Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

DANIEL REHFELDT, THORSTEN KOCH, STEPHEN J. MAHER

Reduction Techniques for the Prize-Collecting Steiner Tree Problem and the Maximum-Weight Connected Subgraph Problem

ZIB Report 16-47 (September 2016)

Zuse Institute Berlin
Takustr. 7
D-14195 Berlin

Telefon: +49 30-84185-0
Telefax: +49 30-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Reduction Techniques for the Prize-Collecting Steiner Tree Problem and the Maximum-Weight Connected Subgraph Problem

Daniel Rehfeldt* · Thorsten Koch · Stephen J. Maher

Abstract

The concept of reduction has frequently distinguished itself as a pivotal ingredient of exact solving approaches for the Steiner tree problem in graphs. In this paper we broaden the focus and consider reduction techniques for three Steiner problem variants that have been extensively discussed in the literature and entail various practical applications: The prize-collecting Steiner tree problem, the rooted prize-collecting Steiner tree problem and the maximum-weight connected subgraph problem.

By introducing and subsequently deploying numerous new reduction methods, we are able to drastically decrease the size of a large number of benchmark instances, already solving more than 90 percent of them to optimality. Furthermore, we demonstrate the impact of these techniques on exact solving, using the example of the state-of-the-art Steiner problem solver SCIP-Jack.

1 Introduction

The *Steiner tree problem in graphs* (*SPG*) is a classical $\mathcal{NP}$-hard problem [22]: Given an undirected connected graph $G = (V, E)$, costs $c : E \to \mathbb{Q}_+$ and a set $T \subseteq V$ of *terminals*, the problem is to find a minimum-cost tree $S \subseteq G$ that spans T. Although commonly cited to entail a variety of practical applications [11, 18, 30, 31, 35], the SPG rarely arises in pristine shape when it comes to modeling real-world problems [17]. Instead, one predominantly encounters variations of the classical Steiner tree problem. Three of these variations will be discussed in this paper: The *prize-collecting Steiner tree problem (PCSTP)*, the *rooted prize-collecting Steiner tree problem (RPCSTP)* and the *maximum-weight connected subgraph problem (MWCSP)*. These problems have been frequently discussed in the literature and involve various practical applications.

For exact solving of the SPG, reduction techniques have distinguished themselves as a pivotal ingredient; indeed, within the empirically most successful exact solving approach for the SPG described in the literature [30] these methods constitute the central pillar. Likewise, state-of-the-art solvers for both the (R)PCSTP and MWCSP heavily rely on such techniques. However, even though there have been notable advances for the (R)PCSTP [7, 24, 27, 34], and also certain developments for the MWCSP [3, 13], reduction techniques for Steiner problem variants fall short of achieving the same scope and potency as their kinsmen for the SPG. Against this backdrop, this paper aims at narrowing the gap towards the SPG by introducing a variety of new reduction methods for the PCSTP, the RPCSTP and the MWCSP. Furthermore, we will show the strength of these new techniques on several benchmark sets. Finally, we will demonstrate the tremendous

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, rehfeldt@zib.de, koch@zib.de, maher@zib.de

influence they can exert on exact solving by incorporating them into the state-of-the-art Steiner problem solver SCIP-Jack [17].

2 The Prize-Collecting Steiner Tree Problem

The prize-collecting Steiner tree problem (PCSTP) has been widely discussed in the literature [6, 21, 25], accompanied by several exact solving approaches [13, 25, 24, 27]. Besides SCIP-Jack, an empirically strong solver can be found in [14]. Several publications [7, 24, 27, 34] have addressed reduction techniques for the PCSTP, with the computational results reported in [34] being the best by far. Practical applications of the PCSTP range from the design of fiber optic networks [25] to computational biology [19].

Formally, the prize-collecting Steiner tree problem can be defined as follows: 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 $S = (V_S, E_S) \subseteq G$ is required such that

$$C(S) := \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_S} p_v \tag{1}$$

is minimized. For ease of presentation, we will henceforth refer to the set $T := \{v \in V \mid p_v > 0\}$ as terminals.

The *rooted prize-collecting Steiner tree problem* (*RPCSTP*) can be regarded a variation of the prize-collecting Steiner tree problem, incorporating the additional condition that one distinguished node r, called the *root*, must be part of every feasible solution to the problem.

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 to an sufficiently large value (e.g. to $p(V)+1$) this instance can be transformed to an equivalent PCSTP.

For the remainder of this section it will be presupposed that a PCSTP instance $P_{PC} = (V, E, c, p)$ is given and we further define $n := |V|$ and $m := |E|$. Thereupon, we set $s := |T|$ and for the sake of simplicity $V := \{v_1, ..., v_n\}$ and $T := \{t_1, ..., t_s\}$. Additionally, the subpath of a path Q between two vertices $v_r, v_s \in V[Q]$ will be denoted by $Q(v_r, v_s)$. The "interior" of a path Q that starts with v_k and ends with v_l is defined as $Q^\circ := (V[Q] \setminus \{v_k, v_l\}, E[Q])$. 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. In [11] 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 section, is the following [28]: A *Voronoi diagram* to (G, T, c) 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

$$v \in N(t) \Rightarrow d(v, t) \leq d(v, t') \text{ for all } t' \in T. \tag{2}$$

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 we will refer to an edge $\{v_j, v_k\}$ such that $base(v_j) \neq base(v_k)$ as *Voronoi-boundary edge*. The Voronoi diagram can be computed in $O(m + n \log n)$ by slightly adapting Dijkstra's algorithm [15] in such a way that all terminals are considered as start vertices of distance zero [28].

2.1 Basic Reductions

Several basic reduction techniques can be readily devised:

***Non-terminal of degree 1 (NTD_1)**: A non-terminal of degree 1 and its incident edge can be deleted.*

***Non-terminal of degree 2 (NTD_2)**: A non-terminal v_i of degree 2 and its incident edges $\{v_i, v_j\}$, $\{v_i, v_k\}$ can be substituted by a single edge $\{v_j, v_k\}$ with $c_{\{v_j,v_k\}} := c_{\{v_i,v_j\}} + c_{\{v_i,v_k\}}$. In the case of two parallel edges, one of lowest cost is retained.*

***Terminal of degree 1 (TD_1)**: Let $t_i \in T$ be a terminal of degree 1 and set $\{e_l\} := \delta(t_i)$. If there exists $t_j \in T$ such that $p_{t_i} \leq p_{t_j}$ and additionally it holds that*

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} must be added to the objective value of each feasible solution to P_{PC} to obtain the value of the corresponding original solution.

***Terminal of degree 2 (TD_2)**: Let $t_i \in T$ such that $\delta(t_i) = \{\{t_i, v_k\}, \{t_i, v_l\}\}$ and $p_{t_i} \leq \min\{c_{\{v_k,t_i\}}, c_{\{t_i,v_l\}}, p_{t_j}\}$ for a $t_j \in T \setminus \{t_i\}$. 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, then $\{v_k, t_i\}, \{v_l, t_i\}$ and t_i can be deleted. For a solution S' to the transformed problem P'_{PC} there is a corresponding solution S to the original problem such that the relation $C'(S') + p_{t_i} = C(S)$ holds.*

Both NTD_1 and NTD_2 were already proposed in [27]. However, to the best of our knowledge, the complete versions of the tests TD_1 and TD_2 have not been previously presented in the literature. In [34] the tests TD_1 b) and TD_2 were suggested, 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 a single terminal could then be discarded. We denote by **Degree-Test (DT)** the successive execution of those four tests. Since in each of them only vertices of degree at most 2 are checked, the worst-case complexity of DT is of $\Theta(n)$, assuming that deleting or inserting an edge can be realized in $O(1)$.

Additionally, we suggest another simple test, which we perform prior to all other reduction methods:

***Unconnected dominated vertex (UDV)**: Each vertex v_i that satisfies $p_{v_i} \leq p_{v_j}$ for a $v_j \in V \setminus \{v_i\}$ and that is not connected to any vertex (except itself) of positive prize can be deleted along with all incident edges.*

By using a modified breadth-first search with all vertices of positive weight in the initial queue, UDV can be realized with worst-case complexity of $O(m + n)$. As shown in Section 4.1, the UDV test allows to eliminate a certain portion of vertices of edges in several benchmark instances. Perhaps surprisingly, these instances contain small connected components that do not include any vertices of positive prize.

2.2 Alternative-Based Reductions

In this subsection several tests are introduced that utilize the existence of alternative solutions. They can be partitioned into exclusion and inclusion tests: The former use the argument that for

any solution containing a certain part of the graph (e.g. an edge) there is a solution of smaller or equal cost that does not contain this part. 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.

The arguably most potent alternative-based concept for the SPG is the bottleneck Steiner distance [12], paving the way for several powerful reduction methods [30]. The paramount achievement of [34] was a redefinition of this concept in the context of the PCSTP:

Let $v_i, v_j \in V$ be two distinct vertices, $\mathcal{Q}(v_i, v_j)$ the set of all simple paths between v_i and v_j and $Q \in \mathcal{Q}(v_i, v_j)$. The *local prize-collecting Steiner distance* of $Q(v_k, v_l)$ for any $v_k, v_l \in V[Q]$ is defined as

$$sd_{loc}(Q(v_k, v_l)) = \sum_{e \in E[Q(v_k, v_l)]} c_e - \sum_{v \in V[Q^\circ(v_k, v_l)]} p_v. \tag{3}$$

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

$$sd(Q) = \max_{v_k, v_l \in V[Q]} sd_{loc}(Q(v_k, v_l)). \tag{4}$$

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

$$s(v_i, v_j) = \min\{sd(Q) \mid Q \in \mathcal{Q}(v_i, v_j)\} \tag{5}$$

and the *excluding bottleneck prize-collecting Steiner distance* between v_i and v_j as

$$s^-(v_i, v_j) = \min\{sd(Q) \mid Q \in \mathcal{Q}(v_i, v_j), \{v_i, v_j\} \notin E[Q]\}. \tag{6}$$

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

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

Corollary 2. *Every edge $e_k = \{v_i, v_j\} \in E$ with $c_{e_k} \geq s^-(v_i, v_j)$ can be discarded.*

Lemma 3. *A non-terminal vertex v_i is of degree at most 2 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 $\delta(v_i) \cap \delta(\Delta)$ is not less than the weight of a minimum spanning tree for the network $(\Delta, \Delta \times \Delta, s)$.*

Corollary 2 and Lemma 3 were formulated and proved in [34], and Lemma 1 can be verified analogously. 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 the actual design of these heuristics. Thereupon, we propose two novel tests to calculate an upper bound on the bottleneck prize-collecting Steiner distance.

Given an edge $\{v_i, v_j\}$, we first 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 $c_{\{v_i, v_j\}}$. The further modifications are as follows: Starting from v_i, the edge $\{v_i, v_j\}$ is continually ignored and the algorithm does not proceed from terminals (other than v_i). If v_j has been labeled (or scanned) and the length of the corresponding path between v_j and v_i is not higher than $c_{\{v_i, v_j\}}$, the edge $\{v_i, v_j\}$ can already be eliminated. Otherwise, we run the analogous limited version of Dijkstra's algorithm from v_j, additionally stopping at all vertices that have been scanned in the course of the first run. Finally, let $q \in \mathbb{N}$ be the number of all vertices labeled or scanned during the first execution of Dijkstra's algorithm. Further,

denote each of these vertices by $v_i^{(k)}$ with $k \in \{1, ..., q\}$. If a $v_i^{(k)}$ has been labeled or scanned during the second run and further the corresponding path Q (from v_i to $v_i^{(k)}$ to v_k) satisfies $C(Q) - p_{v_i^{(k)}} \leq c_{\{v_i, v_j\}}$, the edge $\{v_i, v_j\}$ can be deleted due to Corollary 2. We will refer to this procedure as **Bottleneck Steiner Distance Circuit (SDC)** test.

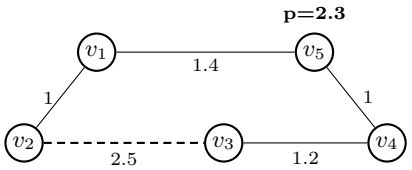

Figure 1: Segment of a PCSTP instance; edge $\{v_2, v_3\}$ (dashed) can be eliminated by the SDC test.

For the second test some groundwork is necessary. We wish to obtain for a given $\alpha \in \mathbb{N}$ to each terminal its α $\underline{d}$-nearest terminals. Duin's nearest terminals algorithm allows to compute a constant number of $\underline{d}$-nearest terminals $v_{i,1}, v_{i,2}, ...$ to each non-terminal v_i, but not to terminals. Remedy is provided by the following 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 4. *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 $t_i^{(1)}, ..., t_i^{(\alpha)}$ to t_i can be computed as follows:*

Set $t_i^{(1)} := v_{k,1}$ with k such that:

$$\underline{d}(t_i, v_{k,1}) = \min_{\{v_j, v_k\} \in \delta(N(t_i))} \{d(t_i, v_j) + c_{\{v_j, v_k\}} + d(v_k, v_{k,1})\}. \tag{7}$$

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

$$\underline{d}(t_i, v_{k,l}) = \min_{\{v_j, v_k\} \in \delta(N(t_i))} \big\{d(t_i, v_j) + c_{\{v_j, v_k\}}$$
$$+ \min\{\underline{d}(v_k, v_{k,l}) \mid l = 1, ..., r+2 : v_{k,l} \neq t_i^{(q)}, q = 0, ..., r\}\big\}. \tag{8}$$

Proof. First, note that both of the minima attained in (7) and (8) 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 $d(t_i, v_j) + c_{\{v_j, v_k\}} + d(v_k, v_{k,1})$. Hence:

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

Furthermore, for $\{v_j, v_k\} \in \delta(N(t_i))$ let Q 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 Q contains no intermediary terminal vertices. Consequently it is of cost at least $\underline{d}(t_i, v_{k,1})$, so:

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

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 Q 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 $Q(t_i, v_j)$, the edge $\{v_j, v_k\}$ and the subpath $Q(v_k, t_{min}^{(r+1)})$. 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:

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

To see the converse, note that each path Q associated with the set in (8) is composed of the components $Q(t_i, v_j)$, $\{v_j, v_k\}$ and $Q(v_k, v_{k,l})$, 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 Q. Hence, $C(Q) \geq \underline{d}(t_i, v_{k,l})$ and therefore:

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

Consequently, the lemma is established. □

Corollary 5. *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.*

Having established Lemma 4, we are in a position to propose a test that scrutinizes alternative paths with up to two intermediary terminals: We compute the four $\underline{d}$-nearest terminals (or as many as exist) to each non-terminal v_i. 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 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:

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

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 $\{v_i, v_j\} \in E$:

Mark each vertex $t_q \in L_i$ satisfying $\underline{d}(v_i, t_q) < c_{\{v_i,v_j\}}$. Thereafter, proceed for each $t_l \in L_j$ such that $\underline{d}(v_j, t_l) < c_{\{v_i,v_j\}}$ as follows. If t_l has been marked and $\underline{d}(v_i, t_l) + \underline{d}(v_j, t_l) - p_{t_l} < c_{\{v_i,v_j\}}$, delete $\{v_i, v_j\}$. 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. $\underline{d}(t_l, t_l^{(r)}) < c_{\{v_i,v_j\}}$,
2. $\underline{d}(t_l, t_l^{(r)}) + \underline{d}(v_j, t_l) - p_{t_l} < c_{\{v_i,v_j\}}$,
3. $\underline{d}(t_l, t_l^{(r)}) + \underline{d}(v_i, t_l^{(r)}) - p_{t_l^{(r)}} < c_{\{v_i,v_j\}}$,
4. $\underline{d}(t_l, t_l^{(r)}) + \underline{d}(v_j, t_l) + \underline{d}(v_i, t_l^{(r)}) - p_{t_l} - p_{t_l^{(r)}} < c_{\{v_i,v_j\}}$

are satisfied. If this is the case, $c_{\{v_i,v_j\}} > s(v_i, v_j)$ holds and we consequently remove $\{v_i, v_j\}$. 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 name the described test **Bottleneck Steiner Distance** (**SD**) test. Since the up to four close terminals are readily available, the foregoing examination of an edge can be performed in constant time. Therefore, these examinations can be accomplished for all edges in a total of $\Theta(m)$. 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 implement a test to Lemma 3: Whenever the test condition is satisfied for a $v_i \in V \setminus T$, this vertex 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 $c_{\{v_i,v_k\}} + c_{\{v_i,v_j\}}$ is inserted. In the case of two parallel edges, only one of minimum cost is retained. Analogously to the SPG [11], we call this test **Non-Terminal of Degree 3** (**NTD$_3$**).

Not only exclusion, but also inclusion tests are possible for the PCSTP: For both the Nearest Vertex (NV) and the Short Links (SL) test for the SPG [30] 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.

Initially, a simple procedure is defined that will in the remainder of this section 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 $C_{G'}$. It should be noted that the prize-collecting cost $C(G') = \sum_{e\in E'} c_e + \sum_{v\in V\setminus V'} p_v$ of G' is not increased by this procedure, since only edges are removed, which are by definition of non-negative cost

Theorem 6. *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:*

$$c_{e_2} \geq d(t_i, v_1) + c_{e_1} + d(v_1', t_j), \tag{9}$$
$$p_{t_j} \geq d(t_i, v_1) + c_{e_1} + d(v_1', t_j), \tag{10}$$
$$p_{t_i} \geq d(t_i, v_1) + c_{e_1}. \tag{11}$$

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

Proof. Let $S = (V_S, E_S)$ be feasible solution to P_{PC}. Furthermore, for this proof it will be assumed that cycle-pruning does not remove the edge e_1.

In the remainder of this proof three major cases will be differentiated, the first one being $t_i \in V_S$, the second $v_1 \in V_S$ and the third $v_1' \in V_S$. It will be demonstrated that in each of these three cases there is a solution $\tilde{S}$ that includes both t_i and the edge $e_1 = \{v_1, v_1'\}$ and moreover satisfies $C(\tilde{S}) \leq C(S)$. These deliberations prove the theorem.

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 Q_1 between t_i and v_1 as well as a shortest path Q_2 between v_1' and t_j and including the edge $\{v_1, v_1'\}$, a new subgraph $\tilde{S}$ is obtained. However, $\tilde{S}$ is not necessarily a tree, since ${Q_1}^\circ$ or ${Q_2}^\circ$ may contain a vertex of S, resulting in a cycle. In this case, $\tilde{S}$ is modified by applying cycle-pruning. Thereby, $\tilde{S}$ is rendered a tree and, containing V_S, it is of cost:

$$C(\tilde{S}) \leq C(S) - c_{e_2} + d(t_i, v_1) + c_{e_1} + d(v_1', t_j) \overset{(9)}{\leq} C(S).$$

In the complementary case $t_j \notin V_S$, construct a tree $\tilde{S}$ by once again including Q_1 and Q_2 and inserting the edge $\{v_1, v_1'\}$, followed by cycle-pruning. The tree $\tilde{S}$ is of cost:

$$C(\tilde{S}) \leq C(S) + d(t_i, v_1) + c_{e_1} + d(v_1', t_j) - p_{t_j} \overset{(10)}{\leq} C(S).$$

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

$$C(\tilde{S}) \leq C(S) - p_{t_i} + d(t_i, v_1) \overset{(11)}{\leq} C(S)$$

is obtained. Since $t_i \in \tilde{S}$, one can proceed as in case i) to find a tree $\tilde{S}'$ with $C(\tilde{S}') \leq C(\tilde{S})$ that contains both t_i and $\{v_1, v_1'\}$. Finally, if $t_i \in V_S$ holds in the first place, one can forthwith procced as in case i).

iii) Suppose $v_1' \in V_S$. If $t_i \in V_S$ but $\{v_1, v_1'\} \notin E_S$, 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 $\tilde{S}$ obtained from S by removing $\{w, \overline{w}\}$ and inserting $e_1 = \{v_1, v_1'\}$ as well as including a shortest path between v_1 and t_i (and if necessary performing cycle-pruning) is of cost:

$$C(\tilde{S}) \leq C(S) + c_{e_1} + d(v_1, t_i) - c_{e_2} \overset{(9)}{\leq} C(S).$$

On the other hand, consider the case $t_i \notin V_S$. The addition of a shortest path between v_1 and t_i and the addition of the edge e_1 (if not already present) connects t_i to S. After cycle-pruning, a tree $\tilde{S}$ of cost:

$$C(\tilde{S}) \leq C(S) + c_{e_1} + d(v_1, t_j) - p_{t_i} \overset{(11)}{\leq} C(S)$$

is obtained. □

Contrary to the NV and SL tests for the SPG, in the case of the PCSTP one cannot assume that $\{v_1, v_1'\}$ is part of at least one minimum Steiner tree. However, the following corollary allows us to contract the edge nevertheless. Initially, consider a PCSTP P'_{PC} resulting from contracting an edge of a PCSTP P_{PC}. Thereupon, a solution to P'_{PC} may correspond to several solutions to P_{PC}. In the following it is presupposed that in such a case among these solutions one of minimum cost is selected.

Corollary 7. *Assume that the premises of Theorem 6 are fulfilled and furthermore that inequality* (11) *holds strictly or $v_1 = t_i$ is satisfied. Let $P'_{PC} = (V', E', c', p')$ be the PCSTP obtained by contracting $e_1 = \{v_1, v_1'\}$ in the following way. If both $v_1 = t_i$ and $v_1' = t_j$, contract v_1' into v_i and set $p'_{t_i} := p_{t_i} + p_{t_j} - c_{e_1}$. Otherwise, define $p'_{t_i} := p_{t_i} - c_{e_1}$ and contract v_1 into v_1' if $v_1' = t_j$ or contract v_1' into v_1 if $v_1' \neq t_j$. In each case set $p'_v := p_v$ for all $v \in V' \setminus \{t_i\}$.*

Thereupon, for each optimal solution S' to P'_{PC} the corresponding solution S to P_{PC} is also optimal.

Proof. Let $S' = (V'_{S'}, E'_{S'})$ be a solution to P'_{PC} and $S = (V_S, E_S)$ the corresponding solution to P_{PC}. The cost of S' (with respect to P'_{PC}) is denoted by $C'(S')$. Additionally, assume that $e_1 = \{v_1, v'_1\}$ is contracted into v_1 (the opposite case is analogous). Initially, two cases are discussed, namely i) $v_1, t_i \notin V'_{S'}$ and ii) $v_1, t_i \in V'_{S'}$.

i) Assume $v_1, t_i \notin V'_{S'}$. In this case S' and S consist of exactly the same edges and vertices, 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:

$$C'(S') + c_{e_1} = C(S). \tag{12}$$

ii) Assume $v_1, t_i \in V'_{S'}$. Consequently, $\sum_{e \in E'_{S'}} c'_e + c_{e_1} = \sum_{e \in E_S} c_e$ and $\sum_{v \notin V'_{S'}} p'_v = \sum_{v \notin V_S} p_v$ hold, so:

$$C'(S') + c_{e_1} = C(S). \tag{13}$$

In the following assume that S' is an optimal solution to P'_{PC}. In this case, one can verify that only the two cases i) and ii) discussed above can occur. This assertion is certainly true if $t_i = v_1$, since in this case it can only hold that either $v_1, t_i \notin V'_{S'}$ or $v_1, t_i \in V'_{S'}$. In the following it will therefore be assumed that $t_i \neq v_1$. Consequently, according to the conditions of this corollary, inequality (11) holds strictly, i.e. $p_{t_i} > d(t_i, v_1) + c_{e_1}$ is satisfied. Once more, two cases need to be considered.

First, suppose $t_i \in V'_{S'}$ and $v_1 \notin V'_{S'}$. This implies for the corresponding solution S to P_{PC} that $t_i \in V_S$ and $v_1 \notin V_S$. Consequently, it holds that $\sum_{e \in E'_{S'}} c'_e = \sum_{e \in E_S} c_e$ and $\sum_{v \notin V'_{S'}} p'_v = \sum_{v \notin V_S} p_v$, so $C'(S') = C(S)$. Furthermore, according to Theorem 6 there would a solution $\tilde{S}$ to P_{PC} of cost $C(\tilde{S}) \leq C(S)$ that contains t_i, v_1, and v'_1. Due to case i) the corresponding solution $\tilde{S}'$ to P'_{PC} would be of cost $C(\tilde{S}) - c_e$, so one could infer that:

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

which contradicts the assumption that S' is optimal.

Conversely, suppose $t_i \notin V'_{S'}$ and $v_1 \in V'_{S'}$. Since $p_{t_i} > d(t_i, v_1) + c_{e_1}$ is satisfied, S' could be connected to t_i, in the graph (V', E'), by a path of cost at most

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

which would give rise to a solution of smaller cost.

Based on the above discussions, one can finally prove the corollary. To this end, let S' be an optimal solution to P'_{PC}. Thereupon, it needs to be demonstrated that S is an optimal solution to P_{PC}. Suppose this is not true, i.e. that there is a solution $\hat{S}$ to P_{PC} such that $C(\hat{S}) < C(S)$. According to Theorem 6, it may be assumed that $\hat{S}$ contains either t_i, v_1, and v'_1 or none of them. Due to (12) and (13), in both cases the corresponding solution $\hat{S}'$ to P'_{PC} would be of cost

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

contradicting the assumption that S' is optimal. Concludingly, S has to be an optimal solution to P_{PC}. □

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

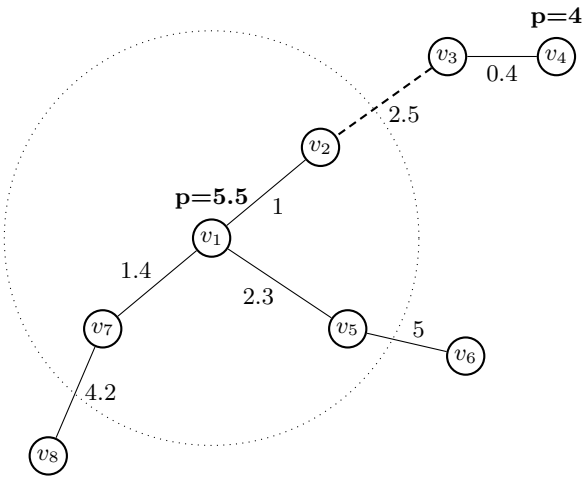

Figure 2: Segment of a PCSTP instance; setting $W := \{v_1, v_2, v_5, v_7\}$, one infers from Corollary 7 that edge $\{v_2, v_3\}$ (dashed) can be contracted.

Lemma 8. *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 $\delta(W) = \{e_1\}$, where $= \{e_1\} = \{v_1, v_1'\}$ and $v_1 \in W$, and moreover the conditions* (10) *and* (11) *of Lemma 6 hold, then for each feasible solution S to P_{PC} 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 $e_1 = \{v_1, v_1'\}$.*

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

Lemma 9. *Let t_i be a terminal of degree at least 2 and let $e_i' = \{t_i, v_i'\}$ and $e_i'' = \{t_i, v_i''\}$ be a shortest and a second shortest incident edge. Assume that there is a terminal $t_j \neq t_i$ such that:*

$$c_{e_i''} \geq c_{e_i'} + d(v_i', t_j), \tag{14}$$

$$p_{t_j} \geq c_{e_i'} + d(v_i', t_j), \tag{15}$$

$$p_{t_i} \geq c_{e_i'}. \tag{16}$$

Thereupon, for each feasible solution S to P_{PC} 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 can identify the lemma as a special case of Theorem 6. □

Condition (14) of Lemma 9 can be verified analogously to the NV test for the SPG [30] in $O(m + n \log n)$. Furthermore, with $d(v_i', t_j)$ available, the additional conditions (15) and (16) can be verified trivially by checking all incident edges of t_i. Since each edge is thereby checked at most twice, this can be rendered for the entire set of terminals in $\Theta(m)$. Additionally, the test can be extended based on the subsequent lemma.

Lemma 10. *Let t_i be a terminal of degree at least 2 and let $e_i' = \{t_i, v_i'\}$ be a shortest edge incident to t_i. Assume that there is a second edge $\{t_i, v_i''\}$ incident to t_i with $v_i'' \notin T$. If there is*

a terminal $t_j \neq t_i$ such that:

$$c_e \geq c_{e'_i} + d(v'_i, t_j) \quad \text{for all } e \in \delta(\{t_i, v''_i\}) \setminus \{e'_i\}, \tag{17}$$

$$p_{t_j} \geq c_{e'_i} + d(v'_i, t_j), \tag{18}$$

$$p_{t_i} \geq c_{e'_i}, \tag{19}$$

then for each feasible solution S to P_{PC} 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 can identify the lemma as a special case of Theorem 6. □

In the context of the PCSTP, we call the test associated with Lemma 9 and Lemma 10 that contracts edges as described in Corollary 7 by **Nearest Vertex** (**NV**) test. Since Lemma 10 can be realized with additional costs of $O(m)$, NV is of $O(m + n \log n)$.

Theorem 6 can furthermore be used to verify the following two lemmata, which in turn set the stage for a Voronoi based inclusion test.

Lemma 11. *Let t_i be a terminal and let $e'_1 = \{v_1, v'_1\}$ and $e'_2 = \{v_2, v'_2\}$ 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:*

$$c_{e'_2} \geq d(t_i, v_1) + c_{e'_1} + d(v'_1, t_j), \tag{20}$$

$$p_{t_j} \geq d(t_i, v_1) + c_{e'_1} + d(v'_1, t_j), \tag{21}$$

$$p_{t_i} \geq d(t_i, v_1) + c_{e'_1}. \tag{22}$$

Then for each feasible solution S to P_{PC} containing v_1 or v'_1, there is a solution S' of lesser or equal cost containing $\{v_1, v'_1\}$.

Proof. Defining $W := N(t_i)$, one can identify the lemma as a special case of Theorem 6. □

Lemma 12. *Let t_i be a terminal and let $e'_1 := \{v_1, v'_1\}$ be a 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:*

$$c_e \geq d(t_i, v_1) + c_{e'_1} + d(v'_1, t_j), \quad \forall\ e \in \delta(\{t_i, v''_i\}) \setminus \{e'_1\}, \tag{23}$$

$$p_{t_j} \geq d(t_i, v_1) + c_{e'_1} + d(v'_1, t_j), \tag{24}$$

$$p_{t_i} \geq d(t_i, v_1) + c_{e'_1}. \tag{25}$$

Then for each feasible solution S to P_{PC} containing v_1 or v'_1, there is a solution S' of lesser or equal cost containing $\{v_1, v'_1\}$.

Proof. Defining $W := N(t_i) \cup \{v'_2\}$, one can identify the lemma as a special case of Theorem 6. □

We denote the test associated with Lemma 11 and Lemma 12 that contracts edges as described in Corollary 7 by **Short Links** (**SL**). Additionally, this test checks whether the conditions of Lemma 8 are satisfied. SL can be realized in $O(m + n \log n)$: The computation of the Voronoi diagram requires $O(m + n \log n)$ [28] 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 12 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.

2.3 Bound-Based Reductions

This section describes a series of reduction methods that identify edges and vertices of P_{PC} for elimination by examining whether they induce a lower bound that exceeds a given upper bound. Two classed of tests will be discussed. The first type is based on the concept of the Voronoi diagram (2), the second one on a fast dual-ascent heuristic.

2.3.1 Voronoi Diagram Tests

For the SPG, bound-based methods can be built upon the *radius* concept that was introduced in [30]. Given a Voronoi diagram N and a terminal $t_i \in T$, $radius(t_i)$ is defined as the minimum cost of any path containing t_i and leaving $N(t_i)$ [30]. This concept can be generalized for the (R)PCSTP as follows:

$$pcradius(t_i) := \min\{radius(t_i), p_{t_i}\},\ t_i \in T. \tag{26}$$

The adaptation of the *radius* definition is necessary because a feasible solution to an SPG contains for each Voronoi region a path that leaves this region, whereas this assumption does not hold for the (R)PCSTP—since feasible solutions to the (R)PCSTP do not need to contain all terminals. Definition (26) sets the stage for a series of lemmata and corollaries presented hereinafter that allow to eliminate both vertices and edges. For ease of understanding, it is presupposed that all terminals are ordered such that $pcradius(t_1) \leq pcradius(t_2) \leq ... \leq pcradius(t_s)$. The first lemma can be seen as an extension of a test already known for the SPG [30]. The reader is reminded that we have defined $s := |T|$.

Lemma 13. *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. Assume that there is a minimum Steiner tree $S = (V_S, E_S)$ such that $v_i \in V_S$. Denote the (unique) path in S between v_i and a terminal $t_j \in V_S$ by Q_j and the set of all such paths by $\mathcal{Q}$. First, note that $|\mathcal{Q}| \geq 2$, because if $\mathcal{Q}$ just contained one path, say Q_l, the single-vertex tree $\{t_l\}$ would be of smaller cost than S, contradicting the initial assumption that the latter is of minimum cost. Second, if a vertex v_k is contained in two distinct paths in $\mathcal{Q}$, the subpaths of these two paths between v_i and v_k coincide. Otherwise there would need to be a cycle in S. Additionally, there are at least two paths in $\mathcal{Q}$ 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 $Q_k \in \mathcal{Q}$ and $Q_l \in \mathcal{Q}$ such that their combined number of Voronoi-boundary edges is minimal and $V[Q_k] \cap V[Q_l] = \{v_i\}$ holds. Further, define $\mathcal{Q}^- := \mathcal{Q} \setminus \{Q_k, Q_l\}$. For all $Q_r \in \mathcal{Q}^-$, denote by Q_r' the subpath of Q_r between t_r and the first vertex not in $N(t_r)$. Suppose that Q_k has an edge $e \in E_S$ in common with a Q_r': Consequently, Q_l cannot have any edge in common with Q_r, because this would require a cycle in S. Furthermore, according to the preceding observations, Q_k and Q_r have to contain a joint subpath including v_i and e. But this would imply that Q_k contained 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 Q_l and Q_r being edge disjoint, Q_r would have initially been selected instead of Q_k.

Following the same line of argumentation, one validates that likewise Q_l has no edge in common with any Q_r'. Conclusively, the paths Q_k, Q_l and all Q_r' are edge disjoint. Using their

combined cost, one can obtain a lower bound on the cost of S by:

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

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

A Steiner vertex v_i can be discarded if its associated lower bound, specified in Lemma 13, 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 14. *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.*

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

Lemma 15. *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 2 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.*

The last lemma can be utilized to show that in all optimal solutions the terminal t_i is either not contained or of degree 1. To make use of this information, the following simple reduction test can be applied.

Lemma 16. *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 1 in at least one minimum Steiner tree, there exists a minimum Steiner tree not containing t_i.*

If the premises of Corollary 14 or Lemma 16 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 17. *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*

$$L := c_{\{v_i, v_j\}} + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{s-2} pcradius(t_q) \tag{27}$$

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

$$L := c_{\{v_i,v_j\}} + \min\left\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1})\right\} + \sum_{q=1}^{s-2} pcradius(t_q) \tag{28}$$

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

Lemma 18. *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* 3 *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 cost of* S.

Occasionally, an even stronger bond can be obtained by utilizing the following lemma, which can be seen as a generalization of a lemma introduced in [30].

Lemma 19. *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* $\{v, w\} \in E$ *such that* $v \in N(t_i)$ *and* $w \in N(t_j)$. *Additionally, define a cost function* c' *on* E' *by*

$$c'_{\{t_i,t_j\}} := \min\{\min\{p_{t_i}, p_{t_j}, d(t_i, v_i), d(t_j, v_j)\} + c_{\{v_i,v_j\}} \mid v_i \in N(t_i), v_j \in N(t_j)\}.$$

The weight, with respect to c', *of a minimum spanning tree for* G' *is a lower bound on the weight of any Steiner tree for* (V, E, c, p)

All antecedent bound-based tests can be easily modified to make use of the bound obtained from Lemma 19. For instance, one may adapt Lemma 17 by substituting the value L by the weight of a minimum spanning tree for G' minus the weight of its longest edge (analogously to the procedure for the SPG described in [30]).

The test associated with the introduced bound-based reduction approaches is denoted by **Bound** (**BND**) test. It works with an upper bound computed by a constructive heuristic that was introduced in [17]. The worst-case complexity, which would otherwise be of $O(m + n \log n)$, is dominated by the constructive heuristic, which exhibits a complexity of $O(s(m + n \log n))$ but is empirically very fast.

2.3.2 Dual-Ascent Tests

For the final bound-based test we transfer a powerful reduction technique for the Steiner tree problem in graphs, the *dual-ascent method* [11, 30]. While this method is a bound-based reduction test, it is distinctively different from the bound-based methods previously introduced in this paper.

To set the stage, we first define the *Steiner arborescence problem* (*SAP*): Given a directed graph $D = (V, A)$, costs $c : A \to \mathbb{Q}_{\geq 0}$, a set $T \subseteq V$ of terminals and a root $r \in T$, a directed tree (arborescence) $S = (V_S, A_S) \subseteq D$ of minimum cost $\sum_{a \in A_S} c_a$ is required such that for all $t \in T$ the tree S contains a directed path from r to t.

Considering an SAP (V, A, c, r), we associate with each arc $a \in A$ a variable y_a indicating whether a is contained in the Steiner arborescence ($y_a = 1$) or not ($y_a = 0$). Thereupon, an IP formulation can be stated as [36]:

Formulation 1. *Directed Cut Formulation*

$$\begin{aligned}
&min \quad c^T y && && (29)\\
&y(\delta^-(W)) \;\geq\; 1 && \text{for all } W \subset V,\ r \notin W, W \cap T \neq \emptyset, && (30)\\
&y_a \;\in\; \{0, 1\} && \text{for all } a \in A. && (31)
\end{aligned}$$

In [36] a dual-ascent algorithm for the SAP was introduced that, empirically, both provides strong lower bounds and allows for fast computation, defying its worst-case time complexity of $O(|E| \min\{|V||T|, |E|\})$ [30]. Efficient implementations of the algorithm can be found in [11] and [29]. We use the latter implementation combined with the heuristic guiding-solution criterion suggested in [30]. At termination, dual-ascent provides a dual solution to the LP relaxation of Formulation 1, involving directed paths along arcs with a reduced cost of 0 from the root to each additional terminal. This information can be used for the SPG reduction method dual-ascent [11, 30], which is based on the possibility to transform the SPG to the SAP, see for instance [32].

In the following we demonstrate the dual-ascent approach for the RPCSTP, but it can be naturally extended to the PCSTP and MWCSP. The first step is to transform a given RPCSTP to an SAP, as we already described in [33]:

Transformation 1 (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* $i \in \{1, ..., s\}$, *setting 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) RPCSTP instance

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

Figure 3: Illustration of an RPCSTP instance with root r (left) and the equivalent SAP obtained by Transformation 1 (right). Terminals are drawn as squares and Steiner vertices as circles (with those of positive weight enlarged).

Lemma 20 (RPCSTP to SAP)**.** *Let* $P' = (V', A', T', c')$ *be an SAP obtained from an RPCSTP* $P = (V, E, c, p)$ *by applying Transformation 1. Each solution* $S' = (V'_{S'}, A'_{S'})$ *to* P' *can be mapped to a solution* $S = (V_S, E_S)$ *to* P *defined by:*

$$V_S := \{v \in V \mid v \in V'_{S'}\} \tag{32}$$
$$E_S := \{\{v, w\} \in E \mid (v, w) \in A'_{S'} \text{ or } (w, v) \in A'_{S'}\} \tag{33}$$

If S' *is an optimal solution to* P', *then* S *is an optimal solution to* P *and their objective values are equal.*

With this transformation at hand, let $v_i \in V/T$ and $S^\star$ be an optimal Steiner arborescence to a given SAP instance (V, A, c, r) and L_{DA} the lower bound obtained by dual-ascent. If $S^\star$ contains v_i, the weight of $S^\star$ can be bounded from below by L_{DA} plus the length (with respect to the reduced costs provided by dual-ascent) of a shortest path from the root to v_i and the length of a shortest path from v_i to a $\underline{d}$-nearest terminal (other than the root) to v_i. Hence, v_i can be deleted if the just defined bound exceeds a known upper bound U. An analogous test can be stated for the elimination of arcs. Since each RPCSTP instance can be transformed to an SAP, the above deliberations forthwith set the stage for an RPCSTP reduction technique: Whenever a vertex can be deleted in the SAP, the same is true for its counterpart in the analogous RPCSTP. Similarly, if two anti-parallel arcs of the SAP have been shown to be removable, the corresponding edge of the RPCSTP can be discarded. Finally, a test to replace vertices by edges that is analogous to NTD_k can be used.

In addition to these methods already known for the SPG, the graph transformation from RPCSTP to SAP sets the stage for two new tests. First, terminals of the RPCSTP can be deleted if the corresponding vertex t_i in the SAP is shown to be removable. Second, if the reduced cost of an arc (r', t_i') is higher than $U - L_{DA}$, we can deduce that the vertex t_i is part of an optimal solution to P. Consequently, we can set its prize to infinity, which possibly allows several tests (such as SDC) to perform additional reductions. Moreover, we can create a new SAP using t_i as the root, which can pave the way for additional eliminations by the above procedure. In our reduction package the maximum number of root changes and subsequent dual-ascent runs is limited to 15.

As with the BND test, all dual-ascent based reduction methods can be extended to the case of equality if a Steiner tree corresponding to the upper bound U is given. We call the entire procedure of computing the reduced costs on the SAP, computing an upper bound to the original problem and eliminating vertices and edges as well as fixing terminals **Dual-Ascent** (**DA**) test.

Obviously, one question remains, namely how to obtain an upper bound for the DA test. As will be shown in the following, the availability of the reduced costs provided by DA allows to transfer an heuristic approach already known for the SPG.

Ascend-and-Prune

The *ascend-and-prune* heuristic has been demonstrated to be a powerful device for the Steiner tree problem in graphs [30]. An extension of the heuristic to the (R)PCSTP (and the MWCSP) was essentially impeded by the lack of two techniques: Transformations of (R)PCSTP and MWCSP to SAP and powerful reduction methods. With both transformations and strong reduction techniques at hand, the ascend-and-prune approach can be extended beyond the SPG.

Let P be the original RPCSTP and P' the SAP resulting from Transformation 1. The

(a) RPCSTP solution

(b) SAP solution

Figure 4: Illustration of a solution to the RPCSTP instance of Figure 3 (left) and an equivalent solution to the corresponding SAP (right).

ascend-and-prune heuristic attempts to find a good solution on the subproblem $\tilde{P}$ constituted by the undirected edges of P corresponding to zero-reduced-cost paths in P' from the root to all additional terminals. This approach is motivated by the assumption that notable similarities exist between an optimal (or nearly optimal) Steiner tree and the LP solution corresponding to the reduced costs provided by dual-ascent.

To find a solution to $\tilde{P}$, the concept of the *prune* heuristic [30] for the SPG is transfered to the RPCSTP. This extension is possible by virtue of the new reduction tests, in particular BND, introduced in this paper. The heuristic works as follows: After applying all previously introduced reduction techniques on $\tilde{P}$, we use an heuristic extension of the BND test to further reduce $\tilde{P}$. While for the original BND test the upper bound is provided by a primal solution, in the prune heuristic the bound is decreased in such a way that in each iteration a constant amount of edges and vertices is eliminated. Thereupon, all (exact) reductions methods are executed on the reduced problem, motivated by the assumption that the (possibly inexact) eliminations performed by the bound-based method will allow for further reductions. This procedure is reiterated until not more than two terminals remain. To avoid infeasibility, we compute a Steiner tree by using the shortest path constructive heuristic described in [32] and forbid eliminations of any of its vertices or edges by the inexact BND test. In [33] we implicitly used the previously described reduction techniques to implement ascend-and-prune for the (R)PCSTP and MWCSP and demonstrated its ability to find strong, often optimal upper bounds within short time.

2.4 Reiteration and Ordering

Studying different combinations and orderings of reduction techniques, we have come to the conclusion that the tests are not overly sensitive to the order of their execution (as with the SPG [30]). However, this 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 to perform the two SD test variants prior to the NTD_3 test, since the former reduces, if successful, the degrees of vertices. Additionally, first performing the SD test allows us to reuse the computed $\underline{d}$-distances for the NTD_3 test. Similarly, due to the NTD_3 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.

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 were eliminated during the last run. Thereby, one obtains the same asymptotic time bound as the most expensive performed reduction test; due to DA this bound is of $O(|E| \min\{|V||T|, |E|\})$. This bound implies in particular that the reduction package is of polynomial worst-case complexity. 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. Moreover, the BND test is executed only if at most three percent of all vertices are terminals; we have observed that the test is otherwise of very little effect.

Additionally, we have implemented an exhaustive version of the reduction package, in which the loop as well as all tests are reiterated until no more reductions can be performed. This loop is not supposed to be used a preprocessing step for exact solving (unless the instance to be solved is known to be hard), but rather for demonstrating the capacity of the reduction tests.

3 The Maximum-Weight Connected Subgraph Problem

The third Steiner variant to be discussed in this paper is the maximum-weight connected subgraph problem (MWCSP): Given an undirected graph $G = (V, E)$ and node weights $p : V \to \mathbb{Q}$,

the objective is to find a connected subgraph $S = (V_S, E_S) \subseteq G$ such that $\sum_{v \in V_S} p_v$ is maximized. The problem has been discussed in various publications [4, 10, 20]. Practical applications of the MWCSP can be found in diverse areas such as wildlife conservation [9], computational biology [10] and computer vision [8].

Several publications about the MWCSP have addressed exact solving approaches [4, 5, 13, 14]. However, compared to the SPG very little research has been performed regarding reduction techniques for the MWCSP [3, 13]. This provided an incentive for us to develop various novel techniques, which will be subsequently introduced along with several others known from the literature.

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 connected component of maximum vertex weight; in the case of only non-positive vertex-weights, the empty set constitutes an optimal solution. For a given solution $S = (V_S, E_S)$ to an MWCSP we denote its weight by $C(S) := \sum_{v \in V_S} p_v$. Note that to each feasible solution there is a solution of the same weight that additionally is a tree—such a solution can for example be obtained by the cycle-pruning procedure introduced in Section 2. For simplicity, in the following lemmata it will be assumed that each solution to an MWCSP is a tree.

Furthermore, throughout this section it will be presupposed that an MWCSP instance $P_{MW} = (V, E, p)$ is given. The remainder of the notation is analogous to the one introduced in Section 2.

3.1 Basic Reductions

Analogous to the PCSTP, reductions for vertices of low degree are readily conceived. The first of the following two simple tests was already suggested in a less general form in [13] (namely only for negative vertices of degree 0):

***Vertex of degree 0 (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 $v_{max} \neq v_i$.*

***Vertex of degree 1 (VD$_1$)**: If there is vertex v_i of degree 1 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.

The next two reduction tests can be found in both [3] and [13].

***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 Q between two vertices v_i, v_j, containing only non-positive interior vertices of degree 2 and fulfilling $|V[Q]| > 3$ can be replaced by a node v' with $p_{v'} := \sum_{v \in V[Q^\circ]} p_v$ 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.*

3.2 Alternative-Based Reductions

Alternative-based reductions for the MWCSP have not been comprehensively researched in the literature yet. This section aims for a more extensive study by introducing many novel reduction techniques, which, to the best of the authors' knowledge, have not been previously published.

The subsequent reduction test was tentatively suggested in [13], to be later generalized in [3]. Only the latter version is stated here:

Adjacent Neighbourhood Subset (ANS) [1] *Let v_i and v_j be two distinct vertices. If $p_{v_i} \leq 0$, $p_{v_i} \leq p_{v_j}$ and*

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

then v_i and all incident edges can be removed.

However, even this revised version can be notably generalized:

Lemma 21. *Let $v_i \in V$ and $W \subseteq V \setminus \{v_i\}$, $W \neq \emptyset$ such that $(W, E[W])$ is connected and $\sum_{w \in W : p_w < 0} p_w \geq p_{v_i}$ holds. If*

$$\big\{v \in V \setminus W \mid \{v_i, v\} \in E\big\} \subseteq \big\{v \in V \setminus W \mid \{w, v\} \in E, w \in W\big\} \tag{34}$$

is satisfied, then there is at least one optimal solution that does not contain v_i.

Proof. Suppose that there is an optimal solution $S = (V_S, E_S)$ containing v_i. Since

$$\sum_{w \in W : p_w < 0} p_w \geq p_{v_i}$$

it can be inferred that $p_{v_i} \leq 0$. Therefore, it can be assumed that v_i is of degree at least 2 in S (if v_i is of degree 1 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:

$$C(S') = \sum_{v \in V_{S'}} p_v \geq \sum_{\substack{w \in W \\ p_w < 0}} p_w + C(S) - p_{v_i} \geq C(S). \tag{35}$$

Due to (35) it holds that S' is optimal. Furthermore, by construction S' does not contain v_i, so the statement of the lemma is established. □

Affiliated to Lemma 21 we suggest the subsequent reduction test. For each vertex $v_j \in V$ proceed as follows: Define W_j as the union of v_j and all non-negative adjacent vertices of v_j. Next, we mark all neighboring vertices of W_j. Finally, we check for all non-positive neighboring vertices v_i of W_j such that $p_{v_i} \leq p_{v_j}$ whether condition (34) holds. If this is the case, v_i can be deleted. In the following, this test is referred to as **Connected Neighborhood Subset (CNS)**.

Furthermore, we suggest an extension of CNS for which not only all v_i adjacent to W_j but all $v_i \in V \setminus W_j$ such that $p_{v_i} \leq p_{v_j}$ are scrutinized. We call this variant **Advanced Connected Neighborhood Subset** (**ACNS**) test.

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

[1] The reduction test was originally named "Neighbourhood Subset", abbreviated "NS", test in [3], but owned to historical considerations we have decided to substitute it. Especially in Germany, "NS" has commonly been used as an abbreviation for National Socialism.

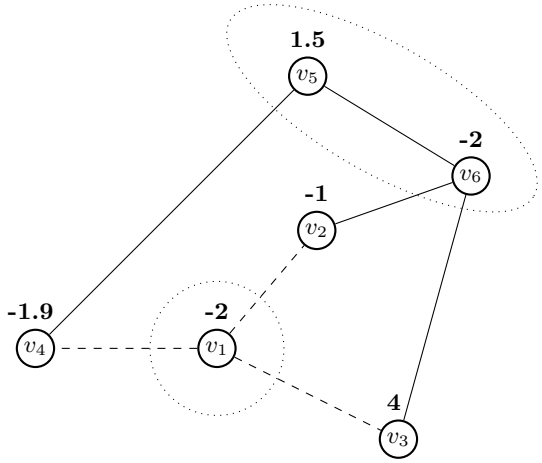

Figure 5: Segment of an MWCSP instance. Lemma 21 guarantees that vertex v_1 and its incident edges (dashed) can be eliminated, since each neighbor of v_1 is a neighbor of either v_5 or v_6

Lemma 22. *Let $e_k = \{v_i, v_j\} \in E$. If there is a path Q between v_i and v_j such that $e_k \notin E[Q]$ and $p_{v_l} \geq 0$ for all $v_l \in V[Q^\circ]$, then there is at least one optimal solution that does not contain e_k.*

Proof. Let S be a solution to P_{MW} containing e_k. By removing e_k from S one obtains two connected subgraphs S_1, S_2 such that $v_i \in S_1$, $v_j \in S_2$. Since Q connects v_i and v_j, there are vertices $v_r \in V[S_1] \cap V[Q]$, $v_s \in V[S_2] \cap V[Q]$ such that for the induced subpath $Q(v_r, v_s)$ it holds that $V[S_1] \cap V[Q(v_r, v_s)] = \{v_r\}$ and $V[S_2] \cap V[Q(v_r, v_s)] = \{v_s\}$. Reconnecting S_1 and S_2 by $Q(v_r, v_s)$ one yields a new connected subgraph $\tilde{S}$ of weight at least $C(S)$.

These discussions prove the lemma, since they imply that to each solution that contains e_k, a solution of greater or equal weight not containing e_k can be constructed. □

We suggest a reduction test based on Lemma 22, 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, to probe an edge $\{v_i, v_j\}$ it suffices to check all adjacent vertices of both v_i and v_j. 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 call this test **Non-Negative Paths (NNP)**.

The development of the next reduction strategy originated from the observation that after the NPC test has been performed each non-positive chain is reduced to a single non-positive vertex of degree 2. 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, similar to that of the bottleneck Steiner distance: Let $v_i, v_j \in V$ be two distinct vertices and $\mathcal{Q}(v_i, v_j)$ the set of all simple paths between v_i and v_j. In the context of the MWCSP we define the *interior cost* of such a subpath as:

$$C^\circ(Q(x,y)) := \sum_{v \in V[Q(x,y)] \setminus \{x,y\}} p_v. \tag{36}$$

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

$$\hat{l}(Q) := \min_{x,y \in V[Q]} C^\circ(Q(x,y)). \tag{37}$$

Given two disjoint connected subgraphs that can be connected by a subpath of a given path Q, by $\hat{l}(Q)$ 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 that we define as

$$\hat{d}(v_i, v_j) := \max\{\hat{l}(Q) \mid Q \in \mathcal{Q}(v_i, v_j)\}. \tag{38}$$

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

Lemma 23. *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*

$$\hat{d}(v_j, v_k) > p_{v_i} \tag{39}$$

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, yielding another optimal solution (or a solution greater weight, which would forthwith contradict the assumption of S being optimal). Removing v_i as well as its incident edges from S, we obtain two connected subgraphs S_1 and S_2. Let Q be a path between v_j and v_k such that $\hat{l}(Q) = \hat{d}(v_j, v_k)$. Additionally, let $Q(v_1', v_2')$ be a subpath of Q 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 $Q(v_1', v_2')$ a connected subgraph S' is obtained, which is of cost:

$$\begin{aligned} C(S') &= C(S) + C^{\circ}(Q(v_1', v_2')) - p_{v_i} \\ &\geq C(S) + \hat{l}(Q) - p_{v_i} \\ &= C(S) + \hat{d}(v_j, v_k) - p_{v_i} \\ &\overset{(39)}{>} C(S). \end{aligned}$$

This proves the lemma. □

We suggest the following reduction test to utilize Lemma 23: First substitute each edge by two anti-parallel arcs such that for each arc $a = (v, w)$:

$$c'_a = \begin{cases} -p_w, & \text{if } p_w < 0 \\ 0, & \text{otherwise} \end{cases}$$

This procedure results in 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 2 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 $\vec{Q}'_k$ and $\vec{Q}'_j$ from v_k and v_j respectively to a vertex $v_l \in V$ have been found, denote by Q_{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 $\hat{l}(Q_{jk}) = \min\{-C'(\vec{Q}'_j), -C'(\vec{Q}'_k), -C'(\vec{Q}'_j) - C'(\vec{Q}'_k) + p_{v_l}\}$;

2. if $p_{v_l} \leq 0$, then $\hat{l}(Q_{jk}) = -C'(\vec{Q}'_j) - C'(\vec{Q}'_k) - p_{v_l}$.

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

If $\hat{l}(Q_{jk}) \geq p_{v_i}$, the vertex v_i and its incident edges are deleted. We denote this procedure by **Non-Positive Vertex of Degree 2** (**NPV$_2$**) test.

The concept of the vertex weight distance not only gives rise to Lemma 23 and the affiliated NPV$_2$ test, but furthermore leads the way to a far more general and powerful result:

Lemma 24. *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 $\Delta' \subseteq \Delta$ denote by $K_{\Delta'} := (\Delta', \Delta' \times \Delta', \hat{d})$ the complete graph on Δ', with edge weight $\hat{d}(v_j, v_k)$ for an edge $\{v_j, v_k\} \in \Delta' \times \Delta'$. If for each $\Delta' \subseteq \Delta$ 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.*

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$_O$, VD$_1$ and Lemma 23 respectively. So suppose $|\Delta| > 2$ and let $S = (V_S, E_S)$ be a connected subgraph 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. Due to the premises of the lemma, there is a maximum spanning tree $T_{\Delta'}$ on $K_{\Delta'}$ of weight greater than p_{v_i}. The solution S can be segmented into k connected subgraphs $S_1, .., S_k$ 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 connected subgraphs to obtain a new connected subgraph 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 Q_{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 $\{v'_r, v'_s\} \subseteq \Delta$, it holds that $\hat{d}(v'_r, v'_s) > p_{v_i}$ (due to the premises of the lemma). Consequently, v_i is not contained in Q_{rs}. Analogously to the proof of Lemma 23, S_r and S_s can be linked to a connected subgraph S'_1 by a subset of Q_{rs} that is of weight at least $\hat{d}(v'_r, v'_s)$. Thereupon, S'_1 can be linked in the same way to a connected subgraph S_l, $k \in \{1, ..., k\} \setminus \{r, s\}$, bringing forth a connected subgraph S'_2 that once more does not contain v_i. Reiterating in this manner, S'_2 can be joined with all not yet incorporated connected subgraphs S_q, finally yielding a connected subgraph S'_{k-1}. This connected subgraph does not contain v_i and is of weight:

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

Consequently, the claim is established. □

Corollary 25. *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 $\Delta' \subseteq \Delta$ define $K_{\Delta'} := (\Delta', \Delta' \times \Delta', d^-_{vw})$. If for each $\Delta' \subseteq \Delta$ 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.*

Proof. The proposition can be verified largely in line with the proof to Lemma (24). As the sole amendment, to demonstrate that for joining the connected subgraphs $S_1, ..., S_k$ 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 3, 4 and 5 and refer to the corresponding methods as **Non-Positive Vertex of Degree k** (**NPV$_\mathbf{k}$**) test ($k \in \{3, 4, 5\}$). In line with the proceeding for the bottleneck based tests to both the SPG [30] 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 25, it is removed along with all incident edges.

Finally, the DA test introduced in Section 2 for the RPCSTP is also used for the MWCSP, based on the transformation of MWCSP to SAP that we introduced in [33].

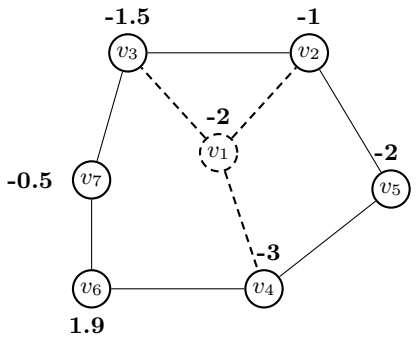

Figure 6: Segment of an MWCSP instance. The NPV$_3$ test shows that vertex v_1 and its incident edges (dashed) can be eliminated.

3.3 Reiteration and Ordering

The settings for the reiteration of the reduction methods within the overall loop as well as for the truncation of the latter are identical to the settings for the PCSTP, see Section 2.4.

Naturally, empirically less expensive tests are performed first. In this way, the DA test, which is both theoretically and empirically the most expensive reduction method, 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. For the extensive reduction package we execute the advanced CNS test once the reduction loop has terminated and run the entire reduction loop again if eliminations can be performed by the advanced CNS test.

4 Computational Results

This section evaluates the various reduction methods introduced in this paper on several benchmark test sets for the MWCSP, the PCSTP, and the RPCSTP. To this end, computationally experiments are performed on problems collected for the 11th DIMACS Challenge [1]. In addition, a number of MWCSP instances collected from a computational biology application are considered.

The 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 [16]. For the computations with SCIP-Jack, CPLEX 12.6.2[2] was employed as the underlying LP solver. For all computations a time limit of two hours was set.

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

When providing average values we will proceed in two ways: First, for the remaining vertices (edges) of a test set the percentage of remaining vertices (edges) for each instance is computed and the arithmetic mean of these percentages is provided. Second, the solving time of both the reduction package and SCIP-Jack are computed by taking the *shifted geometric mean* [2]: 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

$$\sqrt[k]{\prod_{i=1}^{k}(t_i + s)} - s. \tag{40}$$

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 endeavor to reduce the effect of very easy instances on the mean value. In the subsequent computations, a shift of $s = 1$ is used.

Finally, if an instance has been solved to optimality by the reduction techniques, in the statistics the number of remaining edges and vertices is assumed to be 0.

4.1 (Rooted) Prize-Collecting Steiner Tree Problem

Table 1 provides a brief description of the five benchmark test sets that are used for the computational evaluation of the (R)PCSTP reduction techniques. In each line the name of the 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.

Name	Instances	$\|V\|$	Status	Description
JMP	34	100-400	solved	Sparse instances of varying structure, introduced in [21].
CRR	80	500-1000	solved	Instances with up to $25\,000$ edges, based on the C and D test sets of the SteinLib [25].
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 [25].
Cologne2	15	1801-1810	solved	

Table 1: Classes of (R)PCSTP instances.

The results given in Table 2, in which we illustrate the results of extensive runs, bespeak the strength of the employed reduction techniques. On average (with respect to the arithmetic mean) the number of edges is reduced by more than 90 percent. Even for the hard PUCNU test set, whose SPG predecessor allowed for less than one percent of reduction by the reductions techniques described in [30], more than 25 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 [25], which renders the results even more notable. All Cologne1 and Cologne2 instances are solved to optimality during presolving, with no instance requiring more than 0.26 seconds. Furthermore, of the 34 JMP instances 33 can be solved by reduction methods alone. Similarly, the reduction techniques are able to solve 74 of the 80 CRR instances. 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	0.07	0.03	0.01
CRR	3.28	0.34	0.22
PUCNU	85.34	74.60	0.57
Cologne1	0	0	0.09
Cologne2	0	0	0.33
all	11.23	8.54	0.19

Table 2: Computational results of the extensive (R)PCSTP reduction package.

Another interesting aspect is the individual performance of the reduction tests. To this end, we have listed the results of running each test exclusively (until no more eliminations are possible) in Table 3. Before each run, however, we perform the UDV test to dispose of vertices with a weight of 0 that are not connected to any positive-weight vertex. The results of each reduction method are given as the average over all instances of the test sets listed in Table 1. The most effective reduction test is DA, eliminating more than half of the vertices and two thirds of the edges on average. However, it is also the most expensive test, both on average and individually—with one instance requiring almost five seconds to be processed. Remarkably, the second most effective method is BND, notwithstanding the fact that it is only performed for less than forty percent of the instances. The test is also much faster than DA both on average and in the worst-case—BND processes each instance in less than a second.

The SD/SDC test is able to eliminate a notable number of edges, but is on the other hand the second most expensive reduction method. Finally, the NV/SL and the NTD_3 tests are both notably fast, but while the NV/SL test is able to reduce the number of vertices by more than 15 percent and the number of edges by more than ten percent, NTD_3 is the least effective of all reduction methods. However, it should be noted that the reduction methods described in this paper usually become more effective in combination, since eliminations performed after the execution of a specific reduction method often set the stage for renewed reductions by this method.

Reduction Method	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]
UDV	99.86	99.96	0.01
DT	82.04	87.61	0.01
SD/SDC	100	78.33	0.12
NTD_3	96.55	97.40	0.02
NV/SL	85.75	89.83	0.01
BND	81.04	77.33	0.02
DA	43.01	30.70	0.14

Table 3: Computational results of running the (R)PCSTP reduction tests individually.

Having discussed our reduction approach for the (R)PCSTP, we furthermore aspire to classify our experimental results with respect to related publications. The most effective reduction techniques published in the literature are from [34], clearly surpassing previous results [23, 25, 27]. The author performed computational experiments on the CRR instances and a subset of the JMP test set. Unfortunately, no run times are provided in [34]. However, the fact that so called *expansions* of reduction tests were used, which can be exceedingly time-consuming [30], suggests that these computations require far more time than our approach. No expansions of reduction methods are deployed in this paper, since the purpose of the reduction approach is a more subordinate one, namely to also serve as a component for fast exact solving.

Nevertheless, the reduction package introduced in this paper still achieves notably better

results on both test sets than the approach in [34]: For the JMP test sets the number of remaining edges constitutes less than ten percent of the number of remaining edges reported in [34] and for the CRR test set it is even less than four percent. Noticeably, the difference between our package and the results given in [34] become most pronounced on instances of small $|T|$. This phenomenon is epitomized in the D11-A instance, originally consisting of 5000 edges and 1000 vertices: While our reduction package eliminates all edges (more than 3000 by the BND test), the approach described in [34] removes merely 23 vertices and 1029 edges. Detailed comparison between the results reported in [34] and those of our (R)PCSTP reduction package is provided in the appendix in Table 8 and Table 9.

4.2 Maximum-Weight Connected Subgraph Problem

This section provides computational results of the MWCSP reduction package introduced in this paper on three test sets described in Table 4. The ACTMOD and JMPALMK instances were all solved to optimality in the course of the 11th DIMACS Challenge, while the SHINY test set has only been recently introduced [26]. The SHINY test set remains unsolved since there exists one instance that could not be solved in [26], neither by their own algorithm nor by the MWCSP solver Heinz2 Version 2.1 [13].

Name	Instances	$\|V\|$	Status	Description
JMPALMK	72	500-1500	solved	Euclidean, randomly generated instances introduced in [4].
ACTMOD	8	2034-5226	solved	Real-world instances with up to 93 394 edges derived from integrative biological network analysis [10].
SHINY	39	232-3828	unsolved	Sparse, real-world instances from network enrichment analysis in computational biology [26].

Table 4: Classes of MWCSP instances.

Our reduction package manages to solve all 119 MWCSP instances that were part of the computational experiments to optimality, exhibiting an overall average run time of 0.08 seconds. The JMPALMK instances are the fastest to be solved, with the maximum run time being below 0.2 seconds. The ACTMOD instances need more time, 0.3 seconds on average and 1.7 seconds for the longest run. Finally, the vast majority of the SHINY test set is solved in less than 0.1 seconds, but three instance require more time: 5.5 (*25e857e14393*), 6.5 (*25e81700dead*), and 7.4 (*25e814a792c4*) seconds. Notably, prior to this publication instance *25e814a792c4* remained unsolved. As a result of our reduction package this instance is solved to optimality with an objective value of 1083.308.

In order to evaluate the individual strength of the reduction methods, we list in Table 5 the results of running them exclusively (until no more reductions are possible) on all instances of the JMPALMK, ACTMOD and SHINY test sets. For the computations the BT and NNP tests are performed in tandem, as the latter requires that all adjacent non-negative vertices have been contracted.

Reduction Method	Remaining Vertices[%]	Remaining Edges[%]	∅ Time [s]
UNPV/BT	59.12	57.55	0.03
ACNS	45.00	35.56	0.01
BT/NNP	11.76	12.54	0.05
NPV_k	87.42	89.65	0.01
DA	35.63	20.14	0.15

Table 5: Computational results of running the MWCSP reduction tests individually.

As opposed to the (R)PCSTP, the DA test achieves only the second place for the MWCSP in terms of effectiveness, with the BT/NNP test being first. Notably, this combined test is not only three times faster than DA, but furthermore allows on average for a reduction to almost ten percent of the original edges and vertices. Likewise, the UNPV/BT and the advanced CNS test are fast as well as effective, with the latter being clearly superior in both categories. Finally, the NPV_k test exhibits a somewhat lesser performance than the previous tests: although considerably fast, it leaves almost 90 percent of both edges and vertices untouched. However, NPV_k usually becomes more effective after the execution of the other reduction methods, since these methods reduce the degree of the vertices and also facilitate the detection of paths of high vertex weight bottleneck length (37), for instance by contracting vertices of non-negative weight.

For the combined ACTMOD and JMPALMK instances [13] report 16.8 percent vertices and 5.6 percent of edges remain. Unfortunately, no run times for the preprocessing were reported by the authors.

In [3] the results of their reduction package on the ACTMOD instances is reported, yielding an average reduction in the number of edges of about 45 percent. As in the previous publication [13], run times for the preprocessing were not reported by the authors.

4.3 Impact on Exact Solving

Besides already solving many instances to optimality, a salient application of reduction techniques is their use as a preprocessing routine for both heuristics and exact solvers. In the following we demonstrate the impact of integrating our reduction package into the Steiner problem solver SCIP-Jack. To this end, SCIP-Jack is run first without using any reduction methods and second with the entire reduction package introduced in this publication. However, the second run still includes reduction methods implicitly, namely within the ascend-and-prune heuristic. It should be noted that we do not run the reduction methods extensively, but use the termination criterion described in Section 2.4. In this way, we do not only improve the empirical run time of the entire solving procedure, but furthermore obtain a theoretical guarantee for the run time of the reduction package to be polynomial.

Table 6 provides results on the influence of the heretofore introduced (R)PCSTP reduction techniques on SCIP-Jack. For JMP and Cologne1—the two tests that require the least run time both with and without presolving being applied—the benefit of the reduction techniques is the least pronounced. For the JMP class the average run time without applying reductions is more than three times as high and for Cologne1 the increase is more than two times. When the computational difficulty increases, also the disparity between solving with and without applying reduction techniques becomes more distinctive: For CRR the average difference is close to a factor of eight. The hardest instance requires more than 300 seconds without employing reduction techniques, while it is solved in less than one second when the reduction methods described in this paper are used. Similarly, Cologne2 takes more than ten times longer to be solved when the reduction package is disabled. The last and hardest test class PUCNU manifests a different picture: Notably, the reduction techniques are far less effective, as compared to the other test classes. Nevertheless, two more instances can be solved when our reduction package is applied, compared to the computations that do not use reduction methods, and moreover each solved problem requires less than 500 seconds.

Bolstered by our reduction package, the performance of SCIP-Jack as compared to the best results obtained in the DIMACS Challenge is also remarkable. Using the same computational setting, we can solve the RPCSTP instances on average (with respect to the shifted the geometric mean) more than 50 times faster than the best solver in the DIMACS Challenge—which was also SCIP-Jack, albeit in a previous version. Likewise, for all but one of the JMP, CRR and

		With Reductions		Without Reductions	
Class	Instances	Solved	∅ Time [s]	Solved	∅ Time [s]
JMP	34	34	0.0	34	0.3
CRR	80	80	0.2	80	1.74
PUCNU	18	10	432.6	8	937.9
Cologne1	14	14	0.1	14	0.3
Cologne2	15	15	0.3	15	5.0

Table 6: Computational results of SCIP-Jack on the (R)PCSTP instances

PUCNU instances that were part of the Challenge, SCIP-Jack yields the best results.

For the MWCSP the situation is similar. In Table 7 we list the results of running SCIP-Jack with and without applying reduction techniques. The difference is tremendous: The average factor for both ACTMOD and JMPALMK instances is more than ten, with one instance—drosophila0075—taking 0.6 seconds to be solved with and 350 seconds to be solved without reduction techniques. For the SHINY instances the average increase in run time corresponds to a factor of more than four. In particular, the 25e814a792c4 instance that could be solved for the first time to optimality, see Section 4.2, cannot be solved when the reduction techniques are disabled. However, the final gap at the time limit (of two hours) is small, 0.01 percent.

		With Reductions		Without Reductions	
Class	Instances	Solved	∅ Time [s]	Solved	∅ Time [s]
JMPALMK	72	72	0.0	72	0.2
ACTMOD	8	8	0.4	8	9.0
SHINY	39	39	0.3	38	1.4

Table 7: Computational results of SCIP-Jack on the MWCSP instances

One can achieve further insight into the impact of the reduction methods introduced in this publication on exact solving by setting the heretofore computational results against the backdrop of the best results obtained at the DIMACS Challenge. When running SCIP-Jack together with the new reduction techniques in the computational environment of the DIMACS Challenge all eight ACTMOD instances are solved at least three times faster by SCIP-Jack than by any group competing in the Challenge [17]. In contrast, without using reduction techniques, SCIP-Jack is faster than all competing solvers on only one instance. Moreover, on the 20 JMPALMK instances that were part of the actual competition, SCIP-Jack using reduction techniques exceeds all participants of the Challenge on 15 and is outperformed on only one—the run time on the remaining four instances is equal to the respective best run time achieved at the Challenge.

Finally, a possible further approach for exact solving is to use the information obtained by DA that a vertex v_i of a PCSTP instance is part of at least one optimal solution. Thereupon, the PCSTP could be solved as an RPCSTP with v_i as the root. In this case, Transformation 1, which empirically yields strong results [32], could be used to eventually solve the problem as an SAP.

5 Conclusions and Outlook

In this paper we have established a reduction package for the maximum-weight connected subgraph problem, the prize-collecting Steiner tree problem, and the rooted prize-collecting Steiner tree problem that surpasses existing approaches for all three problems. The practical implications of these developments are twofold: First, 257 out of 280 benchmark instances can be solved solely by reduction techniques. Second, if our preprocessing is incorporated into an exact solver, the time required to solve instances to optimality is drastically reduced—by more than ten times on average and by more than a thousand times for several hard instances. Furthermore, besides contributing to the existing literature with various new reduction techniques, we are able to use these new methods to extend the powerful heuristics prune and ascend-and-prune from the Steiner tree problem in graphs to the RPCSTP, the PCSTP and the MWCSP.

The computational results demonstrate that in order to solve PCSTP, RPCSTP and MWCSP instances in short time it is certainly worthwhile to devise powerful reduction techniques. Additional developments of such techniques are very likely to further improve the performance of an underlying exact solver. Furthermore, one might extend the scope and set about developing reduction techniques for additional Steiner problem variants. Another promising approach would be to extend the reduction-based slack-and-prune heuristic [30], which has remained until today the empirically strongest primal SPG heuristic (perhaps with the exception of [29]). Based upon the reduction methods introduced in this paper one could use this approach for the MWCSP and PCSTP both as a stand-alone heuristic and as part of an exact solving approach.

Finally, we hope that by incorporating our reduction techniques into the academic Steiner tree problem solver SCIP-Jack not only existing problems can be solved significantly faster, but also an incentive is provided for researchers to model further real-world phenomena as an MWCSP or (R)PCSTP.

6 Acknowledgements

This work was supported by the BMWi project *Realisierung von Beschleunigungsstrategien der anwendungsorientierten Mathematik und Informatik für optimierende Energiesystemmodelle - BEAM-ME* (fund number 03ET4023DE). This work was supported by DFG in the framework of the Collaborative Research Centre CRC/Transregio 154, *Mathematical Modelling, Simulation and Optimization Using the Example of Gas Networks*. This work was supported by the ICT COST Action TD1207 *Mathematical Optimization in the Decision Support Systems for Efficient and Robust Energy Networks*. The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

References

[1] 11th DIMACS Challenge. `http://dimacs11.zib.de/`. Accessed: June 5. 2016.

[2] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[3] 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.

[4] Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel. The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization*, pages 245–270. Springer Berlin Heidelberg, 2013.

[5] C. Backes, A. Rurainski, G. Klau, O. Müller, D. Stöckel, A. Gerasch, J. Küntzer, D. Maisel, N. Ludwig, M. Hein, A. Keller, H. Burtscher, M. Kaufmann, E. Meese, and H.-P. Lenhof. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Res*, 40(6):e43, 2011.

[6] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Program.*, 59:413–420, 1993.

[7] S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 2001.

[8] Chao-Yeh Chen and Kristen Grauman. Efficient activity detection with max-subgraph search. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1274–1281, 2012.

[9] Bistra Dilkina and Carla P. Gomes. Solving connected subgraph problems in wildlife conservation. In *Proceedings of the 7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, CPAIOR'10, pages 102–116, Berlin, Heidelberg, 2010. Springer-Verlag.

[10] 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.

[11] C. Duin. *Steiner Problems in Graphs*. PhD thesis, University of Amsterdam, 1993.

[12] C.W. Duin and A. Volgenant. An edge elimination test for the Steiner problem in graphs. *Operations Research Letters*, 8(2):79 – 83, 1989.

[13] Mohammed El-Kebir and Gunnar W. Klau. Solving the Maximum-Weight Connected Subgraph Problem to Optimality. *Computing Research Repository*, abs/1409.5308, 2014.

[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. *Mathematical Programming Computations*, 2015.

[15] 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.

[16] Gerald Gamrath, Tobias Fischer, Tristan Gally, Ambros M. Gleixner, Gregor Hendel, Thorsten Koch, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Stefan Vigerske, Dieter Weninger, Michael Winkler, Jonas T. Witt, and Jakob Witzig. The scip optimization suite 3.2. Technical Report 15-60, ZIB, Takustr.7, 14195 Berlin, 2016.

[17] Gerald Gamrath, Thorsten Koch, Stephen J. Maher, Daniel Rehfeldt, and Yuji Shinano. SCIP-Jack – A solver for STP and variants with parallelization extensions. Technical Report 16-41, ZIB, Takustr.7, 14195 Berlin, 2016.

[18] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem.* Annals of Discrete Mathematics. Elsevier Science, 1992.

[19] Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. In *ISMB*, pages 233–240, 2002.

[20] David S. Johnson. The np-completeness column: An ongoing guide. *J. Algorithms*, 6(1):145–159, 1985.

[21] 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.

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

[23] 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.

[24] 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.

[25] Ivana Ljubić. *Exact and Memetic Algorithms for Two Network Design Problems.* PhD thesis, Vienna University of Technology, 2004.

[26] Alexander A. Loboda, Maxim N. Artyomov, and Alexey A. Sergushichev. Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. *CoRR, accepted for Workshop on Algorithms in Bioinformatics 2016*, abs/1605.02168, 2016.

[27] 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.

[28] Kurt Mehlhorn. A Faster Approximation Algorithm for the Steiner Problem in Graphs. *Information Processing Letters*, 27(3):125–128, 1988.

[29] Thomas Pajor, Eduardo Uchoa, and Renato F. Werneck. A Robust and Scalable Algorithm for the Steiner Problem in Graphs. *Computing Research Repository*, 2014.

[30] Tobias Polzin. *Algorithms for the Steiner problem in networks.* PhD thesis, Saarland University, 2004.

[31] Hans-Jürgen Prömel and Angelika Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity.* Advanced Lectures in Mathematics. Vieweg, 2002.

[32] Daniel Rehfeldt. A generic approach to solving the Steiner tree problem and variants. Master's thesis, Technische Universität Berlin, 2015.

[33] Daniel Rehfeldt and Thorsten Koch. Transformations for the prize-collecting Steiner tree problem and the maximum-weight connected subgraph problem to SAP. Technical Report 16-36, ZIB, Takustr.7, 14195 Berlin, 2016.

[34] Eduardo Uchoa. Reduction Tests for the Prize-collecting Steiner Problem. *Oper. Res. Lett.*, 34(4):437–444, July 2006.

[35] Siavash Vahdati-Daneshmand. *Algorithmic approaches to the Steiner problem in networks.* PhD thesis, University of Mannheim, 2004.

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

A Detailed Computational Comparisons

This section presents detailed instance-wise comparisons of our reduction package with results from the literature. The tables provide in columns two to four the number of vertices, edges, and terminals of the instance specified in the first column. The next two columns state the number of vertices and edges that remain after the reduction package described in this publication has been applied. The final two columns show the corresponding information for the reduction package described in [34].

	Original			Reduced		Reduced [34]	
Name	Vertices	Edges	Terminals	Vertices	Edges	Vertices	Edges
C01-A	500	625	5	1	0	105	190
C01-B	500	625	5	1	0	49	77
C02-A	500	625	10	1	0	82	148
C02-B	500	625	10	1	0	71	125
C03-A	500	625	83	1	0	113	190
C03-B	500	625	83	1	0	79	121
C04-A	500	625	125	1	0	72	119
C04-B	500	625	125	1	0	71	113
C05-A	500	625	250	1	0	7	9
C05-B	500	625	250	1	0	1	0
C06-A	500	1000	5	1	0	346	792
C06-B	500	1000	5	1	0	344	778
C07-A	500	1000	10	1	0	353	806
C07-B	500	1000	10	1	0	342	769
C08-A	500	1000	83	1	0	251	531
C08-B	500	1000	83	1	0	217	410
C09-A	500	1000	125	1	0	279	577
C09-B	500	1000	125	1	0	232	440
C10-A	500	1000	250	1	0	103	166
C10-B	500	1000	250	1	0	100	156
C11-A	500	2500	5	1	0	485	1801
C11-B	500	2500	5	1	0	480	1667
C12-A	500	2500	10	1	0	453	1495
C12-B	500	2500	10	1	0	441	1358
C13-A	500	2500	83	1	0	343	799
C13-B	500	2500	83	1	0	317	704
C14-A	500	2500	125	1	0	190	365
C14-B	500	2500	125	1	0	179	330
C15-A	500	2500	250	1	0	1	0
C15-B	500	2500	250	1	0	1	0
C16-A	500	12500	5	1	0	499	2714
C16-B	500	12500	5	1	0	499	2714
C17-A	500	12500	10	1	0	494	2295
C17-B	500	12500	10	1	0	494	2295
C18-A	500	12500	83	1	0	374	1002
C18-B	500	12500	83	1	0	374	997
C19-A	500	12500	125	222	567	246	589
C19-B	500	12500	125	329	971	249	592
C20-A	500	12500	250	1	0	1	0
C20-B	500	12500	250	1	0	1	0

Table 8: Reductions on CRR (C) instances.

	Original			Reduced		Reduced [34]	
Name	Vertices	Edges	Terminals	Vertices	Edges	Vertices	Edges
D01-A	1000	1250	5	1	0	223	422
D01-B	1000	1250	5	1	0	223	416
D02-A	1000	1250	10	1	0	238	450
D02-B	1000	1250	10	1	0	232	423
D03-A	1000	1250	167	1	0	114	194
D03-B	1000	1250	167	1	0	129	202
D04-A	1000	1250	250	1	0	171	297
D04-B	1000	1250	250	1	0	50	70
D05-A	1000	1250	500	1	0	84	125
D05-B	1000	1250	500	1	0	12	17
D06-A	1000	2000	5	1	0	740	1697
D06-B	1000	2000	5	1	0	736	1682
D07-A	1000	2000	10	1	0	721	1664
D07-B	1000	2000	10	1	0	702	1602
D08-A	1000	2000	167	1	0	673	1489
D08-B	1000	2000	167	1	0	561	1142
D09-A	1000	2000	250	1	0	580	1260
D09-B	1000	2000	250	1	0	439	841
D10-A	1000	2000	500	1	0	235	425
D10-B	1000	2000	500	1	0	36	56
D11-A	1000	5000	5	1	0	977	3971
D11-B	1000	5000	5	1	0	972	3740
D12-A	1000	5000	10	1	0	960	3300
D12-B	1000	5000	10	1	0	942	3040
D13-A	1000	5000	167	1	0	708	1713
D13-B	1000	5000	167	1	0	694	1631
D14-A	1000	5000	250	1	0	571	1238
D14-B	1000	5000	250	1	0	512	1062
D15-A	1000	5000	500	1	0	139	217
D15-B	1000	5000	500	1	0	4	5
D16-A	1000	25000	5	1	0	1000	6735
D16-B	1000	25000	5	1	0	1000	6725
D17-A	1000	25000	10	1	0	999	6330
D17-B	1000	25000	10	1	0	999	6330
D18-A	1000	25000	167	332	812	812	2314
D18-B	1000	25000	167	470	1278	806	2276
D19-A	1000	25000	250	407	1007	686	1895
D19-B	1000	25000	250	314	712	680	1870
D20-A	1000	25000	500	1	0	1	0
D20-B	1000	25000	500	1	0	1	0

Table 9: Reductions on CRR (D) instances.