0.1	**7754**			1	3.0
i640-135	640	2560	25	480	2228	25	0.1	**7696**			11	3.3
i640-141	640	81792	25	640	81714	25	1.5	**5199**			485	2224.2
i640-142	640	81792	25	640	81722	25	1.7	**5193**			375	2369.6
i640-143	640	81792	25	640	81728	25	1.7	**5194**			189	1250.3
i640-144	640	81792	25	640	81716	25	1.7	**5205**			340	1917.4
i640-145	640	81792	25	640	81726	25	1.9	**5218**			1299	2694.7
i640-201	640	1920	50	309	1230	46	0.1	**16079**			1	1.2
i640-202	640	1920	50	319	1242	48	0.1	**16324**			1	1.9
i640-203	640	1920	50	325	1272	47	0.1	**16124**			1	4.0
i640-204	640	1920	50	323	1266	48	0.1	**16239**			1	2.6
i640-205	640	1920	50	327	1270	48	0.1	**16616**			5	3.1
i640-211	640	8270	50	640	8270	50	0.1	11904.4959	12016	0.9	5971	>7200.0
i640-212	640	8270	50	640	8270	50	0.1	**11795**			2360	2226.9
i640-213	640	8270	50	640	8268	50	0.1	**11879**			6350	4615.0
i640-214	640	8270	50	640	8270	50	0.1	11843.1152	11898	0.5	5396	>7200.1
i640-215	640	8270	50	640	8262	50	0.1	12030.7071	12081	0.4	9017	>7200.0
i640-221	640	408960	50	640	406624	50	5.5	9788.06153	9821	0.3	23	>7201.3
i640-222	640	408960	50	640	406634	50	5.5	9770.69979	9798	0.3	15	>7205.6
i640-223	640	408960	50	640	406624	50	5.6	9774.94465	9813	0.4	8	>7204.3
i640-224	640	408960	50	640	406626	50	5.5	9778.96187	9805	0.3	15	>7202.5
i640-225	640	408960	50	640	406634	50	5.5	9775.42781	9810	0.4	10	>7203.4
i640-231	640	2560	50	492	2260	50	0.2	**15014**			183	59.8
i640-232	640	2560	50	494	2264	49	0.2	**14630**			17	13.0
i640-233	640	2560	50	508	2288	47	0.2	**14797**			21	26.1
i640-234	640	2560	50	485	2230	48	0.2	**15203**			1	5.3
i640-235	640	2560	50	484	2244	50	0.1	**14803**			210	157.7
i640-241	640	81792	50	640	81396	50	1.2	10161.1383	10252	0.9	193	>7200.0
i640-242	640	81792	50	640	81410	50	1.4	10163.1505	10195	0.3	537	>7200.0
i640-243	640	81792	50	640	81412	50	1.6	10167.8849	10237	0.7	166	>7200.0
i640-244	640	81792	50	640	81362	50	1.6	10166.136	10246	0.8	188	>7200.3
i640-245	640	81792	50	640	81424	50	1.5	10166.7729	10234	0.7	211	>7200.0
i640-301	640	1920	160	328	1202	123	0.1	**45005**			1	1.9
i640-302	640	1920	160	296	1136	110	0.1	**45736**			1	5.5
i640-303	640	1920	160	342	1260	125	0.2	**44922**			1	1.2
i640-304	640	1920	160	327	1206	126	0.1	**46233**			1	4.6
i640-305	640	1920	160	301	1110	115	0.1	**45902**			1	5.4
i640-311	640	8270	160	640	8068	160	0.1	35365.8747	36108	2.1	1256	>7200.1
i640-312	640	8270	160	639	8064	160	0.1	35356.474	36176	2.3	1250	>7200.0
i640-313	640	8270	160	640	8086	160	0.1	35237.914	35751	1.5	1351	>7200.1
i640-314	640	8270	160	640	8076	160	0.1	35165.0635	35849	1.9	1361	>7200.1
i640-315	640	8270	160	640	8060	160	0.1	35367.3911	36016	1.8	1335	>7200.0
i640-321	640	408960	160	640	383902	160	5.8	30991.7511	31301	1.0	1	>7201.5

cont. next page

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
i640-322	640	408960	160	640	383916	160	5.3	30985.7443	31084	0.3	2	>7210.9
i640-323	640	408960	160	640	383890	160	5.2	30997.8703	31272	0.9	2	>7200.1
i640-324	640	408960	160	640	383934	160	5.3	30998.2933	31163	0.5	2	>7202.2
i640-325	640	408960	160	640	383924	160	5.4	30987.4202	31244	0.8	2	>7203.1
i640-331	640	2560	160	488	2206	145	0.2	**42796**			100	65.9
i640-332	640	2560	160	504	2250	152	0.2	**42548**			250	153.2
i640-333	640	2560	160	502	2230	147	0.2	**42345**			1088	386.4
i640-334	640	2560	160	512	2280	155	0.2	**42768**			4852	1756.1
i640-335	640	2560	160	516	2292	153	0.1	**43035**			1016	371.4
i640-341	640	81792	160	640	77124	160	1.6	31861.9808	32101	0.8	28	>7200.3
i640-342	640	81792	160	640	76946	160	1.6	31825.0182	32050	0.7	29	>7200.0
i640-343	640	81792	160	640	77018	160	1.6	31835.4551	32060	0.7	27	>7200.8
i640-344	640	81792	160	640	77240	160	1.6	31836.2263	32058	0.7	33	>7200.7
i640-345	640	81792	160	640	77140	160	1.5	31831.0111	32029	0.6	33	>7201.5

Table 43. Detailed computational results for the **LIN** test set.

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
lin01	53	160	4	36	112	3	0.0	**503**			1	0.0
lin02	55	164	6	0	0	0	0.0	**557**			1	0.0
lin03	57	168	8	15	46	5	0.0	**926**			1	0.0
lin04	157	532	6	9	24	4	0.0	**1239**			1	0.0
lin05	160	538	9	108	410	8	0.0	**1703**			1	0.1
lin06	165	548	14	78	294	10	0.1	**1348**			1	0.1
lin07	307	1052	6	73	240	6	0.1	**1885**			1	0.2
lin08	311	1060	10	295	1014	9	0.1	**2248**			1	0.3
lin09	313	1064	12	298	1022	12	0.1	**2752**			1	0.9
lin10	321	1080	20	241	848	17	0.2	**4132**			1	1.3
lin11	816	2920	10	339	1176	10	1.6	**4280**			1	2.5
lin12	818	2924	12	734	2690	12	0.6	**5250**			1	5.6
lin13	822	2932	16	307	1050	16	1.6	**4609**			1	2.8
lin14	828	2944	22	413	1422	20	2.1	**5824**			1	3.4
lin15	840	2968	34	803	2864	33	0.3	**7145**			1	4.3
lin16	1981	7266	12	700	2500	11	6.1	**6618**			1	9.7
lin17	1989	7282	20	880	3166	20	7.7	**8405**			1	23.0
lin18	1994	7292	25	1097	3916	24	6.7	**9714**			1	38.7
lin19	2010	7324	41	1419	5090	41	7.3	**13268**			1	51.1
lin20	3675	13418	11	1344	4730	11	14.7	**6673**			1	24.2
lin21	3683	13434	20	1063	3692	18	18.4	**9143**			1	47.0
lin22	3692	13452	28	1468	5152	27	13.7	**10519**			1	80.8
lin23	3716	13500	52	2558	9110	51	19.1	**17560**			1	787.8
lin24	7998	29468	16	7985	29426	16	2.3	14885.9122	15076	1.3	1	>7200.1
lin25	8007	29486	24	7969	29382	24	2.9	17556.7813	17803	1.4	1	>7200.2
lin26	8013	29498	30	7999	29452	30	2.6	20851.8774	21768	4.4	1	>7200.1
lin27	8017	29506	36	7997	29442	36	2.3	20138.8247	20678	2.7	1	>7200.0
lin28	8062	29596	81	8032	29504	81	3.1	31535.6843	32627	3.5	1	>7200.1
lin29	19083	71272	24	19052	71204	24	7.3	20033.7382	23765	18.6	1	>7200.2
lin30	19091	71288	31	19069	71218	31	7.4	22598.1936	27690	22.5	1	>7200.1
lin31	19100	71306	40	19078	71226	40	7.6	25976.4075	31726	22.1	1	>7200.2
lin32	19112	71330	53	19091	71248	53	7.9	32240.0159	39995	24.1	1	>7200.2
lin33	19177	71460	117	19114	71256	117	12.9	47904.9813	56088	17.1	1	>7200.2
lin34	38282	143042	34	38265	142974	34	16.3	30461.1131	45051	47.9	1	>7200.1

cont. next page

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
lin35	38294	143066	45	38274	142988	45	16.9	33934.0327	50590	49.1	1	>7200.2
lin36	38307	143092	58	38284	143018	58	17.3	38248.9295	55827	46.0	1	>7200.3
lin37	38418	143314	172	38320	142990	170	16.4	73789.2798	99846	35.3	1	>7200.1

Table 44. Detailed computational results for the **vienna-I** test set.

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
I001	30190	95496	1184	12943	40240	979	36.9	253610976	253922544	0.1	1	>7205.0
I002	49920	155742	1665	21271	65932	1315	69.8	398609665	399814340	0.3	1	>7205.4
I003	44482	146838	3222	13588	40954	2391	21.0	788636184	788783445	0.0	1	>7208.6
I004	5556	17104	570	738	2028	286	1.4	**279512692**			1	7.2
I005	10284	31960	1017	1639	4530	609	2.1	**390876350**			1	42.4
I006	31754	105750	2202	11518	34756	1832	16.6	504512349	504528222	0.0	1	>7205.1
I007	15122	48742	737	6147	19016	614	13.5	**177909660**			1	2166.8
I008	15714	51134	871	5434	16594	711	12.9	**201788202**			3	1808.3
I009	33188	104014	1262	13367	41640	1071	46.0	275281214	275559586	0.1	1	>7206.7
I010	29905	94914	943	10721	34030	778	28.7	**207889674**			1	2302.9
I011	25195	82596	1428	7560	23138	1238	15.6	**317589880**			65	2506.6
I012	12355	39924	503	2419	7542	423	5.6	**118893243**			1	72.8
I013	18242	57952	891	5978	18462	675	11.0	**193190339**			1	1439.46
I014	12715	41264	475	2375	7610	359	4.6	**105173465**			1	41.9
I015	48833	159974	2493	16093	50092	2102	36.1	591844807	592254743	0.1	1	>7205.1
I016	72038	230110	4391	22722	68728	3453	49.5	1110221060	1110942710	0.1	1	>7205.2
I017	15095	48182	478	6542	20714	413	13.8	**109739695**			1	307.9
I018	31121	102226	1898	10765	32812	1584	19.8	463844907	463893137	0.0	1	>7206.2
I019	25946	83290	866	8551	27732	717	21.1	217628086	217647736	0.0	1	>7205.9
I020	21808	69842	594	4440	14264	509	14.2	**146515460**			1	331.1
I021	16013	50538	392	3386	11202	291	10.1	**106470644**			1	228.4
I022	16224	51382	437	7587	24222	364	19.5	**106799980**			3	4508.9
I023	22805	70614	582	12562	39168	438	45.0	**131044872**			1	3912.6
I024	68464	217464	3001	27346	85220	2485	68.9	756810911	758501064	0.2	1	>7205.1
I025	23412	75904	945	7234	22850	845	16.1	232687393	232792392	0.0	1	>7205.1
I026	47429	158614	3334	15019	45210	2796	19.6	927786863	928043068	0.0	1	>7205.1
I027	85085	277776	3954	32570	101462	3520	70.7	973458232	976869730	0.4	1	>7205.3
I028	72701	230860	1790	38476	121400	1631	117.3	381834536	384064899	0.6	1	>7208.8
I029	69988	223608	2162	26904	85158	1940	90.4	489246839	492219164	0.6	1	>7205.2
I030	33188	107360	1263	9200	29242	1074	17.9	**321646787**			3	4060.8
I031	54351	176422	2182	15625	49488	1804	40.5	577697592	578295319	0.1	1	>7211.2
I032	56023	182798	3017	17486	53486	2491	35.2	772960829	773101910	0.0	1	>7206.2
I033	18555	59460	636	7103	22118	560	19.3	**134461857**			3	700.9
I034	22311	71032	735	6772	21626	633	15.5	**165115148**			1	802.8
I035	30585	100908	1704	10630	32736	1438	19.2	414440229	414440370	0.0	362	>7218.1
I036	37208	120712	1411	12783	40694	1233	25.3	374880427	375274673	0.1	1	>7224.8
I037	13694	44252	427	4578	14890	394	10.2	**105720727**			3	520.8
I038	18747	61278	967	5888	18304	797	10.6	**255767543**			12	2566.0
I039	8755	28898	347	2922	9128	328	6.3	**85566290**			3	186.3
I040	40389	131640	1762	14563	45816	1497	28.7	431081352	431511925	0.1	1	>7205.2
I041	47197	150614	1193	17568	56282	1042	56.4	301397934	301916888	0.2	1	>7205.4
I042	51896	171100	2171	19487	61194	1961	39.3	531354770	532147163	0.1	1	>7205.7
I043	10398	33574	367	3338	10558	327	6.0	**95722094**			1	157.6
I044	68905	227778	3358	24891	77356	2940	55.1	803731728	804553575	0.1	1	>7205.2
I045	14685	46932	421	5256	16772	374	12.4	**105944062**			1	261.0

cont. next page

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
I046	70843	234418	3598	25108	78610	3074	49.6	920138185	925523357	0.6	1	>7205.2
I047	28524	92502	2354	8541	25524	1716	11.6	695117478	695167342	0.0	1	>7205.0
I048	13189	42438	358	3844	12410	324	10.4	**91509264**			1	157.2
I049	30857	99182	990	11260	36432	833	28.2	294808863	294812582	0.0	1	>7205.0
I050	43073	142552	2868	14537	44222	2271	33.4	792282177	792618583	0.0	1	>7205.1
I051	27028	90812	1524	9900	30362	1343	15.7	357196933	357233443	0.0	1	>7205.0
I052	2363	7522	40	93	268	28	0.4	**13309487**			1	0.4
I053	3224	10570	126	557	1732	101	0.7	**30854904**			1	2.9
I054	3803	12426	38	380	1252	29	1.5	**15841596**			1	3.3
I055	13332	43160	570	3386	10648	463	10.3	**144164924**			1	128.3
I056	1991	6352	51	194	610	41	0.6	**14171206**			1	0.9
I057	33231	110298	1569	10549	32838	1305	23.7	**412746415**			11	3781.7
I058	23527	79256	1256	5971	18500	1033	9.6	**305024188**			1	411.4
I059	9287	29950	363	1991	6258	296	6.5	**107617854**			3	100.8
I060	42008	135144	1242	14071	45572	1152	31.5	337082707	337300029	0.1	1	>7205.1
I061	39160	127318	1458	17680	55568	1325	42.5	362757775	363047214	0.1	1	>7205.1
I062	66048	220982	3343	17146	54428	2751	28.9	792452883	792967961	0.1	1	>7216.2
I063	26840	87322	1645	7346	22372	1263	12.5	**459801704**			24	2877.8
I064	63158	214690	3458	26956	82030	3177	47.0	860958942	863131705	0.3	1	>7205.6
I065	3898	12712	144	892	2792	120	1.8	**32965718**			1	15.2
I066	15038	49192	551	3234	10402	444	6.6	**174219813**			1	150.2
I067	20547	66460	627	8213	26072	561	19.7	**175540750**			1	3387.1
I068	33118	110254	1553	8491	26302	1241	16.7	**420730046**			10	4047.3
I069	9574	32416	543	2742	8356	455	3.9	**135161583**			1	223.2
I070	15079	49216	550	4940	15794	510	7.7	**136700139**			1	1501.7
I071	33203	108854	1494	9643	30016	1267	17.1	**382539099**			1	1432.7
I072	26948	88388	993	8283	26838	839	16.9	**289019226**			3	2710.7
I073	21653	70342	1847	6197	18312	1308	7.3	**663004987**			1	2333.4
I074	13316	44066	653	3148	9732	532	5.0	**165573383**			3	71.4
I075	57551	190762	2973	17410	54174	2429	30.0	815008510	815427527	0.1	1	>7205.1
I076	14023	45790	598	3324	10432	484	7.0	**166249692**			3	192.5
I077	20856	68474	1787	7878	23480	1478	8.3	**472503150**			7	6238.7
I078	13294	43896	835	5005	15198	701	6.9	**185525490**			11	964.6
I079	19867	62542	565	5773	18436	519	11.5	**150506933**			1	2714.0
I080	18695	59416	548	5363	16978	502	10.9	**164299652**			1	729.7
I081	25081	81478	888	8005	25358	742	17.4	**247527679**			1	6652.8
I082	15592	49576	515	3898	12334	451	11.2	**147407632**			1	576.8
I083	89596	297166	4991	25706	78868	3984	42.9	1400707250	1405679260	0.4	1	>7205.1
I084	44934	147454	2319	12460	38814	1861	19.9	626875799	627199178	0.1	1	>7205.1
I085	9113	28982	301	2074	6576	253	2.8	**80628079**			1	34.3

Table 45. Detailed computational results for the **PUC** test set.

Instance	Original $\|V\|$	$\|A\|$	$\|T\|$	Presolved $\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
bip42p	1200	7964	200	990	7236	200	0.2	24483.7332	24657	0.7	3704	>7205.0
bip42u	1200	7964	200	990	7220	200	0.1	233.157627	236	1.2	3371	>7205.1
bip52p	2200	15994	200	1819	14674	200	0.7	24251.398	24835	2.4	1059	>7205.0
bip52u	2200	15994	200	1819	14652	200	0.6	229.765274	235	2.3	828	>7205.0
bip62p	1200	20004	200	1199	20000	200	0.4	22511.5386	22973	2.0	404	>7205.0
bip62u	1200	20004	200	1199	20000	200	0.3	214.406798	220	2.6	557	>7205.0
bipa2p	3300	36146	300	3140	35594	300	0.9	34709.6655	35871	3.3	100	>7205.5
bipa2u	3300	36146	300	3140	35590	300	0.5	329.773497	342	3.7	90	>7210.4
bipe2p	550	10026	50	550	10026	50	0.1	**5616**			2306	1202.8
bipe2u	550	10026	50	550	10026	50	0.1	**54**			83	114.5
cc10-2p	1024	10240	135	1024	10240	135	0.1	34514.7527	35733	3.5	1	>7205.0

cont. next page

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
cc10-2u	1024	10240	135	1024	10240	135	0.1	334.498719	346	3.4	1	>7205.9
cc11-2p	2048	22526	244	2048	22526	244	0.3	62166.4479	64846	4.3	1	>7206.0
cc11-2u	2048	22526	244	2048	22526	244	0.3	602.818662	624	3.5	1	>7205.4
cc12-2p	4096	49148	473	4096	49148	473	0.8	118482.56	124348	5.0	1	>7205.7
cc12-2u	4096	49148	473	4096	49148	473	0.8	1149.03005	1195	4.0	1	>7205.5
cc3-10p	1000	27000	50	1000	27000	50	0.7	12198.4462	13067	7.1	1	>7205.5
cc3-10u	1000	27000	50	1000	27000	50	0.5	117.414634	127	8.2	1	>7205.2
cc3-11p	1331	39930	61	1331	39930	61	1.0	14773.8367	15917	7.7	1	>7205.4
cc3-11u	1331	39930	61	1331	39930	61	0.8	142.640411	157	10.1	1	>7205.3
cc3-12p	1728	57024	74	1728	57024	74	1.2	17769.6393	19422	9.3	1	>7205.4
cc3-12u	1728	57024	74	1728	57024	74	1.1	172.205224	190	10.3	1	>7205.3
cc3-4p	64	576	8	64	576	8	0.0	**2338**			7217	189.4
cc3-4u	64	576	8	64	576	8	0.0	**23**			219	23.8
cc3-5p	125	1500	13	125	1500	13	0.0	3547.62569	3661	3.2	28060	>7205.0
cc3-5u	125	1500	13	125	1500	13	0.0	34.5383179	36	4.2	30693	>7205.0
cc5-3p	243	2430	27	243	2430	27	0.0	7206.3399	7303	1.3	4707	>7205.0
cc5-3u	243	2430	27	243	2430	27	0.0	**71**			2885	4570.0
cc6-2p	64	384	12	64	384	12	0.0	**3271**			165	7.8
cc6-2u	64	384	12	64	384	12	0.0	**32**			47	3.6
cc6-3p	729	8736	76	729	8736	76	0.1	20139.9549	20525	1.9	304	>7205.0
cc6-3u	729	8736	76	729	8736	76	0.1	195.638889	202	3.3	1	>7205.1
cc7-3p	2187	30616	222	2187	30616	222	0.7	55296.1132	58286	5.4	1	>7205.6
cc7-3u	2187	30616	222	2187	30616	222	0.3	535.84192	562	4.9	1	>7205.6
cc9-2p	512	4608	64	512	4608	64	0.1	16892.5375	17343	2.7	320	>7205.0
cc9-2u	512	4608	64	512	4608	64	0.1	163.718525	170	3.8	172	>7205.0
hc10p	1024	10240	512	1024	10240	512	0.1	59231.0381	60983	3.0	54	>7205.0
hc10u	1024	10240	512	1024	10240	512	0.1	567.777778	592	4.3	1	>7205.0
hc11p	2048	22528	1024	2048	22528	1024	0.3	117383.13	121326	3.4	5	>7205.0
hc11u	2048	22528	1024	2048	22528	1024	0.2	1124.87009	1186	5.4	1	>7205.9
hc12p	4096	49152	2048	4096	49152	2048	0.7	232522.973	244676	5.2	1	>7206.5
hc12u	4096	49152	2048	4096	49152	2048	0.4	2217.66667	2358	6.3	1	>7205.6
hc6p	64	384	32	64	384	32	0.0	**4003**			1547	24.7
hc6u	64	384	32	64	384	32	0.0	**39**			541	12.2
hc7p	128	896	64	128	896	64	0.0	7845.68193	7905	0.8	92143	>7205.0
hc7u	128	896	64	128	896	64	0.0	75.0990594	77	2.5	102661	>7205.0
hc8p	256	2048	128	256	2048	128	0.0	15194.7194	15327	0.9	17155	>7205.0
hc8u	256	2048	128	256	2048	128	0.0	145.247228	148	1.9	11027	>7205.0
hc9p	512	4608	256	512	4608	256	0.0	29933.1635	30254	1.1	1369	>7205.0
hc9u	512	4608	256	512	4608	256	0.0	286.875	292	1.8	213	>7205.0

Table 46. Detailed computational results for the **X** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	t [s]	Optimum	N	t [s]
berlin52	52	2652	16	36	214	13	0.0	**1044**	1	0.0
brasil58	58	3306	25	20	134	7	0.0	**13655**	1	0.0
world666	666	442890	174	480	5588	93	1.0	**122467**	1	5.9

C.2 The Steiner Arborescence Problem

Table 47. Detailed computational results for the **Gene** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
gene41×	335	910	43	189	612	42	0.0	**126**	1	0.1
gene42	335	912	43	190	616	42	0.0	**126**	1	0.1
gene61a	395	1024	82	190	592	75	0.0	**205**	1	0.1
gene61b	570	1616	82	353	1176	78	0.0	**199**	1	0.3
gene61c	549	1580	82	357	1188	80	0.0	**196**	1	0.3
gene61f	412	1104	82	221	704	77	0.0	**198**	1	0.2
gene425	425	1108	86	206	644	78	0.0	**214**	1	0.2
gene442	442	1188	86	238	758	80	0.0	**207**	1	0.1
gene575	575	1648	86	368	1224	84	0.0	**207**	1	0.2
gene602	602	1716	86	380	1266	82	0.0	**209**	1	0.2

Table 48. Detailed computational results for the **Gene2002** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
microtri1	347	952	47	201	656	47	0.0	**128**	1	0.1
microtri3	400	1112	47	252	810	45	0.0	**146**	1	0.1
microtri5	416	1124	47	237	754	46	0.0	**150**	1	0.1
microtri6	419	1164	47	261	842	45	0.0	**146**	1	0.1
microtri7	437	1172	47	261	812	46	0.0	**159**	1	0.1
microtri8	484	1412	47	317	1066	46	0.0	**151**	1	0.1
microtri9	297	792	47	168	528	46	0.0	**131**	1	0.0
microtri10	319	836	47	170	532	44	0.0	**136**	1	0.0
microtri11	382	1024	47	215	676	46	0.0	**152**	1	0.2

C.3 The Rectilinear Steiner Minimum Tree Problem

Table 49. Detailed computational results for the **Solids** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
cube	8	24	8	0	0	0	0.0	**7**	1	0.0
dodecahedron	343	1764	20	343	1764	20	0.0	**7.69398**	6557	1232.6
icosahedron	125	600	12	125	600	12	0.0	**20.944264**	5	3.1
octahedron	27	108	6	0	0	0	0.0	**6**	1	0.0
tetrahedron	18	66	4	8	28	4	0.0	**2.682521**	1	0.0

Table 50. Detailed computational results for the **Estein50** test set.

Instance	Original			Presolved				Dual	Primal	Gap %	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]					
estein50-0	2500	9800	50	1967	7682	50	9.1	**5.494867**			1	1101.3
estein50-10	2500	9800	50	1903	7432	50	8.1	**5.253293**			1	1634.0
estein50-11	2500	9800	50	2462	9720	50	2.0	5.32590095	5.34093	0.3	107	>7200.0
estein50-12	2500	9800	50	1974	7714	49	11.5	**5.389099**			1	1335.1
estein50-13	2500	9800	50	2057	8042	49	7.4	**5.355143**			1	1881.8
estein50-14	2500	9800	50	2108	8246	50	8.6	**5.218085**			1	1124.6
estein50-1	2500	9800	50	1863	7268	50	7.1	**5.548422**			1	1573.6
estein50-2	2500	9800	50	1859	7252	50	9.1	**5.469105**			1	1627.8
estein50-3	2500	9800	50	1890	7382	50	8.6	**5.153576**			1	509.2
estein50-4	2500	9800	50	1911	7456	50	10.3	**5.518601**			1	664.2
estein50-5	2500	9800	50	2087	8164	50	8.0	**5.58043**			1	2017.0
estein50-6	2500	9800	50	1859	7260	50	8.8	**4.996117**			1	4617.0
estein50-7	2500	9800	50	2137	8358	50	8.0	**5.375465**			1	500.5
estein50-8	2500	9800	50	2458	9716	50	2.1	**5.345677**			67	2945.9
estein50-9	2500	9800	50	2448	9684	50	2.1	**5.403795**			1	1955.5

Table 51. Detailed computational results for the **Estein60** test set.

Instance	Original			Presolved				Dual	Primal	Gap %	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]					
estein60-0	3600	14160	60	2637	10326	60	16.0	**5.376143**			1	1796.2
estein60-10	3600	14160	60	3003	11798	60	14.8	**5.614167**			1	5453.9
estein60-11	3600	14160	60	3151	12378	60	13.5	5.96254703	5.979133	0.3	1	>7199.1
estein60-12	3600	14160	60	2791	10942	60	13.9	6.08189194	6.121356	0.6	1	>7199.1
estein60-13	3600	14160	60	2882	11316	60	15.1	**5.603556**			1	3568.7
estein60-14	3600	14160	60	3561	14074	60	2.6	**5.662257**			1	4161.8
estein60-1	3600	14160	60	2848	11168	59	15.6	5.53180429	5.536782	0.1	21	>7199.3
estein60-2	3600	14160	60	2814	11032	60	11.7	**5.656678**			1	4042.8
estein60-3	3600	14160	60	3058	12012	60	13.6	5.492529	5.537805	0.8	1	>7199.1
estein60-4	3600	14160	60	2769	10856	60	12.2	5.46620971	5.470499	0.1	308	>7199.0
estein60-5	3600	14160	60	2968	11654	60	12.0	**6.042196**			1	3821.4
estein60-6	3600	14160	60	3554	14060	60	2.6	5.89335023	5.897803	0.1	1	>7199.1
estein60-7	3600	14160	60	2943	11548	60	15.6	**5.813816**			1	3880.4
estein60-8	3600	14160	60	2976	11692	59	12.4	**5.587713**			1	4926.0
estein60-9	3600	14160	60	2767	10844	60	13.9	**5.762446**			1	2618.9

Table 52. Detailed computational results for the **Cancer** test set.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
c10_6D	10000	94000	82	576	3266	11	0.7	**92**			1	0.8
c11_8D	4762800	64777860	75	4762776	64777218	51	0.0	–	–	–	1	memout
c12_8D	918750	12031250	58	918717	12030418	33	294.9	–	116	–	1	>7200.1
c13_8D	86400	1039680	70	44806	474578	16	11.6	**88**			1	384.8
c14_8D	27648	308736	54	621	3642	10	4.0	**63**			1	4.1
c1_4D	600	3820	20	584	3644	6	0.0	**28**			1	0.1
c2_4D	256	1536	20	237	1318	3	0.0	**21**			1	0.0
c3_6D	20580	197078	110	20493	195108	36	1.9	**146**			1	99.7
c4_6D	34560	340416	93	34491	338830	26	5.6	132.90	136	2.3	1	>7200.1
c5_6D	8000	74400	48	7833	72108	18	1.0	**69**			1	444.9
c6_6D	5120	46592	50	120	672	4	0.5	**55**			1	0.5
c7_6D	21000	203300	109	20919	201382	30	1.9	**140**			1	158.0
c8_6D	8640	80064	77	5520	46190	14	0.9	**89**			1	7.1
c9_6D	6000	54800	46	2064	14486	11	0.6	**59**			1	2.3

C.4 The Prize Collecting Tree Problem

Table 53. Detailed computational results for the **JMP** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	t [s]	Optimum	N	t [s]
K100-10	115	722	15	71	432	11	0.0	**133567**	1	0.0
K100-1	112	762	12	78	550	10	0.0	**124108**	1	0.1
K100-2	114	756	14	57	390	9	0.0	**200262**	1	0.1
K100-3	111	874	11	62	494	8	0.0	**115953**	1	0.1
K100-4	111	788	11	86	562	10	0.0	**87498**	1	0.1
K100-5	117	812	17	79	554	10	0.0	**119078**	1	0.1
K100-6	112	680	12	67	432	10	0.0	**132886**	1	0.1
K100-7	114	708	14	69	440	10	0.0	**172457**	1	0.1
K100-8	116	776	16	78	512	12	0.0	**210869**	1	0.1
K100-9	112	732	12	77	518	11	0.0	**122917**	1	0.1
K100-0	115	786	15	70	478	13	0.0	**135511**	1	0.1
K200-0	234	1580	34	156	1016	22	0.0	**329211**	1	0.1
K400-10	450	3308	50	327	2220	41	0.1	**394191**	1	2.1
K400-1	465	3324	65	310	2018	45	0.1	**490771**	1	1.4
K400-2	462	3420	62	287	1868	47	0.1	**477073**	1	1.8
K400-3	456	3314	56	343	2380	47	0.1	**415328**	1	1.4
K400-4	456	3182	56	312	1998	46	0.1	**389451**	1	1.6
K400-5	477	3368	77	319	1948	59	0.1	**519526**	1	1.8
K400-6	456	3482	56	329	2348	47	0.1	**374849**	1	1.3
K400-7	468	3286	68	315	1936	54	0.1	**474466**	1	1.8
K400-8	461	3392	61	335	2364	52	0.1	**418614**	1	1.2
K400-9	454	3318	54	323	2210	42	0.1	**383105**	1	0.9
K400-0	463	3402	63	334	2248	50	0.1	**350093**	1	0.5
P100-1	133	760	33	69	294	23	0.0	**926238**	1	0.0
P100-2	127	750	27	70	306	19	0.0	**401641**	1	0.1
P100-3	125	776	25	3	6	2	0.0	**659644**	1	0.0
P100-4	133	760	33	21	86	10	0.0	**827419**	1	0.0
P100-0	134	832	34	24	100	10	0.0	**803300**	1	0.0
P200-0	249	1462	49	111	504	27	0.1	**1317874**	1	0.1
P400-1	521	3144	121	243	1184	65	0.2	**2808440**	1	0.5
P400-2	508	3034	108	292	1422	67	0.2	**2518577**	1	0.4
P400-3	514	3028	114	346	1720	84	0.2	**2951725**	1	0.5
P400-4	495	2852	95	308	1482	65	0.2	**2852956**	1	0.6
P400-0	495	2964	95	272	1298	61	0.1	**2459904**	1	0.3

Table 54. Detailed computational results for the **CRR** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	t [s]	Optimum	N	t [s]
C01-A	506	1280	6	3	6	2	0.0	**18**	1	0.1
C01-B	506	1280	6	13	46	5	0.0	**85**	1	0.1
C02-A	511	1310	11	3	6	2	0.0	**50**	1	0.1
C02-B	511	1310	11	79	304	9	0.0	**141**	1	0.1
C03-A	584	1748	84	179	728	41	0.0	**414**	1	0.2
C03-B	584	1748	84	114	490	34	0.0	**737**	1	0.2
C04-A	626	2000	126	172	730	50	0.0	**618**	1	0.2
C04-B	626	2000	126	136	578	39	0.0	**1063**	1	0.1
C05-A	751	2750	251	94	456	42	0.0	**1080**	1	0.1
C05-B	751	2750	251	16	64	7	0.0	**1528**	1	0.0
C06-A	506	2030	6	3	6	2	0.1	**18**	1	0.1
C06-B	506	2030	6	66	282	6	0.1	**55**	1	0.1

cont. next page

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
C07-A	511	2060	11	3	6	2	0.1	**50**	1	0.1
C07-B	511	2060	11	286	1330	10	0.1	**102**	1	0.2
C08-A	584	2498	84	374	1770	56	0.2	**361**	1	0.5
C08-B	584	2498	84	306	1404	50	0.2	**500**	1	0.4
C09-A	626	2750	126	414	1986	84	0.2	**533**	1	0.6
C09-B	626	2750	126	358	1662	75	0.2	**694**	1	0.7
C10-A	751	3500	251	287	1482	99	0.1	**859**	1	0.6
C10-B	751	3500	251	3	6	2	0.1	**1069**	1	0.1
C11-A	506	5030	6	3	6	2	0.1	**18**	1	0.1
C11-B	506	5030	6	161	786	6	0.1	**32**	1	0.2
C12-A	511	5060	11	201	970	10	0.2	**38**	1	0.2
C12-B	511	5060	11	399	2442	10	0.2	**46**	1	0.3
C13-A	584	5498	84	459	2652	59	0.3	**236**	1	0.6
C13-B	584	5498	84	424	2338	54	0.3	**258**	1	0.7
C14-A	626	5750	126	388	2058	72	0.3	**293**	1	0.9
C14-B	626	5750	126	326	1672	61	0.3	**318**	1	0.7
C15-A	751	6500	251	243	1274	76	0.2	**501**	1	0.6
C15-B	751	6500	251	3	6	2	0.2	**551**	1	0.2
C16-A	506	25030	6	71	318	6	0.3	**11**	1	0.4
C16-B	506	25030	6	71	318	6	0.4	**11**	1	0.4
C17-A	511	25060	11	362	2342	9	0.5	**18**	1	0.6
C17-B	511	25060	11	185	966	9	0.5	**18**	1	0.5
C18-A	584	25498	84	482	3388	48	0.6	**111**	1	1.0
C18-B	584	25498	84	482	3384	48	0.5	**113**	1	1.1
C19-A	626	25750	126	322	1694	39	0.5	**146**	1	1.1
C19-B	626	25750	126	326	1710	41	0.5	**146**	1	1.1
C20-A	751	26500	251	3	6	2	0.3	**266**	1	0.3
C20-B	751	26500	251	3	6	2	0.4	**267**	1	0.4
D01-A	1006	2530	6	3	6	2	0.1	**18**	1	0.1
D01-B	1006	2530	6	139	552	6	0.1	**106**	1	0.1
D02-A	1011	2560	11	3	6	2	0.1	**50**	1	0.1
D02-B	1011	2560	11	228	872	11	0.1	**218**	1	0.2
D03-A	1168	3502	168	295	1228	65	0.1	**807**	1	0.3
D03-B	1168	3502	168	184	800	53	0.1	**1509**	1	0.3
D04-A	1251	4000	251	341	1480	99	0.1	**1203**	1	0.4
D04-B	1251	4000	251	97	414	34	0.1	**1881**	1	0.1
D05-A	1501	5500	501	295	1400	116	0.0	**2157**	1	0.8
D05-B	1501	5500	501	55	240	24	0.1	**3135**	1	0.1
D06-A	1006	4030	6	3	6	2	0.1	**18**	1	0.1
D06-B	1006	4030	6	384	1968	6	0.3	**67**	1	0.4
D07-A	1011	4060	11	3	6	2	0.1	**50**	1	0.1
D07-B	1011	4060	11	484	2298	11	0.3	**103**	1	0.4
D08-A	1168	5002	168	844	3992	126	0.5	**755**	1	2.2
D08-B	1168	5002	168	745	3482	117	0.5	**1036**	1	1.7
D09-A	1251	5500	251	842	4104	178	0.4	**1070**	1	4.0
D09-B	1251	5500	251	629	3078	139	0.5	**1420**	1	3.7
D10-A	1501	7000	501	697	3742	234	0.3	**1671**	1	5.4
D10-B	1501	7000	501	207	1068	76	0.2	**2079**	1	0.3
D11-A	1006	10030	6	3	6	2	0.2	**18**	1	0.2
D11-B	1006	10030	6	184	912	6	0.3	**29**	1	0.4
D12-A	1011	10060	11	371	1886	10	0.4	**42**	1	0.5
D12-B	1011	10060	11	360	1796	10	0.4	**42**	1	0.6
D13-A	1168	11002	168	975	5788	122	0.7	**445**	1	2.7
D13-B	1168	11002	168	896	5084	108	0.7	**486**	1	2.4

cont. next page

Instance	Original			Presolved						
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Optimum	N	t [s]
D14-A	1251	11500	251	821	4618	138	1.0	**602**	1	3.7
D14-B	1251	11500	251	730	3942	119	0.8	**665**	1	2.2
D15-A	1501	13000	501	487	2666	146	0.8	**1042**	1	1.9
D15-B	1501	13000	501	8	28	4	0.8	**1108**	1	0.8
D16-A	1006	50030	6	132	690	6	1.1	**13**	1	1.1
D16-B	1006	50030	6	132	690	6	1.1	**13**	1	1.1
D17-A	1011	50060	11	867	8350	10	1.3	**23**	1	1.8
D17-B	1011	50060	11	647	4648	10	1.3	**23**	1	1.6
D18-A	1168	51002	168	1002	7976	95	1.5	**218**	1	4.1
D18-B	1168	51002	168	1002	7946	95	1.5	**223**	1	5.0
D19-A	1251	51500	251	857	5372	102	1.5	**306**	1	6.3
D19-B	1251	51500	251	845	5258	98	1.4	**310**	1	6.9
D20-A	1501	53000	501	3	6	2	2.1	**536**	1	2.1
D20-B	1501	53000	501	3	6	2	2.1	**537**	1	2.1

Table 55. Detailed computational results for the **PUCNU** test set.

Instance	Original			Presolved				Dual	Primal	Gap %	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]					
bip42nu	1401	9164	201	1191	8420	201	0.3	224.641338	226	0.6	2745	>7203.0
bip52nu	2401	17194	201	2020	15852	201	0.9	220.394682	223	1.2	1145	>7203.0
bip62nu	1401	21204	201	1400	21200	201	0.9	210.260966	215	2.3	7	>7203.4
bipa2nu	3601	37946	301	3441	37390	301	1.9	320.440715	329	2.7	20	>7203.3
bipe2nu	601	10326	51	601	10326	51	0.3	**53**			9	393.6
cc10-2nu	1160	11050	136	1066	9872	89	0.5	165.736515	168	1.4	129	>7203.0
cc11-2nu	2293	23990	245	2123	21678	160	1.0	299.885636	307	2.4	1	>7203.3
cc12-2nu	4570	51986	474	4220	46846	299	1.9	557.50564	569	2.1	1	>7203.0
cc3-10nu	1051	27300	51	1051	16500	51	0.8	**61**			35	424.3
cc3-11nu	1393	40296	62	1393	23826	62	1.1	**79**			6	237.1
cc3-12nu	1803	57468	75	1803	33048	75	1.0	**95**			1	114.3
cc3-4nu	73	624	9	73	480	9	0.0	**10**			1	0.1
cc3-5nu	139	1578	14	139	1110	14	0.1	**17**			1	0.5
cc5-3nu	271	2592	28	267	2282	26	0.1	**36**			1	2.9
cc6-2nu	77	456	13	77	456	13	0.0	**15**			1	0.1
cc6-3nu	806	9192	77	736	7268	42	0.3	**95**			1	29.1
cc7-3nu	2410	31948	223	2250	26534	143	1.1	267.717976	273	2.0	1	>7203.2
cc9-2nu	577	4992	65	567	4874	60	0.2	**83**			1	49.9

Table 56. Detailed computational results for the **Cologne1** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
i101M1	758	12704	11	3	4	2	0.2	**109271.503**	1	0.2
i101M2	758	12704	11	469	7650	11	0.4	**315925.31**	1	3.4
i101M3	758	12704	11	468	7644	11	0.4	**355625.409**	1	5.0
i102M1	760	12730	12	3	4	2	0.2	**104065.801**	1	0.2
i102M2	760	12730	12	427	7036	9	0.5	**352538.819**	1	6.4
i102M3	760	12730	12	470	7358	12	0.5	**454365.927**	1	7.9
i103M1	764	12738	14	3	4	2	0.2	**139749.407**	1	0.2
i103M2	764	12738	14	459	7680	11	0.3	**407834.228**	1	3.6
i103M3	764	12738	14	463	7578	14	0.5	**456125.488**	1	5.8
i104M2	744	12598	4	59	616	3	0.3	**89920.8353**	1	0.3
i104M3	744	12598	4	378	6912	4	0.4	**97148.789**	1	1.0
i105M1	744	12604	4	3	4	2	0.2	**26717.2025**	1	0.2

cont. next page

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
i105M2	744	12604	4	349	6520	4	0.3	**100269.619**	1	2.1
i105M3	744	12604	4	373	6890	4	0.3	**110351.163**	1	4.2

Table 57. Detailed computational results for the **Cologne2** test set.

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
i201M2	1812	33522	10	330	5634	2	0.4	**355467.684**	1	0.6
i201M3	1812	33522	10	1117	22088	7	0.8	**628833.614**	1	23.2
i201M4	1812	33522	10	1588	28070	10	0.8	**773398.303**	1	47.6
i202M2	1814	33520	11	1010	19758	6	0.9	**288946.832**	1	6.2
i202M3	1814	33520	11	1231	22462	10	1.1	**419184.159**	1	31.4
i202M4	1814	33520	11	1258	22852	10	1.1	**430034.264**	1	23.8
i203M2	1824	33584	16	1020	20182	10	1.0	**459894.776**	1	3.5
i203M3	1824	33584	16	1240	22562	13	1.1	**643062.02**	1	51.9
i203M4	1824	33584	16	1450	25076	14	0.9	**677733.067**	1	54.2
i204M2	1805	33454	5	3	4	2	0.3	**161700.545**	1	0.3
i204M3	1805	33454	5	1093	21536	5	0.7	**245287.203**	1	24.6
i204M4	1805	33454	5	1093	21536	5	0.7	**245287.203**	1	28.6
i205M2	1823	33640	14	1132	21786	10	1.5	**571031.415**	1	19.0
i205M3	1823	33640	14	1198	22182	13	1.4	**672403.143**	1	36.8
i205M4	1823	33640	14	1343	24040	14	1.1	**713973.623**	1	26.4

C.5 The Maximum Weight Connected Subgraph Problem

Table 58. Detailed computational results for the **ACTMOD** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Optimum	N	t [s]
drosophila001	5298	187214	72	2452	43998	54	1.3	**24.3855064**	1	16.2
drosophila005	5421	187952	195	2190	24790	106	1.5	**178.663952**	1	17.6
drosophila0075	5477	188288	251	1998	20288	118	1.6	**260.523557**	1	8.5
HCMV	3919	58916	56	1445	21840	45	0.9	**7.55431486**	1	5.4
lymphoma	2102	15914	68	1092	9204	46	0.6	**70.1663087**	1	1.9
metabol_expr_mice_1	3674	9590	151	763	2670	77	0.4	**544.94837**	1	2.0
metabol_expr_mice_2	3600	9174	86	736	2520	47	0.4	**241.077524**	1	0.9
metabol_expr_mice_3	2968	7354	115	488	1748	53	0.2	**508.260877**	1	0.8

Table 59. Detailed computational results for the **JMPALMK** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Original $\|T\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved $\|T\|$	Presolved t [s]	Optimum	N	t [s]
1000-a-0-6-d-0-25-e-0-25	1443	12524	443	44	156	12	0.1	**931.538552**	1	0.2
1000-a-0-6-d-0-25-e-0-5	1638	13694	638	7	20	3	0.2	**1872.2754**	1	0.2
1000-a-0-6-d-0-25-e-0-75	1814	14750	814	3	6	2	0.1	**2789.57911**	1	0.1
1000-a-0-6-d-0-5-e-0-25	1621	13592	621	3	6	2	0.4	**522.525615**	1	0.4
1000-a-0-6-d-0-5-e-0-5	1757	14408	757	3	6	2	0.3	**1197.85102**	1	0.3
1000-a-0-6-d-0-5-e-0-75	1881	15152	881	3	6	2	0.6	**1762.70747**	1	0.6
1000-a-0-6-d-0-75-e-0-25	1815	14756	815	3	6	2	0.2	**332.791924**	1	0.2
1000-a-0-6-d-0-75-e-0-5	1894	15230	894	3	6	2	0.3	**754.300601**	1	0.3
1000-a-0-6-d-0-75-e-0-75	1949	15560	949	3	6	2	0.2	**998.215414**	1	0.2
1000-a-1-d-0-25-e-0-25	1443	29210	443	3	6	2	0.1	**939.39337**	1	0.1
1000-a-1-d-0-25-e-0-5	1638	30380	638	3	6	2	0.1	**1883.21361**	1	0.1
1000-a-1-d-0-25-e-0-75	1814	31436	814	3	6	2	0.1	**2789.57911**	1	0.1
1000-a-1-d-0-5-e-0-25	1621	30278	621	3	6	2	0.1	**533.4294**	1	0.1
1000-a-1-d-0-5-e-0-5	1757	31094	757	3	6	2	0.1	**1205.42131**	1	0.1
1000-a-1-d-0-5-e-0-75	1881	31838	881	3	6	2	0.1	**1770.27776**	1	0.1
1000-a-1-d-0-75-e-0-25	1815	31442	815	3	6	2	0.1	**336.829944**	1	0.1
1000-a-1-d-0-75-e-0-5	1894	31916	894	3	6	2	0.1	**760.284581**	1	0.1
1000-a-1-d-0-75-e-0-75	1949	32246	949	3	6	2	0.1	**1004.19939**	1	0.1
1500–a-0-6-d-0-25-e-0-25	2164	19302	664	93	326	24	0.2	**1333.47643**	1	0.3
1500–a-0-6-d-0-25-e-0-5	2457	21060	957	3	6	2	0.4	**2799.67722**	1	0.4
1500–a-0-6-d-0-25-e-0-75	2732	22710	1232	3	6	2	0.9	**4230.25112**	1	0.9
1500–a-0-6-d-0-5-e-0-25	2432	20910	932	3	6	2	3.4	**847.452011**	1	3.4
1500–a-0-6-d-0-5-e-0-5	2633	22116	1133	3	6	2	2.5	**1858.0926**	1	2.5
1500–a-0-6-d-0-5-e-0-75	2812	23190	1312	3	6	2	1.0	**2697.45876**	1	1.0
1500–a-0-6-d-0-75-e-0-25	2739	22752	1239	3	6	2	1.5	**502.17599**	1	1.5
1500–a-0-6-d-0-75-e-0-5	2850	23418	1350	3	6	2	0.7	**1089.77117**	1	0.7
1500–a-0-6-d-0-75-e-0-75	2924	23862	1424	3	6	2	0.2	**1423.61063**	1	0.2
1500–a-1-d-0-25-e-0-25	2164	45032	664	3	6	2	0.3	**1377.0144**	1	0.3
1500–a-1-d-0-25-e-0-5	2457	46790	957	3	6	2	0.2	**2820.05174**	1	0.2
1500–a-1-d-0-25-e-0-75	2732	48440	1232	3	6	2	0.3	**4230.25112**	1	0.3
1500–a-1-d-0-5-e-0-25	2432	46640	932	3	6	2	0.3	**860.618961**	1	0.3
1500–a-1-d-0-5-e-0-5	2633	47846	1133	3	6	2	0.4	**1865.66289**	1	0.4
1500–a-1-d-0-5-e-0-75	2812	48920	1312	3	6	2	0.2	**2707.70001**	1	0.2
1500–a-1-d-0-75-e-0-25	2739	48482	1239	3	6	2	0.3	**502.17599**	1	0.3
1500–a-1-d-0-75-e-0-5	2850	49148	1350	3	6	2	0.2	**1089.77117**	1	0.2

cont. next page

Instance	Original			Presolved				Optimum	N	t [s]
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]			
1500–a-1-d-0-75-e-0-75	2924	49592	1424	3	6	2	0.2	**1423.61063**	1	0.2
500-a-0-62-d-0-25-e-0-25	712	6460	212	27	94	7	0.0	**460.577357**	1	0.0
500-a-0-62-d-0-25-e-0-5	818	7096	318	3	6	2	0.0	**992.967111**	1	0.1
500-a-0-62-d-0-25-e-0-75	910	7648	410	3	6	2	0.0	**1447.54452**	1	0.1
500-a-0-62-d-0-5-e-0-25	805	7018	305	3	6	2	0.0	**280.832378**	1	0.1
500-a-0-62-d-0-5-e-0-5	878	7456	378	3	6	2	0.0	**655.623217**	1	0.1
500-a-0-62-d-0-5-e-0-75	945	7858	445	3	6	2	0.0	**965.554694**	1	0.1
500-a-0-62-d-0-75-e-0-25	910	7648	410	7	20	3	0.1	**171.628785**	1	0.1
500-a-0-62-d-0-75-e-0-5	945	7858	445	3	6	2	0.0	**362.188212**	1	0.1
500-a-0-62-d-0-75-e-0-75	972	8020	472	3	6	2	0.0	**490.623986**	1	0.1
500-a-1-d-0-25-e-0-25	712	14304	212	3	6	2	0.0	**471.393285**	1	0.0
500-a-1-d-0-25-e-0-5	818	14940	318	3	6	2	0.0	**995.313181**	1	0.0
500-a-1-d-0-25-e-0-75	910	15492	410	3	6	2	0.0	**1447.54452**	1	0.0
500-a-1-d-0-5-e-0-25	805	14862	305	3	6	2	0.0	**286.920868**	1	0.0
500-a-1-d-0-5-e-0-5	878	15300	378	3	6	2	0.0	**661.711707**	1	0.0
500-a-1-d-0-5-e-0-75	945	15702	445	3	6	2	0.0	**965.554694**	1	0.0
500-a-1-d-0-75-e-0-25	910	15492	410	3	6	2	0.0	**171.628785**	1	0.0
500-a-1-d-0-75-e-0-5	945	15702	445	3	6	2	0.0	**362.188212**	1	0.0
500-a-1-d-0-75-e-0-75	972	15864	472	3	6	2	0.0	**490.623986**	1	0.0
750-a-0-647-d-0-25-e-0-25	1079	10406	329	3	6	2	0.1	**702.644057**	1	0.1
750-a-0-647-d-0-25-e-0-5	1229	11306	479	3	6	2	0.1	**1419.77986**	1	0.1
750-a-0-647-d-0-25-e-0-75	1364	12116	614	3	6	2	0.1	**2116.58233**	1	0.1
750-a-0-647-d-0-5-e-0-25	1206	11168	456	3	6	2	0.5	**403.177763**	1	0.5
750-a-0-647-d-0-5-e-0-5	1315	11822	565	3	6	2	0.1	**946.129495**	1	0.1
750-a-0-647-d-0-5-e-0-75	1412	12404	662	3	6	2	0.1	**1382.77203**	1	0.1
750-a-0-647-d-0-75-e-0-25	1366	12128	616	3	6	2	0.2	**266.983922**	1	0.2
750-a-0-647-d-0-75-e-0-5	1423	12470	673	3	6	2	0.1	**580.407832**	1	0.1
750-a-0-647-d-0-75-e-0-75	1462	12704	712	3	6	2	0.0	**764.156726**	1	0.0
750-a-1-d-0-25-e-0-25	1079	21612	329	3	6	2	0.1	**708.143835**	1	0.1
750-a-1-d-0-25-e-0-5	1229	22512	479	3	6	2	0.1	**1426.44904**	1	0.1
750-a-1-d-0-25-e-0-75	1364	23322	614	3	6	2	0.0	**2116.58233**	1	0.0
750-a-1-d-0-5-e-0-25	1206	22374	456	3	6	2	0.1	**403.177763**	1	0.1
750-a-1-d-0-5-e-0-5	1315	23028	565	3	6	2	0.1	**946.129495**	1	0.1
750-a-1-d-0-5-e-0-75	1412	23610	662	3	6	2	0.1	**1382.77203**	1	0.1
750-a-1-d-0-75-e-0-25	1366	23334	616	3	6	2	0.1	**266.983922**	1	0.1
750-a-1-d-0-75-e-0-5	1423	23676	673	3	6	2	0.0	**580.407832**	1	0.0
750-a-1-d-0-75-e-0-75	1462	23910	712	3	6	2	0.1	**764.156726**	1	0.1

C.6 The Degree Constrained Steiner Tree Problem

Table 60. Detailed computational results for the **TreeFam** test set.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
TF101057-t1	52	2652	35	52	2652	35	0.0	**infeasible**			1	0.0
TF101057-t3	52	2652	35	52	2652	35	0.0	**2756**			1226	46.7
TF101125-t1	304	92112	155	304	92112	155	0.1	**infeasible**			1	4.6
TF101125-t3	304	92112	155	304	92112	155	0.2	53636.694	55601	3.7	528	>7210.0
TF101202-t1	188	35156	72	188	35156	72	0.1	79350.8282	81122	2.2	1743	>7210.0
TF101202-t3	188	35156	72	188	35156	72	0.0	77640.4004	78144	0.6	2087	>7210.3
TF102003-t1	832	691392	407	832	691392	407	1.8	190488.292	384836	102.0	3	>7213.7
TF102003-t3	832	691392	407	832	691392	407	1.8	175828.601	189124	7.6	3	>7235.0
TF105035-t1	237	55932	104	237	55932	104	0.1	34519.3276	45833	32.8	612	>7210.0
TF105035-t3	237	55932	104	237	55932	104	0.1	32512.735	33046	1.6	1679	>7210.1
TF105272-t1	476	226100	223	476	226100	223	0.5	132587.705	282527	113.1	24	>7215.3
TF105272-t3	476	226100	223	476	226100	223	0.5	122923.989	133024	8.2	33	>7210.0
TF105419-t1	55	2970	24	55	2970	24	0.0	**18668**			10242	161.4
TF105419-t3	55	2970	24	55	2970	24	0.0	**18223**			41	5.9
TF105897-t1	314	98282	133	314	98282	133	0.2	106229.702	172495	62.4	194	>7210.0
TF105897-t3	314	98282	133	314	98282	133	0.1	96089.1786	98536	2.5	202	>7211.1
TF106403-t1	119	14042	46	119	14042	46	0.0	**54124**			1423	493.3
TF106403-t3	119	14042	46	119	14042	46	0.0	**53760**			31	64.4
TF106478-t1	130	16770	54	130	16770	54	0.0	54957.8681	55197	0.4	16271	>7210.1
TF106478-t3	130	16770	54	130	16770	54	0.0	54722.7843	54839	0.2	39489	>7210.0

C.7 The Group Steiner Tree Problem

Table 61. Detailed computational results for the **GSTP1** test set.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
andre30f2	474	1828	30	455	1780	30	0.1	**569**			1	3.0
andre31f2	349	1284	31	320	1210	30	0.1	**635**			1	2.9
andre33f2	452	1746	33	437	1704	33	0.1	**513**			1	2.3
andre34f2	1253	5000	34	1234	4952	34	0.5	638.967318	646	1.1	142	>7210.0
andre36f2	442	1672	36	428	1632	36	0.2	**610**			1	4.9
andre37f2	1054	4216	37	1042	4186	37	0.3	**485**			1	262.9
andre38f2	618	2504	38	608	2480	38	0.2	**656**			614	1672.0
andre39f2	707	3310	39	700	3294	39	0.1	433.793547	452	4.2	223	>7210.0

Table 62. Detailed computational results for the **GSTP2** test set.

Instance	Original			Presolved								
	$\|V\|$	$\|A\|$	$\|T\|$	$\|V\|$	$\|A\|$	$\|T\|$	t [s]	Dual	Primal	Gap %	N	t [s]
andre50f2	1142	4622	50	1120	4572	50	0.2	666.012134	674	1.2	192	>7210.0
andre55f2	1751	6804	55	1691	6680	55	0.4	**888**			7	5604.2
andre60f2	838	3528	60	835	3522	60	0.1	1154.41773	1164	0.8	1565	>7210.0
andre64f2	1860	7380	64	1789	7212	60	1.0	**931**			1	3446.7
andre66f2	2623	10100	66	2478	9804	62	2.0	**920**			1	3455.5
andre73f2	1911	7308	73	1797	7044	65	1.0	**1207**			1	2846.9
andre76f2	1818	6990	76	1657	6540	67	1.9	**1026**			1	1752.9
andre78f2	2355	9384	78	2275	9204	74	1.1	1065.17799	1096	2.9	15	>7210.1
andre83f2	3177	12530	83	3050	12270	80	1.9	890.996184	906	1.7	1	>7210.2
andre84f2	2358	9134	84	2169	8648	74	1.6	1080.97377	1094	1.2	3	>7210.3

C.8 The Hop Constrained Directed Steiner Tree Problem

The hop-constrained directed Steiner tree instances described hereinafter encompass ten terminals each, both before and after preprocessing.

Table 63. Detailed computational results for the **Gr12** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved t [s]	Optimum	N	t [s]
wo11-cr100-se10	809	7432	335	3390	0.1	**136516**	1	2.8
wo11-cr100-se11	809	7430	507	5212	0.1	**145251**	1	3.5
wo11-cr100-se1	809	7444	609	6550	0.1	**182082**	1	3.8
wo11-cr100-se2	809	7394	481	5092	0.1	**163872**	1	2.7
wo11-cr200-se10	809	15262	483	9946	0.1	**59523**	1	5.1
wo11-cr200-se11	809	15260	661	14214	0.1	**66786**	1	9.6
wo11-cr200-se1	809	15274	663	14594	0.1	**76353**	1	10.8
wo11-cr200-se2	809	15224	651	13856	0.1	**75434**	1	9.2
wo12-cr100-se10	809	9360	510	6256	0.1	**167223**	1	3.2
wo12-cr100-se11	809	9852	569	7056	0.1	**199679**	1	4.6
wo12-cr100-se1	809	9446	544	6724	0.1	**164198**	1	2.6
wo12-cr100-se7	809	9702	293	3574	0.1	**136232**	1	1.8
wo12-cr200-se9	809	28346	281	7772	0.1	**46408**	1	3.9
wo10-cr100-se0	809	14396	809	14396	0.1	**171486**	1	38.2
wo10-cr100-se10	809	14428	616	10690	0.1	**117081**	1	8.1
wo10-cr100-se11	809	14386	663	11312	0.1	**125785**	1	12.7
wo10-cr200-se7	809	44696	435	19174	0.2	**46306**	1	11.0
wo10-cr200-se8	809	44654	805	44392	0.3	**61177**	495	434.1
wo10-cr200-se9	809	44670	721	37392	0.3	**51737**	643	376.0

Table 64. Detailed computational results for the **Gr14** test set.

Instance	Original $\|V\|$	Original $\|A\|$	Presolved $\|V\|$	Presolved $\|A\|$	Presolved t [s]	Dual	Primal	Gap %	N	t [s]
wo10-cr100-se0	3209	215940	3209	215940	1.0	149509.585	177721	18.9	59	>7210.0
wo10-cr100-se11	3209	215932	2786	182390	1.3	113542.879	124962	10.1	103	>7210.0
wo10-cr200-se3	3209	643552	3187	635486	4.3	45450.3838	53467	17.6	15	>7210.7
wo10-cr200-se4	3209	643414	3157	622726	3.8	40143.3633	56435	40.6	30	>7210.3
wo11-cr100-se6	3209	115502	2773	115502	0.7	200986.868	218436	8.7	40	>7210.1
wo11-cr200-se2	3209	232858	2725	225484	1.7	**71134**			615	3340.8
wo11-cr200-se3	3209	233104	1795	129456	1.4	**57930**			219	553.1
wo11-cr200-se4	3209	233038	2773	232526	1.4	**63313**			967	3158.2
wo12-cr100-se0	3209	153366	933	39686	0.7	**116288**			1	52.7
wo12-cr100-se5	3209	156578	1436	67248	0.8	**131631**			1	186.3
wo12-cr100-se6	3209	157214	2030	99256	0.9	**146049**			3078	6981.4
wo12-cr100-se7	3209	158984	1336	63418	0.7	**122306**			9	195.0
wo12-cr100-se8	3209	157912	1511	73304	0.8	**116077**			5	242.9
wo12-cr100-se9	3209	156658	712	28752	0.6	**99170**			1	23.5
wo12-cr200-se0	3209	445774	1512	180812	2.0	**53883**			9	615.6
wo12-cr200-se10	3209	446040	2317	311642	2.2	54658.2011	69264	26.7	97	>7210.1
wo12-cr200-se11	3209	457496	2383	330774	2.3	59396.3375	81839	37.8	124	>7210.2
wo12-cr200-se4	3209	460250	2395	329386	2.4	64347.5161	76291	18.6	119	>7210.3
wo12-cr200-se5	3209	456998	2159	277314	2.6	56114.3522	58096	3.5	831	>7210.5
wo12-cr200-se6	3209	460500	2377	327696	3.2	55507.5041	63843	15.0	78	>7210.0
wo12-cr200-se7	3209	464090	1777	216648	2.7	59315.5318	62393	5.2	647	>7210.3

Table 65. Detailed computational results for the **Gr16**, test set.

Instance	Original $\|V\|$	Original $\|A\|$	Presolved $\|V\|$	Presolved $\|A\|$	t [s]	Dual	Primal	Gap %	N	t [s]
wo10-cr100-se0	12509	2843882	11604	2843882	7.4	66973.2336	178781	166.9	1	>7202.5
wo10-cr100-se10	12509	2844058	8712	2052626	6.6	68262.564	117703	72.4	1	>7207.1
wo10-cr100-se6	12509	2843894	11604	2843894	8.3	67271.103	195958	191.3	1	>7209.2
wo10-cr200-se0	12509	8741560	11604	8741556	37.2	36040	73205	103.1	1	>7204.2
wo10-cr200-se3	12509	8741850	11574	8696942	37.9	32990	61182	85.5	1	>7218.9
wo10-cr200-se4	12509	8741234	11604	8740240	42.8	32514.3333	67764	108.4	1	>7226.9
wo10-cr200-se5	12509	8740874	11604	8740258	34.8	36667	70322	91.8	1	>7218.4
wo10-cr200-se7	12509	8741906	7144	4598798	30.2	33804.1825	46743	38.3	1	>7212.3
wo11-cr100-se0	12509	1634066	10654	1634066	3.4	87738.7339	199018	126.8	1	>7201.7
wo11-cr100-se10	12509	1633968	5796	823916	3.1	90636.2695	124333	37.2	13	>7201.3
wo11-cr200-se2	12509	3416158	10556	3309796	11.6	44726.0407	75610	69.1	1	>7204.5
wo11-cr200-se3	12509	3416916	7052	2080676	8.8	46697.5926	58933	26.2	2	>7201.9
wo12-cr100-se2	12509	2172502	9244	1706008	4.8	105972.314	194887	83.9	1	>7201.3
wo12-cr100-se3	12509	2173508	8427	1543676	5.1	99608.1257	152280	52.9	1	>7201.2
wo12-cr200-se2	12509	6560440	9244	4840680	19.7	43934.0818	82444	87.7	1	>7211.2
wo12-cr200-se3	12509	6557828	9109	4728220	22.5	40624.491	66228	63.0	1	>7207.6
wo12-cr200-se4	12509	6420904	9224	4737270	21.0	43293.0246	81199	87.6	2	>7204.0
wo12-cr200-se7	12509	6766046	7388	3497142	26.9	43853.9189	69449	58.4	1	>7205.2
wo12-cr200-se8	12509	6207724	8650	4109658	23.7	38963.7265	52826	35.6	1	>7203.4
wo12-cr200-se9	12509	6571406	4924	2013348	25.3	36482.7692	53142	45.7	18	>7204.5