Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

DANIEL REHFELDT, THORSTEN KOCH

Combining NP-Hard Reduction Techniques and Strong Heuristics in an Exact Algorithm for the Maximum-Weight Connected Subgraph Problem

ZIB Report 17-45 (July 2017, revised version July 2018)

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

Combining NP-Hard Reduction Techniques and Strong Heuristics in an Exact Algorithm for the Maximum-Weight Connected Subgraph Problem

Daniel Rehfeldt* · Thorsten Koch

Abstract

Borne out of a surprising variety of practical applications, the maximum-weight connected subgraph problem has attracted considerable interest during the past years. This interest has not only led to notable research on theoretical properties, but has also brought about several (exact) solvers—with steadily increasing performance. Continuing along this path, the following article introduces several new algorithms such as reduction techniques and heuristics and describes their integration into an exact solver. Based on the presented new algorithms and a new formulation our solver is able to outperform previous methods by two orders of magnitude on average. Moreover, one large-scale benchmark instance from the 11th DIMACS Challenge can be solved for the first time to optimality and the primal-dual gap for two other ones can be significantly reduced. Although this article is set against the backdrop of improved practical solving, theoretical properties (such as $\mathcal{NP}$-hardness) of the algorithmic components will receive considerable attention.

1 Introduction

The past five years have witnessed a surge of research articles dealing with the *maximum-weight connected subgraph problem* (MWCSP). As practitioners, for instance in computational biology, have become more aware of this problem and its practical potential, their work has in turn (re-)fueled the interest of mathematicians and computer scientists. The source of this symbiotic interplay is a surprisingly plain looking problem: Given an undirected graph $G = (V, E)$ and vertex weights $p : V \to \mathbb{Q}$, the task is to find a connected subgraph $S = (V(S), E[S]) \subseteq G$ such that $\sum_{v \in V(S)} p(v)$ is maximized. While computational biology [3, 12, 22] seems to be the predominant application field for the MWCSP, one also encounters the problem in other, disparate, areas such as wildlife conservation [11] and computer vision [9].

The MWCSP has been discussed in various publications, see e.g. [5, 12, 19]. Several articles have contributed to a solid theoretical understanding of the problem [8, 33] while others have addressed exact solving approaches [7, 16, 17]. Recently, also the scope of preprocessing techniques has been considerably enhanced [29].

This article aims at further enhancing the state of the art in exact solving by combining several approaches: The introduction and analysis of new reduction techniques in Section 2. The use of an integer programming (IP) formulation based on a transformation of the (rooted) MWCSP to the directed Steiner tree problem in Section 3. And finally the development of several empirically strong primal heuristics in Section 5. This section will also discuss the incorporation of the new

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

techniques into an exact MWCSP solver and furthermore analyze its performance. The combined exact algorithm will be embedded in the Steiner tree problem framework SCIP-Jack [17, 23].

1.1 Preliminaries and Notation

Throughout this article it will be assumed that in each MWCSP at least one vertex is assigned a negative and one a positive weight. In the case of only non-negative vertex weights, the MWCSP 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. Moreover, in the remainder of this article it will usually be presupposed that an MWCSP $P_{MW} = (V, E, p)$ is given such that the underlying graph (V, E) is connected. The latter assumption does not limit the generality, as one can optimize each connected components of a non-connected MWCSP separately.

As to notation, to denote the vertices and edges of a specific graph G we will write $V(G)$ and $E(G)$, respectively. In contrast, for a subset of vertices $W \subseteq V$ we define

$$E[W] := \big\{\{v_i, v_j\} \in E \mid v_i, v_j \in W\big\}.$$

Given an MWCSP, the weight of a subgraph $S = (V(S), E(S))$ will be denoted by $P(S) := \sum_{v \in V(S)} p(v)$. Further, the notation $n := |V|$ and $m := |E|$ will be used. Moreover, we define $T := \{v \in V \mid p(v) > 0\}$, set $s := |T|$, and write for the sake of simplicity $V := \{v_1, ..., v_n\}$ as well as $T := \{t_1, ..., t_s\}$.

Paths will be considered as subgraphs, and the subpath of a path Q between two vertices $v_i, v_j \in V(Q)$ will be denoted by $Q(v_i, v_j)$. A path between two vertices v_i, v_j will be referred to as (v_i, v_j)-path. This article introduces for an MWCSP instance (V, E, p) the distance function $\overline{d} : V \times V \mapsto \mathbb{Q} \cup \{-\infty\}$ defined as

$$\overline{d}(v_i, v_j) := \sup\{P(Q) \mid Q \text{ is a } (v_i, v_j)\text{-path and } (V(Q) \setminus \{v_i, v_j\}) \cap T = \emptyset\}$$

for any $v_i, v_j \in V$. In particular, $\overline{d}(v_i, v_j) = \overline{d}(v_j, v_i)$ and $\overline{d}(v_i, v_i) = p(v_i)$. Also, with the convention $\sup \emptyset = -\infty$, one observes that $\overline{d}(v_i, v_j) = -\infty$ if and only if there is no path between v_i and v_j without intermediary positive vertices. Given a vertex v_0 and two additional vertices $v_i, v_j \in V \setminus \{v_0\}$, it will be said that for v_0 vertex v_i is *$\overline{d}$-nearer* than vertex v_j if $\overline{d}(v_0, v_i) \geq \overline{d}(v_0, v_j)$. For each vertex v_i the k $\overline{d}$-nearest vertices of positive weight (if existent) are denoted by $\overline{v}_{i,1}, \overline{v}_{i,2}, ..., \overline{v}_{i,k}$. In [13] a similar distance function is defined for the Steiner tree problem in graphs that looks for paths of minimum edge weight without intermediary terminals.

For any function $x : M \mapsto \mathbb{Q}$ with M finite, and any $M' \subseteq M$ define $x(M') := \sum_{i \in M'} x(i)$. Additionally, for $W \subseteq V$ define $\delta(W) := \{\{u, v\} \in E \mid u \in W, v \in V \setminus W\}$ and for a subgraph $G' \subseteq G$ and $W' \subseteq V(G')$ define $\delta_{G'}(W') := \{\{u, v\} \in E(G') \mid u \in W', v \in V(G') \setminus W'\}$. A corresponding notation is used for directed graphs (V, A): For $W \subseteq V$ define $\delta^+(W) := \{(u, v) \in A \mid u \in W, v \in V \setminus W\}$ and $\delta^-(W) := \delta^+(V \setminus W)$.

2 Reduction Techniques

Reduction techniques for the MWCSP have not been widely studied in the literature. This characteristic sets the MWCSP in stark contrast to kinsmen such as the Steiner tree problem in graphs for which a plethora of research articles has addressed preprocessing, see e.g. [13, 27]. For the MWCSP the first ground was broken in the course of the 11th DIMACS Challenge, with two articles [4, 15] containing reduction techniques as part of an exact solving approach. Two

years later a significantly more comprehensive reduction package for the MWCSP was introduced in [29] and a dual-ascent based branch-and-bound algorithm with strong reduction properties was described in [21]. The reductions techniques introduced in the following continue the work started in [29] by introducing both bound-based and alternative-based reduction methods. To render proof techniques more perspicuous, throughout this section it will without loss of generality be assumed that each solution to P_{MW} is given as a tree (and not as an arbitrary connected subgraph).

2.1 Bound-Based Reductions

The term *bound-based reductions* describes preprocessing methods that identify edges and vertices for elimination by examining whether they induce an upper bound that is lower than a given lower bound (or vice versa) [26, 29]. In the following we will introduce a new reduction technique that could also be used to generalize the bound-based Voronoi reduction concept for the Steiner tree problem in graphs [26] and the prize-collecting Steiner tree problem [29]—in this way the proof technique for the followings three propositions is similar to the Voronoi reduction proofs in [26] and [29] and is therefore presented in the appendix only.

The base of the reduction technique is the following, new, concept: a *positive-vertex decomposition* of P_{MW}—with underlying graph (V, E)—is a partition $H = \{H_{t_i} \subseteq V \mid T \cap H_{t_i} = \{t_i\}\}$ of V such that for each $t_i \in T$ the subgraph $(H_{t_i}, E[H_{t_i}])$ is connected. Each of the H_{t_i} is called *region* with *center* t_i. Furthermore, a vertex $v_j \in H_{t_i}$ adjacent to a vertex $v_k \notin H_{t_i}$ is called *boundary vertex* of region H_{t_i}; the set of all such vertices to a region H_{t_i} will be denoted by $B(H_{t_i})$. Additionally, an edge $\{v_i, v_j\}$ with v_i and v_j in different regions will be called *H-boundary edge*.

To set the stage for the computation of an upper bound, define for all $t_i \in T$ the *positive-vertex decomposition radii*:

$$r_H(t_i) := \max\{\overline{d}(t_i, v_k) \mid v_k \in B(H_{t_i})\} \tag{1}$$

and

$$r_H^+(t_i) := \max\{r_H(t_i), 0\}. \tag{2}$$

Definition (2) allows to establish three bound-based reduction criteria presented in the following. An important observation underlying all these criteria is that for each positive vertex t_i that is part of an optimal solution S with $|V(S) \cap T| \geq 2$ there needs to be a path in $V(S) \cap H_{t_i}$ from t_i to a vertex in $B(H_{t_i})$—and the weight of this path is bounded by $r_H(t_i)$. Since t_i does not have to be in $V(S)$, one cannot use $r_H(t_i)$ to obtain a bound on the weight of S; however, one can use $r_H^+(t_i)$ instead. Moreover, one can observe that if a negative vertex v_i is part of an optimal solution, there need to be two paths in S connecting v_i to positive vertices and having no vertices but v_i in common. These two observations lead to:

Proposition 1. *Let H be a positive-vertex decomposition of P_{MW} and assume that $|T| \geq 2$. Furthermore, let $v_i \in V \setminus T$ and assume that for each optimal solution S to P_{MW} it holds that $v_i \in V(S)$. Finally, let*

$$U_2 := \sum_{t \in T} r_H^+(t) - \min\{r_H^+(t) + r_H^+(t') \mid t, t' \in T,\ t \neq t'\}. \tag{3}$$

Thereupon,

$$U := U_2 + \overline{d}(v_i, \overline{v}_{i,1}) + \overline{d}(v_i, \overline{v}_{i,2}) - p(v_i) \tag{4}$$

is an upper bound on the weight of S.

A proof can be found in Appendix A.1. It follows from the proposition that vertex of non-positive weight can be eliminated if the associated upper bound U in (4) is smaller than a known lower bound (e.g. the weight of a given solution). An application of the proposition is exemplified in Figure 1 for a simple MWCSP instance. Denote the upper left positive vertex by t_1, the lower left by t_2 and the lower right by t_3. With H being the positive-vertex decomposition as marked by the dotted ellipses it holds that $r_H^+(t_1) = 1$, $r_H^+(t_2) = 2$, and $r_H^+(t_3) = 0.5$. Consequently, for Proposition 1 with v_i being the (upper right) filled vertex it holds that $U_2 = 2$ and $U = 2.5$. Therefore, v_i can be eliminated if a lower bound higher than 2.5 is given.

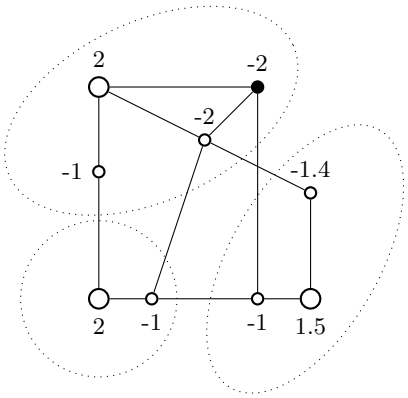

Figure 1: A positive-vertex decomposition of an MWCSP instance with regions marked by dotted ellipses.

The following proposition can be used to moreover eliminate vertices of positive weight. It can be proven similarly to Proposition 1 (see Appendix A.2).

Proposition 2. *Let H be a positive-vertex decomposition of P_{MW} and assume that $|T| \geq 2$. Furthermore, let $v_i \in T$ and assume that an optimal solution S exists such that $v_i \in V(S)$ and $|V(S) \cap T| \geq 2$. Define*

$$U_1 := \sum_{t \in T \setminus \{v_i\}} r_H^+(t) - \min\{r_H^+(t) \mid t \in T \setminus \{v_i\}\}. \tag{5}$$

Then

$$U := U_1 + \overline{d}(v_i, \overline{v}_{i,1}) \tag{6}$$

is an upper bound on the weight of S.

The positive vertex decomposition concept does not only allow for direct elimination of vertices, but can furthermore be used for a criterion that guarantees that a vertex cannot be of degree higher than 2 in any optimal solution. This information will be utilized in Section 2.2.

Proposition 3. *Let H be a positive-vertex decomposition of P_{MW} and assume that $|T| \geq 3$. Furthermore, let $v_i \in V \setminus T$ and assume that for an optimal solution S to P_{MW} it holds that $|\delta_S(v_i)| \geq 3$. Finally, let*

$$\begin{aligned} U_3 := & \sum_{t \in T} r_H^+(t) \\ & - \min\{r_H^+(t_j) + r_H^+(t_k) + r_H^+(t_l) \mid t_j, t_k, t_l \in T;\ t_j, t_k, t_l \text{ disjoint}\}. \end{aligned} \tag{7}$$

Thereupon,

$$U := U_3 + \overline{d}(v_i, \overline{v}_{i,1}) + \overline{d}(v_i, \overline{v}_{i,2}) + \overline{d}(v_i, \overline{v}_{i,3}) - 2p(v_i) \tag{8}$$

is an upper bound on the weight of S.

To efficiently apply Proposition 1, one would like to minimize (3)—and for Proposition 2 and Proposition 3 to minimize (5) and (7), respectively. Unfortunately, this problem turns out to be $\mathcal{NP}$-hard.

The decision variant of the problem can be stated as follows. Let $\alpha \in \mathbb{N}_0$ and let $G_0 = (V_0, E_0)$ be an undirected, non-empty graph. Furthermore, let $p_0 : V_0 \to \mathbb{Z}$, set $T_0 := \{v \in V_0 \mid p(v) > 0\}$, and assume that $\alpha < |T_0|$. For each positive-vertex decomposition H_0 of G_0 choose $T_0' \subsetneq T_0$ such that $|T_0'| = \alpha$ and $r^+_{H_0}(t') \leq r^+_{H_0}(t)$ for all $t' \in T_0'$ and $t \in T_0 \setminus T_0'$. Let:

$$C_{H_0} := \sum_{t \in T_0 \setminus T_0'} r^+_{H_0}(t). \tag{9}$$

We now define the α-positive-vertex decomposition problem as follows: Given a $k \in \mathbb{N}$, is there a positive-vertex decomposition H_0 such that $C_{H_0} \leq k$? In the following proposition it is shown that this problem is $\mathcal{NP}$-complete, which forthwith establishes the $\mathcal{NP}$-hardness of finding a positive-vertex decomposition that minimizes (3), (5), or (7)—which corresponds to $\alpha = 2$, $\alpha = 1$, and $\alpha = 3$, respectively.

Proposition 4. *For each $\alpha \in \mathbb{N}_0$ the α-positive-vertex decomposition problem is $\mathcal{NP}$-complete.*

Proof. Given a positive-vertex decomposition H_0 it can be tested in polynomial time whether the associated C_{H_0} is less than or equal to k. This can be be done for instance as follows: Consider the set of (directed) arcs $A' := \{(v, w) \in V_0 \times V_0 \mid \{v, w\} \in E\}$ and define edge costs $c' : A' \to \mathbb{Z}_{\geq 0}$ such that for $a = (v_i, v_j) \in A'$:

$$c'(a) = \begin{cases} -p_0(v_j), & \text{if } p_0(v_j) < 0 \\ 0, & \text{otherwise} \end{cases}$$

Thereupon, C_{H_0} can be computed by running (the directed version of) Dijkstra's algorithm for each subgraph $(H_{t_i}, A'[H_{t_i}])$, starting from t_i and using the arcs costs c'. Consequently, the positive-vertex decomposition problem is in $\mathcal{NP}$.

Next, it will be shown that the ($\mathcal{NP}$-complete [18]) independent set problem can be reduced to the positive-vertex decomposition problem. To this end, let $G_{ind} = (V_{ind}, E_{ind})$ be an undirected, non-empty graph and $k \in \mathbb{N}$. The problem is to determine whether an independent set in G_{ind} of cardinality at least k exists. Without loss of generality it will be assumed that G_{ind} does not include any vertices of degree 0.

To establish the reduction, construct a graph G_0 from G_{ind} as follows. Initially, set $G_0 = (V_0, E_0) := G_{ind}$ and define vertex weights $p_0(v_i) := 1$ for all $v_i \in V_0$. Next, extend G_0 by replacing each edge $e_l = \{v_i, v_j\} \in E_0$ with a vertex v_l' of weight $p_0(v_l') = -1$ and the two edges $\{v_i, v_l'\}$ and $\{v_j, v_l'\}$. Finally, if $\alpha > 0$, choose an arbitrary $v_0 \in V_0 \cap V_{ind}$ and add vertices v_j' of weight -1 for $j = 0, ..., \alpha$ and vertices v_j'' of weight 1 for $j = 1, ..., \alpha$ to G_0. Additionally, add an edge $\{v_0, v_0'\}$. Finally, add edges $\{v_0', v_j'\}$ and $\{v_j', v_j''\}$ for $j = 1, ..., \alpha$.

First, one observes that the size $|V_0| + |E_0|$ of the new graph G_0 is a polynomial in the size $|V_{ind}| + |E_{ind}|$ of G_{ind}. Next, $r^+_{H_0}(v_i) = 0$ holds for a vertex $v_i \in G_0 \cap G_{ind}$ if and only if H_{v_i} contains all (newly inserted) adjacent vertices of v_i in G_0. The latter condition implies that for each adjacent vertex v_k of v_i in $G_0 \cap G_{ind}$ it holds that $r^+_{H_0}(v_k) = 1$. Moreover, in a positive-vertex decomposition for (G_0, p_0) of minimum cost C_{H_0}, it holds that $r^+_{H_0}(v_j'') = 0$ for

$j = 1, ..., \alpha$. Hence, there is an independent set in G_{ind} of cardinality at least k if and only if there is a positive-vertex decomposition H_0 for (G_0, p_0) such that

$$C_{H_0} \leq |V_{ind}| - k.$$

This proves the proposition. □

Since attempting to find an exact algorithm for minimizing (3) seems to be overly optimistic, a greedy heuristic based on Dijkstra's algorithm will instead be used. Moreover, a local search heuristic has been developed to improve the decomposition found by the greedy approach. The combined algorithm runs in $O(m \log n)$, which also gives the whole bound-based reduction test a worst-case complexity of $O(m \log n)$—if a lower bound is already available. This reduction test will be referred to as **Positive Vertex Decomposition** (**PVD**) test; the computation of a lower bound will be discussed in Section 4.

2.2 Alternative-Based Reductions

This section covers several *exclusion tests* [13]: reduction methods that attempt to prove that a specified part of the problem graph—usually a single vertex or edge—is not contained in at least one optimal solution. The usual procedure is to show that for each solution that contains this specified subgraph there is another, alternative, solution of equal or better objective value that does not.

Bottleneck Distances

First, an exclusion concept introduced in [29] will be revisited. Define the *interior cost* of a subpath $Q(v_k, v_l)$ (of a path Q) in (V, E) as:

$$C^-(Q(v_k, v_l)) := \sum_{v \in V(Q(v_k, v_l)) \setminus \{v_k, v_l\}} p(v), \tag{10}$$

where the convention that the empty sum equals 0 is assumed, so the interior cost of an edge is likewise 0. Furthermore, define the *vertex weight bottleneck length* of Q as:

$$\hat{l}(Q) := \min_{v_k, v_l \in V(Q)} C^-(Q(v_k, v_l)). \tag{11}$$

Note that $\hat{l}(Q) \leq 0$ holds, because the interior cost of an edge is 0. For motivating the vertex weight bottleneck length consider the deletion of a vertex v_k of a solution (tree) S to P_{MW}. This deletion results in two trees S_1, S_2. Assume now that there is a path Q that contains vertices of both S_1 and S_2. If $\hat{l}(Q) > p(v_k)$, one can reconnect S_1 and S_2 to a solution S' that does not contain v_k and that satisfies $P(S') > P(S)$. Naturally, one would like to find a maximum lower bound. To this end, let $\mathcal{Q}(v_i, v_j)$ be the set of all paths between v_i and v_j. Thereupon, define the *vertex weight bottleneck distance* as:

$$\hat{d}(v_i, v_j) := \max\{\hat{l}(Q) \mid Q \in \mathcal{Q}(v_i, v_j)\}. \tag{12}$$

In [29] a reduction test based on the vertex weight bottleneck distance to eliminate vertices of up to degree 5 was employed. The next proposition and its subsequent corollary allow to design a complementary reduction test.

Proposition 5. *Let $v_i \in V \setminus T$ and assume that there is at least one optimal solution S to P_{MW} such that $|\delta_S(v_i)| \leq 2$. Denote by Δ the set of all vertices adjacent to v_i. If for each two, non-identical, vertices $v_j, v_k \in \Delta$ it holds that $\hat{d}(v_j, v_k) > p(v_i)$, then there is at least one optimal solution that does not contain v_i.*

Proof. Let S be a tree such that $|\delta_S(v_i)| \in \{1, 2\}$. If $|\delta_S(v_i)| = 1$, then the (possibly empty) tree S' obtained by removing v_i and its incident edge from S is of no lower weight than S. If $|\delta_S(v_i)| = 2$, denote the two vertices contained in $\delta_S(v_i)$ by v_i', v_i'' and construct a new tree S' as follows. First, set $S' = S$ and subsequently remove v_i and its incident edges from S'. Second, consider a path Q corresponding to $\hat{d}(v_i', v_i'')$. Because $\hat{d}(v_i', v_i'') > p(v_i)$ holds (by assumption), Q cannot contain v_i. Let $v_r, v_s \in V(Q) \cap V(S')$ such that $V(Q(v_r, v_s)) \cap V(S') = \{v_r, v_s\}$. Adding $Q(v_r, v_s)$ to S' one obtains a tree that does not contain v_i and is of higher weight than S. □

If the test condition of Proposition 5 is not successful for a vertex v_i, one may still be able to delete edges incident to v_i by using:

Corollary 6. *Let $v_i \in V \setminus T$ and assume that there is at least one optimal solution S to P_{MW} such that $|\delta_S(v_i)| \leq 2$. Denote by Δ the set of all vertices adjacent to v_i and let $v_j \in \Delta$. If for each vertex $v_k \in \Delta \setminus \{v_j\}$ it holds that $\hat{d}(v_j, v_k) > p(v_i)$, then there is at least one optimal solution that does not contain the edge $\{v_i, v_j\}$.*

The reduction test based on Proposition 5 and Corollary 6 will be referred to as **Extended Non-Positive Vertex** (**ENPV**) test. To find vertices that are of degree at most 2 in at least one optimal solution, the criterion described in Proposition 3 and the dual-ascent criterion mentioned in Section 3 are used. Furthermore, to limit the computation time, the ENPV test inspects only vertices of degree smaller than or equal to 6—the number which gave the best results in preliminary tests.

In both [29] and this article heuristics (with worst-case complexity of $O(1)$ [29]) are employed to compute lower bounds on the vertex weight bottleneck distance. To justify the use of heuristics, it will be demonstrated that computing the vertex weight bottleneck distance is $\mathcal{NP}$-hard—which does not come as a surprise, since also the corresponding problem for the prize-collecting Steiner tree problem is $\mathcal{NP}$-hard [32].

First, the decision variant of the vertex weight bottleneck distance is defined. Let $G_0 = (V_0, E_0)$ be an undirected and connected graph with $|V_0| \geq 2$. Furthermore, let $p_0 : V_0 \to \mathbb{Z}$. Given two distinct vertices $v_i, v_j \in V_0$ and a $k \in \mathbb{Z}_{\leq 0}$, the vertex weight bottleneck distance problem is to determine whether $\hat{d}(v_i, v_j) \geq k$. The $\mathcal{NP}$-hardness of the problem can be shown by a reduction from the Hamiltonian path problem—as in the $\mathcal{NP}$-hardness proof of the bottleneck distance test for the prize-collecting Steiner tree problem [32].

Proposition 7. *The vertex weight bottleneck distance problem is $\mathcal{NP}$-complete.*

Proof. First, note that the vertex weight bottleneck length of a given path Q can be computed in $O(|V_0(Q)|^2)$; hence, the vertex weight bottleneck distance problem is in $\mathcal{NP}$. Next, let $G_{Ham} = (V_{Ham}, E_{Ham})$ be an undirected, connected graph with two distinct vertices v_i, v_j. The Hamiltonian path problem asks whether a (simple) path between v_i and v_j exists that contains all vertices. This problem can be reduced to the vertex weight bottleneck distance problem as follows. Initially, set $G_0 := (V_0, E_0) := G_{Ham}$ and define $p_0(v_i) := 1$ for all $v_i \in V_0$. Next, extend G_0 by adding vertices v_i', v_i'' with weights $p_0(v_i') = -|V_{Ham}|,\ p_0(v_i'') = 0$ and vertices v_j', v_j'' with weights $p_0(v_j') = -|V_{Ham}|,\ p_0(v_j'') = 0$ to V_0. Finally, add edges $\{v_i, v_i'\}$, $\{v_i', v_i''\}$ and $\{v_j, v_j'\}$, $\{v_j', v_j''\}$ to E_0. Thereupon, G_{Ham} contains an Hamiltonian path between v_i and v_j if and only if $\hat{d}(v_i'', v_j'') \geq -|V_{Ham}|$ on (V_0, E_0, p_0). □

Notwithstanding its $\mathcal{NP}$-hardness, the vertex weight bottleneck distance maximization problem can be approximated by heuristics well enough to allow for a strong practical performance of the ENPV test.

Dominating Connected Sets

Besides paths, one can also use general connected subgraphs for alternative-based reductions tests. This article introduces the concept of *dominating connected sets* for the MWCSP: Let $X \subset V$ such that $(X, E[X])$ is connected and let $W \subset V \setminus X$. Then X will be said to *MWCS-dominate* W if

$$\big\{v_i \in V \setminus W \mid \exists \{v_i, v_j\} \in E, v_j \in W\big\} \subseteq \big\{v_i \in V \mid \exists \{v_i, v_j\} \in E, v_j \in X\big\} \cup X.$$

Importantly, one can remove W from any feasible solution and reconnect the resulting components by using only vertices of X. In the following, additional conditions will be formulated that allow to remove W, or parts of it, without reducing the weight of at least one optimal solution. The first such condition is stated in the following proposition, which generalizes a lemma from [29].

Proposition 8. *Let $W \subseteq V \setminus T$ and $X \subseteq V \setminus W$ such that X MWCS-dominates W and assume*

$$\sum_{w \in W} p(w) \leq \sum_{u \in X : p(u) < 0} p(u). \tag{13}$$

Then there exists an optimal solution S such that $W \nsubseteq V(S)$. The set X will be said to ***all-weights MWCS-dominate*** *W.*

Proof. Let S be a feasible solution with $W \subseteq V(S)$. Note that by construction $p(w) \leq 0$ for all $w \in W$. Define

$$\Delta_S := \big\{v \in V(S) \setminus W \mid \exists \{v, w\} \in E(S), w \in W\big\}.$$

Next, remove W from S. In this way one obtains a new (possibly empty) subgraph S' that contains at most $|\Delta_S|$ many (inclusion-wise maximal) connected components. If S' is connected, no further discussion is necessary. Otherwise, note that each connected component of S' contains a vertex $v_i \in \Delta_S$. Therefore, these components can be reconnected as follows. First, add $X \setminus V(S')$ to $V(S')$ to obtain a new subgraph S''. Second, because X MWCS-dominates W and because each connected component contains a $v_i \in \Delta_S$, there exists a set of edges $\tilde{E}_{S''} \subseteq E[V(S'')]$ that reconnects S''. Adding $\tilde{E}_{S''}$ to S'', one obtains a, finally connected, subgraph S'''. Finally, the construction of S''' implies:

$$\sum_{v \in V(S''')} p(v) \geq \sum_{v \in V(S)} p(v) - \sum_{w \in W} p(w) + \sum_{u \in X : p(u) < 0} p(u) \overset{(13)}{\geq} \sum_{v \in V(S)} p(v).$$

This concludes the proof. □

While Proposition 8 guarantees that set W is not part of at least one optimal solution, the same may not be true for subsets of W. Therefore, one cannot just eliminate W in general. However, in the case of $|W| = 1$ one can forthwith eliminate W, and in the case of $|W| = 2$ with $W = \{v_i, v_j\} \in E$ one can eliminate the edge $\{v_i, v_j\}$. Figure 2 shows an MWCSP instance for which an edge can be eliminated by means of the criterion formulated in Proposition 8. The vertices of the edge drawn in bold have a summed weight of -4.3, lower than the weight of the (sole) negative vertex in the MWCS-dominating set marked by the dotted ellipse (which is -3.5).

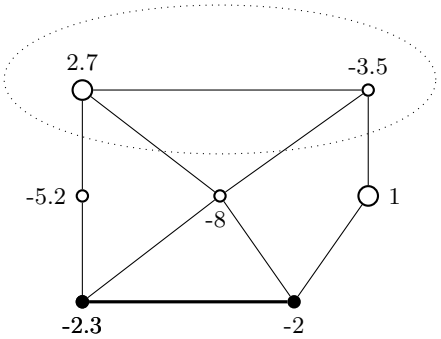

Figure 2: An MWCSP instance. Considering the vertices enclosed by the dotted ellipse as the set X, one can verify with Proposition 8 that the bold edge between the two filled vertices can be deleted.

In contrast to Proposition 8, the following proposition allows to eliminate non-trivial (i.e. larger than single-vertex or single-edge) subgraphs of $(V \setminus T, E[V \setminus T])$—but also involves a more restricting test condition.

Proposition 9. *Let $W \subseteq V \setminus T$ and $X \subseteq V \setminus W$ such that X MWCS-dominates W and assume*

$$\max_{w \in W} p(w) \leq \sum_{u \in X : p(u) < 0} p(u). \tag{14}$$

Then there exists an optimal solution S such that $W \cap V(S) = \emptyset$. The set X will be said to ***max-weight MWCS-dominate*** *W.*

Proof. Let S be a feasible solution with $W \cap V(S) \neq \emptyset$. Further, define Δ_S as in the proof of Proposition 8. Remove $W \cap V(S)$ from S to obtain a new (possibly empty) subgraph S' that contains at most $|\Delta_S|$ many (inclusion-wise maximal) connected components. Assume that there are at least two connected components. Each of these components contains a vertex $v_i \in \Delta_S$. These components can therefore be reconnected as in the proof of Proposition 8 to obtain a connected subgraph S''' with $W \cap V(S''') = \emptyset$. Because of (14) it holds for the resulting connected subgraph S''' that $\sum_{v \in V(S''')} p(v) \geq \sum_{v \in V(S)} p(v)$. □

Figure 3 shows an MWCSP instance that can be reduced by Proposition 9. Associated with Proposition 8 and Proposition 9 the subsequently sketched reduction test will be used. For each vertex $v_i \in V \setminus T$ the union of v_i and all non-negative adjacent vertices of v_i is considered as the set X in Proposition 8. Thereupon, one attempts to eliminate neighboring vertices of X by using the conditions of Proposition 8—testing for all $v_i \in V \setminus (X \cup T)$ whether they are MWCS-dominated by X proved to be too expensive. If the test fails for a neighbor v_j of X but there is only one neighbor v_k of v_j that is neither in X nor a neighbor of X, then it is checked: first, whether the vertex set $\{v_j, v_k\}$ can be eliminated by using Proposition 8 and otherwise whether the corresponding edge can be deleted by means of Proposition 9. The number and choice of neighbors is restricted such that the entire algorithm can be guaranteed to be of worst-case complexity of $O(m^2)$. However, in practice one observes a run time that is much better than this bound suggests. This test will be referred to as **Dominating Vertex Set** (**DVS**).

For the special case of $|W| = 1$ a vertex set X max-weight MWCS-dominates a vertex set W if and only if X all-weights MWCS-dominates W. Therefore, such a set will be called **single-weight MWCS-dominating**. As will be shown in the following, already this special case is $\mathcal{NP}$-hard.

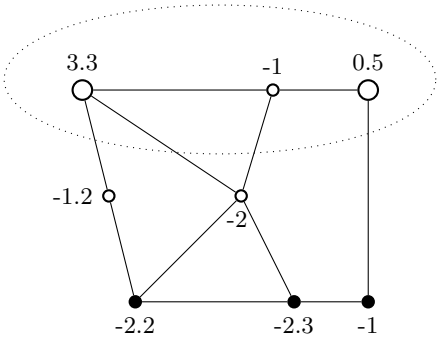

Figure 3: An MWCSP instance. Considering the vertices enclosed by the dotted ellipse as the set X, one can verify with Proposition 9 that the three filled vertices can be deleted.

Let $G_0 = (V_0, E_0)$ be an undirected, non-empty graph. Furthermore, let $p_0 : V_0 \to \mathbb{Z}$. Given a vertex $v_i \in V_0$ with $p_0(v_i) \leq 0$ the single-weight MWCS-domination problem is to determine whether a subset of $V \setminus \{v_i\}$ exists that single-weight MWCS-dominates v_i.

Proposition 10. *The single-weight MWCS-domination problem is $\mathcal{NP}$-complete.*

Proof. Given a vertex subset X it can be verified with worst-case complexity of $O(|E_0| + |V_0|)$ whether this is an MWCS-dominating set to v_i. Hence, the single-weight MWCS-dominating decision problem is in $\mathcal{NP}$.

In the following it will be demonstrated that the ($\mathcal{NP}$-complete [18]) vertex cover problem can be reduced to the single-weight MWCS-domination problem. Let $G_{cov} = (V_{cov}, E_{cov})$ be an undirected, non-empty graph and $k \in \mathbb{N}$. Thereupon, for the vertex cover problem it has to be determined whether a set in V_{cov} of cardinality at most k exists that is incident to all edges E_{cov}.

To establish the reduction, construct a graph G_0 from G_{cov} as follows: Start with $G_0 = (V_0, E_0) := G_{cov}$ and extend this graph as follows: First, define vertex weights $p_0(v_i) := -1$ for all $v_i \in V_0$. In the next step replace each edge $e_l = \{v_i, v_j\} \in E_0$ by a vertex v'_l of weight $p_0(v'_l) := -(k+1)$ and the two edges $\{v_i, v'_l\}$ and $\{v_j, v'_l\}$. Moreover, add edges $\{v_i, v_j\}$ for each pair of distinct vertices $v_i, v_j \in V_0 \cap V_{cov}$ to E_0. Due to the previous step, this procedure does not lead to multi-edges. Finally, add a vertex $v_0^\star$ of weight $p_0(v_0^\star) := -k$ to V_0 and add edges $\{v_0^\star, v_i\}$ for all $v_i \in V_0 \setminus (V_{cov} \cup \{v_0^\star\})$.

Scrutinizing the graphs G_0 and G_{cov}, one can verify that a single-weight MWCS-dominating set X to $v_0^\star$ exists if and only if to each (newly added) vertex $v_i \in V_0 \setminus (V_{cov} \cup \{v_0^\star\})$ there is an adjacent vertex $v_j \in V_0 \cap V_{cov}$ with $v_j \in X$. The latter condition is satisfied if and only if there is a vertex cover in G_{cov} of cardinality at most k. □

Combining Dominating Sets and Bottleneck Distances

Although both being $\mathcal{NP}$-hard, the MWCS-domination and the bottleneck distance concept can be merged into a powerful additional reduction test. The stage for this combined routine is set by the following:

Proposition 11. *Let $W \subseteq V \setminus T$ and define*

$$\Delta := \big\{ v_i \in V \setminus W \mid \exists \{v_i, v_j\} \in E, v_j \in W \big\}$$

If $\Delta = \emptyset$, then no optimal solution to P_{MW} contains any vertex of W. Otherwise, let $X \subseteq V \setminus W$ such that

$$\Delta_1 := \Delta \cap \left(\left\{v_i \in V \setminus X \mid \exists \{v_i, v_j\} \in E, v_j \in X\right\} \cup X\right)$$

is non-empty and $(X, E[X])$ is connected. Define

$$C_1 := \sum_{u \in X : p(u) < 0} p(u). \tag{15}$$

Further, let $\Delta_2 := \Delta \setminus \Delta_1$ and choose for each $v_k \in \Delta_2$ an, arbitrary, $v'_k \in X$. Define

$$C_2 := \sum_{v_k \in \Delta_2} \hat{d}(v_k, v'_k). \tag{16}$$

If

$$C := C_1 + C_2 > \sum_{w \in W} p(w), \tag{17}$$

then each optimal solution S to P_{MW} satisfies $W \nsubseteq V(S)$.

Proof. Let S be a feasible solution with $W \subseteq V(S)$. Note that both $C_1 \leq 0$ and $C_2 \leq 0$. Define

$$\Delta_1^S := \Delta_1 \cap V(S)$$

and

$$\Delta_2^S := \Delta_2 \cap V(S).$$

In the following it will be demonstrated how to construct a connected subgraph S''' that does not contain all vertices of W and satisfies $P(S''') \geq P(S)$.

Let S' be the subgraph obtained from S by removing W and all incident edges. Note that each maximal connected component of S' contains at least one vertex of $\Delta_1^S \cup \Delta_2^S$. Furthermore, it holds that

$$P(S') = P(S) - \sum_{w \in W} p(w). \tag{18}$$

Set $S'' := S'$. If $\Delta_1^S \neq \emptyset$, add $(X \setminus V(S''), E[X] \setminus E(S'')$ to S''. Moreover, in this case add for each $v_i \in \Delta_1^S \setminus V(S'')$ an edge $\{v_i, v_j\}$ for a $v_j \in X$ to $E(S'')$. Consequently, it holds for S'' that

$$P(S'') \geq P(S') + C_1 \overset{(18)}{=} P(S) - \sum_{w \in W} p(w) + C_1. \tag{19}$$

Moreover, all Δ_1^S are part of one connected component of S''.

Set $S''' := S''$. Consider each $v_k \in \Delta_2^S \setminus V(S''')$ consecutively and choose a (v_k, v'_k)-path Q^k (with v'_k as defined in the statement of this proposition) such that $\hat{l}(Q^k) = \hat{d}(v_k, v'_k)$. If v_k and v'_k are in different connected components of S''', there exist $v_q \in V(Q^k)$ in the connected components of v_k and $v'_q \in V(Q^k)$ in the connected component of v'_k such that $V(Q(v_q, v'_q)) \cap V(S''') = \{v_q, v'_q\}$. Add $Q(v_q, v'_q)$ to S'''. Because of condition (17) there is at least on vertex of W that is not contained in any of these newly added paths—otherwise it would hold that $C_2 \leq \sum_{w \in W} p(w)$ and therefore also $C \leq \sum_{w \in W} p(w)$. Moreover, because of condition (16) the overall procedure

reduces the weight of S'' by at most $|C_2|$. Hence, it holds for the new (now connected) subgraph S''' that

$$P(S''') \geq P(S'') + C_2 \overset{(19)}{\geq} P(S) - \sum_{w \in W} p(w) + C_1 + C_2. \tag{20}$$

Finally, $W \nsubseteq V(S''')$ holds and due to (17) it follows from (20) that

$$P(S''') > P(S). \tag{21}$$

Hence the proposition is proven. □

Corollary 12. *Assume that the conditions of Proposition 11 hold, but instead of* (17) *assume*

$$C_1 + C_2 > \max_{w \in W} p(w). \tag{22}$$

Then each optimal solution S to P_{MW} satisfies $W \cap V(S) = \emptyset$.

Proof. Let S be a feasible solution. Further, let S''' be a connected subgraph created from S by the procedure described in the proof of Proposition 11. S''' is connected and it holds that $P(S''') > P(S)$, so only the equation $W \cap V(S''') = \emptyset$ needs to be verified. By construction all vertices of S''' are in one of the three sets: $(V(S) \setminus W)$, X, and the set of vertices that are part of a (v_k, v'_k)-path Q^k with $\hat{l}(Q^k) = \hat{d}(v_k, v'_k)$ and $v_k \in \Delta_2^S$. By definition the first two of these sets cannot contain any vertices of W. Furthermore, because of (22), none of the paths Q^k can contain a vertex of W since otherwise it would hold that $\hat{l}(Q^k) \leq \max_{w \in W} p(w)$—which is a contradiction because of $C_1 + C_2 \leq C_2 \leq \hat{l}(Q^k)$. Thus, $W \cap V(S) = \emptyset$. □

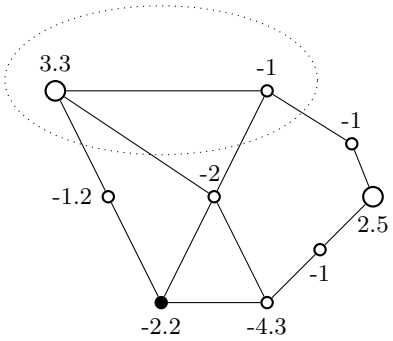

Figure 4: An MWCSP instance. Considering the vertices enclosed by the dotted ellipse as the set X, one can verify with Proposition 11 that the (bottom left) filled vertex can be deleted.

Once again, for the special case of $|W| = 1$, corollary and proposition coincide. Figure 4 shows an MWCSP for which a vertex can be deleted by means of this special case. Consider the two encircled vertices as the set X. The right neighbor of the filled vertex can be connected by a path of bottleneck length -1 to X, so $C_2 \geq -1$. Since $C_1 = -1$ and all other neighbors of the filled vertex are also neighbors of X, one can delete the vertex.

For Proposition 11 and Corollary 12 a reduction test similar to DVS is used. It will be referred to as **Alternative Vertex Set** (**AVS**) test. As in the ENPV test, only lower bounds for the bottleneck distance $\hat{d}$ are computed for AVS.

2.3 Using Reduction Techniques for Preprocessing and Propagation

Within the exact solver described in this article, reduction techniques can be found in three components: In preprocessing, in propagation during branch-and-bound, and in the heuristics. Only the former two applications will be discussed in the following, with the latter being described in Section 5.1.

Besides the PVD, ENPV, DVS, and AVS tests, the combined reduction package incorporates three routines described in [29]: Basic (a set of simple reductions such as contracting adjacent non-negative vertices), Non-Negative Paths (a method to delete redundant edges), and Non-Positive Vertex of Degree k (a method to delete vertices based on the bottleneck distance). All seven routines are executed iteratively within a loop that is reiterated as long as a predefined percentage of vertices has been eliminated during the previous round. The procedure will be referred to as *Reduce-Basic*. This package, is used as a domain propagation routine during branch-and-bound: Instead of deleting edges or vertices, the corresponding variables are fixed to zero in the integer programming formulation described in Section 3.

Reduce-Basic can be improved by using lower bound based reduction methods from Section 3. This combination will be referred to as *Reduce-Advanced* and is used as a preprocessing tool for exact solving. Strong preprocessing is instrumental for the practical performance of the exact solver described in this article, being able to already find optimal solutions for more than 90% of all instances. While the techniques introduced in [29] are empirically already notably successful, the combination with the new reduction methods not only enhances the strength of the overall reduction package, but also to leads to a significantly lower total execution time, as computationally expensive tests (such as dual-ascent) are applied on an already much smaller problem.

3 From Dual-Ascent to Exact Solving

Dual-ascent has recently been shown to be a powerful tool for the MWCSP, both for graph reduction and for heuristics [21, 29]. In both [29] and this article, dual-ascent is based on the *Steiner arborescence problem* (*SAP*), which is defined as follows: 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, T, c, r), one can associate with each arc $a \in A$ a variable $x(a)$ indicating whether a is contained in the Steiner arborescence ($x(a) = 1$) or not ($x(a) = 0$). Thereupon, an IP formulation can be stated as [34]:

Formulation 1. *Directed Cut Formulation*

$$\begin{aligned}
&\min \quad c^T x && && (23)\\
&x(\delta^-(W)) \;\geq\; 1 && \text{for all } W \subset V,\ r \notin W, W \cap T \neq \emptyset, && (24)\\
&x(a) \;\in\; \{0,1\} && \text{for all } a \in A. && (25)
\end{aligned}$$

In [34] a dual-ascent algorithm for Formulation 1 was introduced that, empirically, both provides strong lower bounds and allows for fast computation, defying its worst-case time complexity of $O(|A| \min\{|V||T|, |A|\})$ [25]. At termination, dual-ascent provides a dual solution to the LP-relaxation of Formulation 1, involving directed paths along arcs of reduced cost 0 from the root to each additional terminal. This information can be used to facilitate the solving process for an MWCSP as will be delineated in the following.

The first step is to transform a given MWCSP to an SAP as originally described in [28]. The underlying idea is to treat the MWCSP vertices of positive weight as terminals in the SAP. However, since all terminals need to be part of any feasible SAP solution, several vertices (including a root) and arcs are added to the SAP that allow to model not including a positive vertex in a feasible solution.

Transformation 1 (MWCSP to SAP)**.**
Input: *An MWCSP* $P_{MW} = (V, E, p)$
Output: *An SAP* $P' = (V', A', T', c', r')$

1. *Set* $V' := V$, $A' := \{(v, w) \in V' \times V' \mid \{v, w\} \in E\}$.
2. *Set* $c' : A' \to \mathbb{Q}_{\geq 0}$ *such that for* $a = (v, w) \in A'$*:*
   $c'(a) = \begin{cases} -p(w), & \text{if } p(w) < 0 \\ 0, & \text{otherwise} \end{cases}$
3. *Add two vertices* r' *and* v'_0 *to* V'.
4. *Denote the set of all* $v \in V$ *with* $p(v) > 0$ *by* $T = \{t_1, ..., t_s\}$ *and define* $M := \sum_{t \in T} p(t)$.
5. *For each* $i \in \{1, ..., s\}$*:*
   (a) *Add an arc* (r', t_i) *of weight* M *to* A'.
   (b) *Add a new node* t'_i *to* V'.
   (c) *Add arcs* (t_i, v'_0) *and* (t_i, t'_i) *to* A'*, both being of weight* 0.
   (d) *Add an arc* (v'_0, t'_i) *of weight* $p(t_i)$ *to* A'.
6. *Define the set of terminals* $T' := \{t'_1, ..., t'_s\} \cup \{r'\}$.
7. ***Return*** (V', A', T', c', r').

This transformation is utilized in [29] together with dual-ascent to eliminate both edges and vertices of an MWCSP. In the same way dual-ascent can be used to show that a vertex $v \in V \setminus T$ cannot be of degree larger than 2 in any optimal solution (which is a prerequisite for the ENPV test). In this article, Transformation 1 is additionally used to show that a vertex $v \in T$ is part of at least one optimal solution. Consider the SAP instance $P' = (V', A', T', c', r')$ obtained by applying Transformation 1 on P_{MW}. Moreover, let L_{DA} be the lower bound obtained by dual-ascent and U an upper bound for P'. If the reduced cost of an arc (r', t'_i) is higher than $U - L_{DA}$, it can be deduced that the vertex t_i is part of at least one optimal solution to P_{MW}. If at least one (positive) vertex can be shown to be part an optimal solution, the MWCSP can be solved as a *rooted maximum-weight connected subgraph problem* (RMWCSP). This problem incorporates the additional condition that a non-empty set of vertices $R \subseteq V$ needs to be part of all feasible solutions [6]. Note that in the exact branch-and-cut solver SCIP-Jack 1.0 a transformation for the MWCSP similar to Transformation 1 is used [17]. A disadvantage of both transformations is the existence of symmetric solutions in the resulting IP formulation (for a solution S, there are $|V(S) \cap T| - 1$ many). For the RMWCSP one can instead apply the following (new) transformation, which gives way to a problem that is not burdened with symmetric solutions. The transformation will be provided in a more general setting, namely for the directed variant of the RMWCSP [6]: Given a directed graph $D = (V, A)$, vertex weights $p : V \to \mathbb{Q}$, a non-empty set $R \subseteq V$ and an $r \in R$, find a connected subgraph $S = (V(S), E(S)) \subseteq D$ such that

1. $R \subseteq V(S)$,
2. each $v_i \in V(S)$ can be reached from r on a directed path in S,
3. $\sum_{v \in V(S)} p(v)$ is maximized.

One verifies that any undirected RMWCSP can be formulated in directed form by choosing an arbitrary $r \in R$ and replacing each edge by two anti-parallel arcs. In the following it will be assumed that all vertices in R have 0-weight.

Transformation 2 (Directed RMWCSP to SAP)**.**
Input: *A directed RMWCSP* $P_{RMW} = (V, A, R, p, r)$
Output: *An SAP* $P' = (V', A', T', c', r')$

1. *Set* $V' := V$, $A' := A$, $r' := r$.
2. *Set* $c' : A' \to \mathbb{Q}_{\geq 0}$ *such that for* $a = (v, w) \in A'$*:*
$$c'(a) = \begin{cases} -p(w), & \text{if } p(w) < 0 \\ 0, & \text{otherwise} \end{cases}$$
3. *Denote the set of all* $v \in V \setminus R$ *with* $p(v) > 0$ *by* $T = \{t_1, ..., t_s\}$
4. *For each* $i \in \{1, ..., s\}$*:*
   (a) *Add a new node* t'_i *to* V'.
   (b) *Add an arc* (r', t'_i) *of weight* $p(t_i)$ *to* A'.
   (c) *Add an arc* (t_i, t'_i) *of weight* 0 *to* A'.
5. *Define the set of terminals* $T' := \{t'_1, ..., t'_s\} \cup \{R\}$.
6. ***Return*** (V', A', T', c', r').

The transformation is illustrated in Figure 5. Moreover, the correspondence between a directed RMWCSP and the SAP resulting from Transformation 2 is established by the following proposition.

(a) Directed RMWCSP instance

(b) Transformed SAP instance

Figure 5: Illustration of a directed RMWCSP instance with root r (left) and the equivalent SAP obtained by Transformation 2 (right). Terminals are drawn as squares.

Proposition 13 (Directed RMWCSP to SAP)**.** *Let* $P' = (V', A', T', c', r')$ *be an SAP obtained from a directed RMWCSP* $P_{RMW} = (V, A, R, p, r)$ *by applying Transformation 2. Each solution* S' *to* P' *can be mapped to a solution* S *to* P_{RMW} *defined by:*

$$V(S) := V \cap V'(S'), \tag{26}$$
$$A(S) := A \cap A'(S') \tag{27}$$

If S' is an optimal solution to P', then S is an optimal solution to P_{RMW} and their objective values satisfy:

$$p(V(S)) = \sum_{v \in V : p(v) > 0} p(v) - c(A'(S')). \tag{28}$$

In the following, the Directed Cut Formulation for the SAP (V', A', T', c', r') obtained from an directed RMWCSP by performing Transformation 2 will be referred to as $TransCut$. Note that with the remarks prior to Transformation 2, $TransCut$ can also be used for an undirected RMWCSP and for an MWCSP for which a specific vertex that is part of at least one optimal solution is known. The objective value of a solution $x \in \{0,1\}^{|A'|}$ to $TransCut$ is defined as:

$$v(TransCut) := \sum_{v \in V : p(v) > 0} p(v) - {c'}^T x. \tag{29}$$

The SAP resulting from Transformation 2 displays two immediate advantages as compared to the one from Transformation 1. First, the number of arcs is reduced by at least $2|T|$ (with T as defined in Transformation 1). Second, while for each (LP) solution to the Directed Cut Formulation of the SAP from Transformation 1 there can be up to $|T| - 1$ equivalent solutions, this symmetry has vanished in the $TransCut$ formulation. In addition to these advantages, the new SAP can be solved by the separation algorithm of SCIP-Jack without any alterations. This algorithm is based on the SAP integer programming formulation introduced in [20]. For an SAP (V, A, T, c, r) this formulation is defined as the Directed Cut Formulation plus the constraints

$$x(\delta^-(v)) \leq x(\delta^+(v)), \qquad \text{for all } v \in V \setminus T \tag{30}$$

$$x(\delta^-(v)) \geq x(a), \qquad \text{for all } a \in \delta^+(v), v \in V \setminus T \tag{31}$$

Empirically, $TransCut$ together with (30) and (31) proves to be considerably (and consistently) stronger than the formulation used by SCIP-Jack [17]. The experiments conducted for this article revealed a speedup of more than a 200% for the main solution process on all instances for which the transformation could be applied—which was the case for more than 80 % of all test instances that were not already solved in preprocessing.

Although being able to reuse the separation algorithm of SCIP-Jack is practically highly advantageous, another important question is how $TransCut$ compares with other IP formulations for the (directed) RMWCSP. To this end, we introduce a formulation that was used by [6] in a branch-and-cut based directed RMWCSP solver, which displayed a strong performance on several (real-world) test sets. Moreover, this formulation was identified in [6] as the strongest (with respect to the LP-relaxation) of several IP formulations for the directed RMWCSP. Two definitions are required to set the stage. Let v_i and v_j be two distinct vertices in a directed graph (V, A). A subset $N \subseteq V \setminus \{v_i, v_j\}$ is called $(v_i, v_j)-$*separator* if there is no directed path from v_i to v_j in the graph $(V \setminus N, A[V \setminus N])$. The family of all $(v_i, v_j)-$separators is denoted by $\mathcal{N}(v_i, v_j)$. Furthermore, for a directed graph (V, A) and a $v_i \in V$ define $D^+(v_i) := \{v \in V \mid (v_i, v) \in A\}$.

With the node-separator concept at hand, an IP formulation for the directed RMWCSP can be stated as follows [6]:

Formulation 2. *Node Separator Formulation*

$$max \quad p^T y \tag{32}$$

$$y(N) \geq y(v) \qquad \textit{for all } v \in V \setminus (\{r\} \cup D^+(r)),\ N \in \mathcal{N}(r, v), \tag{33}$$

$$y(v) = 1 \qquad \textit{for all } v \in R, \tag{34}$$

$$y(v) \in \{0,1\} \qquad \textit{for all } v \in V. \tag{35}$$

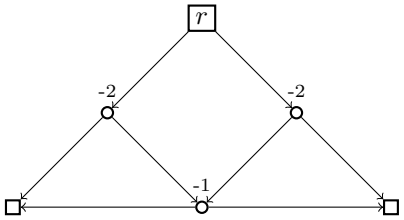

Figure 6: Illustration of a directed RMWCSP instance with vertices R drawn as squares. For this instance $v_{LP}(Cut_r) = -2.5$ and $v_{LP}(TransCut) = v(TransCut) = -3$ holds.

The Node Separator Formulation is also referred to as Cut_r in [6] and its optimal objective value will be denoted by $v(Cut_r)$.

From (28) it follows that $v(TransCut) = v(Cut_r)$. However, for the optimal LP solution values $v_{LP}(TransCut)$ of $TransCut$ and $v_{LP}(Cut_r)$ of Cut_r equality does not in general hold. The next proposition shows that $v_{LP}(TransCut) \leq v_{LP}(Cut_r)$ is always true, but that $v_{LP}(Cut_r) \leq v_{LP}(TransCut)$ does not in general hold—i.e., the LP relaxation of $TransCut$ is *strictly stronger* [14] than the LP relaxation of Cut_r. Consequently, also the LP relaxation of $TransCut$ together with the constraints (30) and (31) is strictly stronger than the LP relaxation of Cut_r

Proposition 14. *The relation $v_{LP}(TransCut) \leq v_{LP}(Cut_r)$ is always true and there exist problem instances for which the inequality is strict.*

Proof. Due to its technical nature, the proof of the first part of the proposition—that $v_{LP}(TransCut) \leq v_{LP}(Cut_r)$ holds—is provided in Appendix A.3. For the second part consider the directed RMWCSP instance depicted in Figure 6. In the optimal LP solution to Cut_r each vertex $v \notin R$ has the value $y(v) = 0.5$. Hence, the objective is -2.5. On the other hand, $v_{LP}(TransCut) = -3$; one optimal solution is $x(a) = 0.5\ \forall a \in A$, but also any (integer) optimal solution to $TransCut$ is an optimal LP solution. □

The argumentation of the proof implies that for the directed RMWCSP in Figure 6 in particular $v_{LP}(TransCut) = v(TransCut)$ holds.

One might argue that despite its inferior LP-relaxation the Cut_r formulation leads to a problem with far less variables, as it only considers nodes and $TransCut$ moreover requires additional arcs for each positive non-terminal. However, preprocessed instances are in practice sparse and include only a small amount of positive-weight vertices [29]. Moreover, the fact that for none of the 147 instances considered in this article branching was necessary underlines the practical strength of the $TransCut$ formulation.

4 Primal Heuristics

Several primal heuristics for the MWCSP have been described in the literature. In [5], for instance, a breadth-first-search based heuristic that makes use of the reduced cost obtained during a branch-and-cut algorithm was suggested. In [21] several variants of a dual-ascent based heuristic for the rooted prize-collecting Steiner tree problem were suggested and were also applied for the MWCSP by using a transformation initially introduced in [12]. Moreover, in [17] a connected subgraph was constructed by running a heuristic for the Steiner arborescence problem on the problem obtained by Transformation 1.

4.1 Constructive Heuristics

As the name suggests, constructive heuristics build up a new solution from scratch, repeatedly extending the current sub-solution.

A Greedy Approach

The first heuristic is similar to the classical shortest paths heuristic for the Steiner tree problem in graphs [10, 31] and is conceptually straightforward: Starting with a single vertex, the heuristic iteratively connects the current subtree to vertices of positive weight as long as this is profitable.

In detail, the heuristic works as follows. Initially, step 1 and step 2 of Transformation 1 are performed to obtain a directed graph (V', A') and arc-weights c'. Next, a start vertex v_r is chosen and the initial connected subset S is defined as $V(S) := \{v_r\}, E(S) := \emptyset$. Then, Dijkstra's algorithm is started from S, using only outgoing arcs. Denote the distance function of Dijkstra's algorithm by d. Whenever a vertex v_i of non-negative weight is scanned (i.e. removed from the priority queue of Dijkstra's algorithm) and $d(v_i) \leq p(v_i)$, the corresponding path from S to v_i is added to S, and all vertices of this path are reinserted into the priority queue of Dijkstra's algorithm with their distance value d set to 0. As soon as a vertex is scanned whose distance exceeds the maximum among all weights of vertices that are not part of the incumbent solution S, the computation is stopped. The final connected subset S' constitutes a feasible solution.

If an LP solution to the MWCSP is available, the arcs weights c' used for the heuristic can be adapted. For instance, assume an LP solution $x \in [0,1]^{|A'|}$ to the SAP (V', A', T', c', r') obtained by Transformation 1 is given. Thereupon, define for each $i \in \{1, ..., n\}$ the variable $z(i) := x(\delta^-(v_i))$. Furthermore, let $A := \{(v_i, v_j) \in A' \mid \{v_i, v_j\} \in E\}$ and set, with a slight abuse of notation, $c'(a) := (1 - z(j)) \cdot c'(a)$ for all $a = (v_i, v_j) \in A$. Thus, a stimulus for the heuristic to choose vertices with high (fractional) indegree in the LP solution is provided. Moreover, the heuristic is started from a (constant) number of distinct vertices. To this end, vertices v_i with highest value $z(i)$ in the incumbent LP solution are chosen as starting points. In case of ties, vertices with higher weight p are preferred.

In the following the above algorithm is referred to as *Greedy Construction* (GC) heuristic. It should be noted that the idea of reinserting vertices into the priority queue of Dijkstra's algorithm was already used in a heuristic for the Steiner tree problem in graphs [10]. Furthermore, the concept of using LP solutions to guide primal heuristics for combinatorial optimization problems is widely used, see for instance [17, 20].

A Reduction-based Approach

The first of two reduction based approaches described in this article builds on a concept introduced as *prune* in [26] for Steiner tree problems in graphs. By virtue of the PVD method introduced in Section 2, this approach can now be used for the MWCSP as well. While for the original PVD test an upper bound is provided by the weight of a given solution, in the prune heuristic the bound is chosen such that in each iteration a certain number of vertices is eliminated. Thereupon, all exact reductions methods are executed on the reduced graph, motivated by the assumption that the (possibly inexact) eliminations performed by the bound-based method will allow for further (exact) reductions. To avoid infeasibility, initially a feasible solution is computed (by using GC and the subsequently described local-search heuristics) of which no vertices or edges are allowed to be deleted by the (inexact) bound-based method.

The second reduction-based heuristic approach is borne from the combination of the prune heuristic and dual-ascent: the *ascend-reduce* method (based on an approach originally suggested in [34] for the Steiner tree problem in graphs). Let P_{MW} be the original MWCSP and P' the

SAP resulting from Transformation 1. The ascend-reduce heuristic attempts to find a good solution on the subproblem $\tilde{P}_{MW}$ constituted by the undirected edges of P_{MW} corresponding to zero-reduced-cost paths in P' from the root to all additional terminals—in this article a solution is computed by employing reduction techniques and heuristics. This approach is motivated by the assumption that notable similarities exist between an optimal (or near-optimal) Steiner tree and the LP solution corresponding to the reduced costs provided by dual-ascent.

4.2 Local Search Heuristics

Given a solution S to a problem, a local search algorithm examines a *neighborhood* of S, i.e. a set of solutions obtainable from S by performing a predefined set of operations. Consequently, all heuristics described in this section assume that a feasible solution S is given.

Greedy Extension

The *Greedy Extension* (GE) heuristic works in two phases: In phase one all vertices $V(S)$ are inserted into the priority queue of Dijkstra's algorithm with their distance values set to 0. Thereupon, the GC heuristic is executed. However, the GC algorithm is only stopped when all vertices of positive weight have been scanned (the criterion for including vertices to the current solution remains unchanged). Furthermore, the heuristic saves (a constant number) $\alpha \in N$ (or as many as exist) vertices $t'_k \in T \setminus V(S), k = 1, ..., \alpha$ such that $p(t'_k) - d(t'_k) \geq p(t_u) - d(t_u)$ for all other vertices $t_u \in T \setminus V(S)$. In phase two, α iterations $k = 1, ..., \alpha$ are performed; in each iteration vertex t'_k is connected to S (by using the shortest path computed by Dijkstra's algorithm) to obtain a solution S_k. Next, the GC heuristic is executed from S_k and updates S in case a better solution could be found. Analogously to the GC heuristic, GE can be executed with altered arc weights if an LP solution is available.

Vertex Inclusion and Vertex Exclusion

The idea of the *Vertex Inclusion* (VI) heuristic is to add a vertex to a given solution such that other negative-weight vertices of the solution can be discarded. First, compute a spanning tree S_{span} on S such that as many vertices as possible are of degree 2. Next, iterate through all neighboring vertices v_i of S: Let $\delta_{S,i}$ be the set of all edges between v_i and $V(S)$. If $|\delta_{S,i}| \leq 1$, continue. Otherwise, add an (arbitrary) edge $a'_0 \in \delta_{S,i}$ to S_{span}. Afterwards, iteratively add each edge $a'_j \in \delta_{S,i} \setminus \{a'_0\}$ to S_{span}. Whenever a new edge has been added, search for a minimum-weight sequence of vertices of degree 2 (with respect to S_{span}) on the newly created cycle. If such a sequence being of negative weight exists (including single vertices), remove it from S_{span}, otherwise remove a'_j. When all edges $\delta_{S,i}$ have been checked and if the weight of the removed vertices is smaller than that of v_i, leave S_{span} in its modified form, otherwise restore it. In the implementation of the heuristic (linear) *link-cut trees* [30] are used. This data structure allows to easily dis- and reconnect trees.

An adaptation of the VE heuristic could also be used for related problems such as the Steiner tree problem in graphs, for which the literature describes a weaker version that merely seeks to eliminate edges, see e.g. [24].

A complementary approach is taken by the *Vertex Exclusion* (VE) heuristic: it aims to remove vertices of S. Consider the connected subgraph

$$G_S := (V(S), E[V(S)])$$

. Thereupon, the heuristic employs the reduction package Reduce-Basic on G_S to obtain a new graph $\tilde{G}_S$. On this new graph the GE heuristic is used to obtain a solution $\tilde{S}$. Finally, this solution is retransformed to a solution S' to the original problem.

5 Solving to Optimality

With several algorithmic components introduced and discussed two central questions remain: How to assemble these threads and weave them into a coherent exact solver, and, equally important, how does the resulting algorithm perform?

5.1 A Full-fledged Exact Solver

The exact solver described in this article is realized within the Steiner tree problem solver SCIP-Jack, which itself is embedded in the academic mixed-integer program solver SCIP [23]. As the implementations of preprocessing (and propagation) and the IP formulation have already been discussed in the preceding sections, only the missing pieces in-between will be explained.

The first, and indispensable, piece is the separation routine for the SAP. The separation algorithm uses a maximum-flow algorithm with warm-start capabilities to detect violated inequalities in the Directed Cut Formulation [20]. Another component instrumental for empirically successful exact solving is constituted by the primal heuristics described in Section 4. In the implementation for this article, ascend-reduce uses the prune heuristic to find a solution on the graph obtained by dual-ascent and employs GE, VI and VE to improve it. In turn, the prune heuristic calls GC to obtain an initial feasible solution, calls the local search heuristics GE, VI to improve it, and employs Reduce-Basic as a reduction package. Finally, to improve the solution obtained by ascend-reduce all local search heuristics are used. This heuristic package is repeatedly used during preprocessing. Furthermore, the heuristics GC, GE, and VI are used during branch-and-bound.

Finally, branching is performed on vertices—by assigning the vertex v_i to branch on weight $p(v_i) = \infty$ in one branch-and-bound child node and removing it in the other—which seems to be the natural choice for the MWCSP. However, none of the instances tested in this article go past the root node, so this choice is of rather limited practical impact.

A look beyond the surface of the new exact MWCSP solver reveals an intricate synergy of the three major solving components introduced in this article: First, heuristics and reduction techniques are deeply intertwined. Reduction methods are crucial for the success of both prune and ascend-reduce, while the quality of the primal bound obtained by these heuristics determines the effectiveness of the dual-ascent reduction method. Indeed, the prune heuristic could only be realized due to the newly introduced PVD concept. Furthermore, only the combination of dual-bound and reduced costs obtained by dual-ascent, and the primal bound provided by ascend-reduce consistently gives rise to the transformation of MWCSP to RMWCSP and the subsequent application of the *TransCut* formulation. In turn, on the SAP obtained from this transformation one can again execute the dual-ascent reduction method. As demonstrated in the next section, with any of these three components—heuristics, reduction techniques, or new IP formulation—being deactivated, the performance of the solver considerably deteriorates.

5.2 Computational Results

The computational evaluation for this article has been performed on the five test sets described in Table 1. The ACTMOD and JMPALMK instances were all solved to optimality in the course of the 11th DIMACS Challenge [1], whereas the SHINY test set [22] has only recently been

introduced. The two test sets with the largest instances, HANDBI and HANDBD, are also from the 11th DIMACS Challenge, but were originally formulated as prize-collecting Steiner tree problems. However, the instances have uniform edge weights and can therefore be transformed to MWCSP. Most of these large instances proved to be intractable for the solvers participating in the 11th DIMACS Challenge and could only recently be solved to optimality [21]. Still, three instances of these two test sets have remained unsolved until now.

Name	Instances	$\|V\|$	$\|E\|$	Status	Description
JMPALMK	72	500-1500	2597-20527	solved	Euclidean, randomly generated instances introduced in [5].
SHINY	39	232-3828	202-4494	solved	Real-world instances from network enrichment analysis in computational biology [22].
ACTMOD	8	2034-5226	3335-93394	solved	Real-world instances derived from from integrative biological network analysis [12].
HANDBI	14	158400	315808	unsolved	Images of hand-written text derived from a signal processing problem [1].
HANDBD	14	169800	338551	unsolved	Images of hand-written text derived from a signal processing problem [1].

Table 1: Classes of MWCSP instances.

The computational experiments have been performed on a cluster of Intel Xeon X5672 CPUs with 3.20 GHz and 48 GB RAM. CPLEX 12.7.1[1] was employed as the underlying LP solver. SCIP-Jack also allows to switch to the non-commercial, but slower, LP solver SoPlex [35]. For all computations a time limit of one hour was set and only single-thread mode was used.

	mean time [s]		max time [s]		solved instances	
Test set	[17]	new	[17]	new	[17]	new
JMPALMK	0.0	0.0	0.5	0.1	72	72
SHINY	0.1	0.0	1.1	0.1	39	39
ACTMOD	0.3	0.1	1.3	0.3	8	8
HANDBD	124.6	8.3	>3600	>3600	13	13
HANDBI	127.5	8.9	>3600	>3600	12	13

Table 2: Computational comparison of SCIP-Jack 1.0, denoted by *[17]*, and the MWCSP solver described in this article, denoted by *new*.

Table 2 provides aggregated results for SCIP-Jack 1.0 and the new solver. The first column shows the test set considered in the current row. Columns two and three show the shifted geometric mean [2] (with shift 1) of the run time taken by the respective solvers. The next two columns provide the maximum run, and the last two columns the number of solved instances.

The results for the first three tests sets—JMPALMK, SHINY, and ACTMOD—reveal a strong performance of both SCIP-Jack 1.0 and the new solver. The maximum run time on the three test sets is more than a factor of 5 faster with the solver described in this article, but most instances of the test sets are solved within 0.1 seconds by either solver—as reflected in a shifted geometric mean of 0.1 seconds or less on JMPALMK and SHINY. The best other results for these test sets are reported in [21], for ACTMOD and JMPALMK, and in [22], for SHINY. The new solver is both on average and with respect to the maximum run time more than one order of magnitude faster than the approach in [21] and more than two orders of magnitude faster than the solver described in [22]. Furthermore, the maximum run time in [22] on the SHINY test set is more than a thousand seconds (with a few instances remaining unsolved), while it is less than

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

	Primal/new primal		Gap [%]		Time [s]	
Instance	[17]	new	[17]	new	[17]	new
handbd01	728.963591	–	–	–	22.1	2.5
handbd02	296.496486	–	–	–	202.5	10.1
handbd03	135.070605	–	–	–	89.3	7.6
handbd04	1813.95916	–	–	–	340.6	8.1
handbd05	105.474688	–	–	–	91.7	6.6
handbd06	1528.76544	–	–	–	248.9	9.5
handbd07	77.861959	–	–	–	93.3	5.4
handbd08	1368.16677	–	–	–	56.7	7.4
handbd09	62.71716	–	–	–	99.3	4.9
handbd10	1137.42973	–	–	–	33.3	3.4
handbd11	46.772533	–	–	–	96.2	4.1
handbd12	321.204744	–	–	–	75.7	2.8
handbd13	13.20885	13.196987	0.82	0.14	> 3600.0	> 3600.0
handbd14	4379.10424	–	–	–	142.9	0.5
handbi01	1358.56338	–	–	–	20.5	2.2
handbi02	531.810883	–	–	–	77.0	6.7
handbi03	243.134201	–	–	–	81.4	9.0
handbi04	3202.18574	3202.18574	0.02	–	> 3600.0	24.7
handbi05	184.467331	–	–	–	75.9	8.0
handbi06	2921.54472	–	–	–	139.5	9.3
handbi07	150.974258	–	–	–	70.9	7.0
handbi08	2270.28462	–	–	–	38.8	4.5
handbi09	107.768806	–	–	–	88.4	6.4
handbi10	1874.29296	–	–	–	17.3	2.3
handbi11	68.944709	–	–	–	74.6	4.5
handbi12	138.257023	–	–	–	211.9	2.1
handbi13	4.268146	4.251	1.60	0.13	> 3600.0	> 3600.0
handbi14	7881.76874	–	–	–	228.7	1.1

Table 3: Computational comparison of SCIP-Jack 1.0, denoted by *[17]*, and the MWCSP solver described in this article, denoted by *new*.

0.1 seconds for the new solver. The computational environment for the experiments in [22] is described as AMD Opteron 6380 CPUs with 2.5 GHz. The solver described in [21] is publicly available and the results reported here for this solver were obtained on the same cluster (detailed above) that was used for the new solver and SCIP-Jack 1.0.

The results for the two hardest test sets, HANDBD and HANDBI, show a more pronounced difference between the two solvers also with respect to the shifted geometric mean, with a difference of more than an order of magnitude. More detailed results for these two sets are reported in Table 3. The first column of the table lists the name of the considered instance. The second column shows the solution value of the best solution found by SCIP-Jack, transformed back to a solution to the original prize-collecting problem—which unlike the MWCSP is a minimization problem. The third column provides the percentage of the duality gap defined as $\frac{\text{ub}-\text{lb}}{\text{ub}}$ for final upper and lower bound ub and lb. The fourth column shows the run time of SCIP-Jack for this instance. The last three columns provide the corresponding information for the new solver described in this article (with the primal bound only printed if the gap has improved). SCIP-Jack leaves three instances—those that have also been intractable for the approach in [21]—unsolved, while the new solver reaches optimality for one of them and significantly reduces the primal-dual gap for the other two. The previously best known gaps [21] are 1.87 % for *handbi13* and 0.41% for *handbd13*, whereas the new solver achieves gaps of 0.14 % and 0.13 % respectively. The reported gaps are already computed in preprocessing, as the remaining solving time is spent in the

	Test set	Reduction tests	Heuristics	RMWCSP Transformation
mean time	ACTMOD	**+123**	**+77**	**+31**
	HANDBD*	**+544**	**+27**	**+26**
	HANDBI*	**+462**	**+78**	**+43**
max time	ACTMOD	**+125**	**+150**	**+100**
	HANDBD*	**+698**	**+1415**	**+224**
	HANDBI*	**+1194**	**+11150**	**+678**

Table 4: Each column reports the results of the MWCSP solver without the specified new methods. The values denote the percentual change with respect to the default settings (see Table 2).

first LP-solve. The *handbi04* instance has not been solved to optimality before. Furthermore, as compared to SCIP-Jack also the run times of the new solver are notably reduced—all instances are solved faster, many by more than one order of magnitude. A prominent example for the improved run time is the instance *handbi14* for which the difference in run time is more than two orders of magnitude. Similarly, the new solver is on all instances faster than the framework described in [21]; for several instances by more than one or even two orders of magnitude.

To get some insight into the individual impact of the three main algorithmic components described in this article, Table 4 shows results for the new solver when deactivating either the new reduction tests, the heuristics, or the transformation to RMWCSP. As the vast majority of the JMPALMK and SHINY instances can be solved in less than 0.01 seconds by both SCIP-Jack and the new solver, these two test sets are not considered in the experiment. The first three rows list the percentual change in the average run time (with respected to the shifted geometric mean); the last three rows provide the corresponding percentual change with respect to the maximum run time. In this way, the first column of each row states the test set to be considered. Ensuing, each of the next three columns provides the result of excluding the solving component specified in the head of the table. It should be noted that positive values signify a favourable impact of the respective algorithmic component on the new solver. The (unsolved) instances *handbd13* and *handbi13* are not considered in the experiment—the corresponding smaller test sets are denoted by HANDBD* and HANDBI*.

The results of deactivating the new reduction techniques are dramatic, both the mean and the maximum run time increases by more than 100 % on all test sets. The HANDBD* and HANDBI* instances show the strongest effect, an increase of more than a factor of 5 in both categories. Interestingly, the maximum run time increase does not differ significantly from the increase of the mean time, which suggests that most instances are similarly affected by the reduction techniques. It should be noted that part of the performance change derives from a new (more cache-efficient) implementation of the dual-ascent reduction method. While the new implementation does not affect ACTMOD, SHINY, and JMPALMK (as their instances are already drastically reduced when dual-ascent is applied), it leads to most HANDBD* and HANDBI* instances being solved by a factor of around 2 faster than by SCIP-Jack.

Also the heuristics exert a strong impact on all three test sets, especially with respect to the maximum run time. In contrast to the reductions techniques, there is a large difference between the maximum and mean time increase. This observation suggests that the heuristics are more effective on the subset of hard instances. And indeed, while most instances are not strongly affected from deactivating the new heuristics, the run times of a few harder instances such as *handbi06* is notably increased. Similarly, the formerly unsolved *handbi04* instance cannot be

solved anymore when the new heuristics are deactivated. Also (although this cannot be seen in the table) the dual gaps of *handbi13* and *handbd13* are more than doubled. An explanation for this behavior is that without the new heuristics the duality gaps for the bound-based reduction methods (dual-ascent and PVD) significantly increases on the hard instances, which notably reduces the strength of the reduction methods. In the same way, the number of instances for which the transformation to RMWCSP can be applied is reduced—which again is only important for the harder instances that are not solved before the transformation could possibly be applied.

Finally, of the three components the transformation to RMWCSP shows the least overall impact, although for the harder instances it is still considerable—with a run time increase of more than 600 % for the *handbi04* instance. A reason for the comparatively smaller impact is that most instances are already solved to optimality or at least drastically reduced before the transformation can even be applied. Still, the performance on the hard instances strongly demonstrates its use.

Overall, the computational results reveal a strong performance of the new solver, which outperforms the previous state of the art on all test sets.

6 Conclusion and Outlook

This article has introduced an exact solving approach for the MWCSP based on three central components: Preprocessing, an integer programming formulation based on graph transformations, and heuristics. Arguably, each of these components deserves individual interest, be it of more practical (as for the heuristics) or likewise theoretical (as for the reduction techniques) nature. Notwithstanding, only the—surprisingly symbiotic—synergy of all three components ultimately gives rise to paramount computational advancement, outperforming previous approaches, and allowing to solve three sets of benchmark instances in fractions of a second. Furthermore, one large-scale benchmark instance from the 11th DIMACS Challenge originally formulated for the prize-collecting Steiner tree problem can be solved for the first time to optimality, and the best known primal-dual gap of two other ones can be considerably reduced.

Further work could focus on developing new reduction techniques, which are, at least in the setting of this article, a pivotal solving ingredient. Hopefully, the analyses provided in this article contribute to a better understanding of reduction techniques for the MWCSP and set the stage for further developments. Moreover, it might be well worthwhile to study new IP formulations for the MWCSP to further improve the strength of the LP-relaxation.

The integration of the algorithmic framework developed in this article into SCIP gives way to a solver freely available for academic use and completely open in source code—as part of the next SCIP release. Therefore, this article and the accompanying exact solver provide access to both researchers interested in particular solving techniques or implementations and to practitioners (for instance from computational biology) who acquire to solve their MWCSP instances to optimality. In this way, the authors hope to contribute not only to further improvements of solving technology for the MWCSP, but also to increased utilization of the MWCSP for solving real-world problems.

7 Acknowledgements

The authors would like to thank Felipe Serrano for several fruitful discussions and helpful comments related to the work presented in this article. Further thanks go to Gerald Gamrath, Ambros Gleixner, Gregor Hendel, Benjamin Müller, and Yuji Shinano.

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*. 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: August 10. 2017.

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

[3] Nicolas Alcaraz, Josch Pauling, Richa Batra, Eudes Barbosa, Alexander Junge, Anne GL Christensen, Vasco Azevedo, Henrik J. Ditzel, and Jan Baumbach. Keypathwayminer 4.0: condition-specific pathway analysis by combining multiple omics studies and networks with cytoscape. *BMC Systems Biology*, 8(1):99, Aug 2014.

[4] 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.

[5] 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.

[6] Eduardo Álvarez-Miranda, Ivana Ljubić, and Petra Mutzel. *The Rooted Maximum Node-Weight Connected Subgraph Problem*, pages 300–315. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[7] 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.

[8] Mohamed Didi Biha, Hervé L. M. Kerivin, and Peh H. Ng. Polyhedral study of the connected subgraph problem. *Discrete Mathematics*, 338(1):80–92, 2015.

[9] 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.

[10] Marcus Poggi de Aragao and Renato F. Werneck. On the Implementation of MST-based Heuristics for the Steiner Problem in Graphs. In *Proceedings of the 4th International Workshop on Algorithm Engineering and Experiments*, pages 1–15. Springer, 2002.

[11] 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.

[12] 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.

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

[14] H.A. Eiselt and Carl-Louis Sandblom. *Integer Programming and Network Models*. Springer, 2000.

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

[16] Matteo Fischetti, Markus Leitner, Ivana Ljubić, 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 Computation*, 9(2):203–229, Jun 2017.

[17] Gerald Gamrath, Thorsten Koch, Stephen Maher, Daniel Rehfeldt, and Yuji Shinano. SCIP-Jack A solver for STP and variants with parallelization extensions. *Mathematical Programming Computation*, 9(2):231 – 296, 2017.

[18] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

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

[20] Thorsten Koch and Alexander Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

[21] Markus Leitner, Ivana Ljubi, Martin Luipersbeck, and Markus Sinnl. A dual ascent-based branch-and-bound framework for the prize-collecting steiner tree and related problems. *INFORMS Journal on Computing*, 30(2):402–420, 2018.

[22] A. A. Loboda, M. N. Artyomov, and A. A. Sergushichev. *Solving Generalized Maximum-Weight Connected Subgraph Problem for Network Enrichment Analysis*, pages 210–221. Springer International Publishing, Cham, 2016.

[23] Stephen J. Maher, Tobias Fischer, Tristan Gally, Gerald Gamrath, Ambros Gleixner, Robert Lion Gottwald, Gregor Hendel, Thorsten Koch, Marco E. Lübbecke, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Dieter Weninger, Jonas T. Witt, and Jakob Witzig. The scip optimization suite 4.0. Technical Report 17-12, ZIB, Takustr.7, 14195 Berlin, 2017.

[24] Michel Minoux. Efficient greedy heuristics for steiner tree problems using reolptimization and super modularity. *INFOR: Information Systems and Operational Research*, 28(3):221–233, 1990.

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

[26] Tobias Polzin and Siavash Vahdati Daneshmand. Improved algorithms for the steiner problem in networks. *Discrete Appl. Math.*, 112(1-3):263–300, September 2001.

[27] Tobias Polzin and Siavash Vahdati Daneshmand. *Extending Reduction Techniques for the Steiner Tree Problem*, pages 795–807. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[28] Daniel Rehfeldt and Thorsten Koch. Transformations for the Prize-Collecting Steiner Tree Problem and the Maximum-Weight Connected Subgraph Problem to SAP. *Journal of Computational Mathematics*, 36(3):459 – 468, 2018.

[29] Daniel Rehfeldt, Thorsten Koch, and Stephen Maher. Reduction techniques for the prize-collecting steiner tree problem and the maximum-weight connected subgraph problem. Technical Report 16-47, ZIB, Takustr.7, 14195 Berlin, 2016.

[30] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, June 1983.

[31] H. Takahashi and Mastsuyama A. An approximate solution for the Steiner problem in graphs. *Mathematica Japonicae*, 24:573 – 577, 1980.

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

[33] Yiming Wang, Austin Buchanan, and Sergiy Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, pages 1–31, 2017.

[34] R.T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271287, 1984.

[35] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus.* PhD thesis, Technische Universität Berlin, 1996.

A Further Proofs

A.1 Proof of Proposition 1

Proof. Let S be an optimal solution to P_{MW} such that $v_i \in V(S)$. As before, it is assumed (without limiting generality) that S is a tree. Denote the (unique) path in S between v_i and a $t_j \in V(S) \cap T$ by Q_j and set $\mathcal{Q} := \{Q_j \mid t_j \in V(S)\}$. First, note that $|\mathcal{Q}| \geq 2$, because if $\mathcal{Q}$ just contained one path, say Q_j, it would follow for $S' := \{t_j\}$ that $v_i \notin S'$ and $P(S') \geq P(S)$ (which contradicts the assumptions of the proposition). Second, if a vertex v_k is contained in two distinct paths of $\mathcal{Q}$, 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 (distinct) paths $Q_k, Q_l \in \mathcal{Q}$ such that $V[Q_k] \cap V[Q_l] = \{v_i\}$. Otherwise, due to the precedent observation, all paths in $\mathcal{Q}$ would have one edge $\{v_i, v_i'\}$ in common, which could be discarded to obtain a tree S' with $v_i \notin V[S']$ and $P(S') \geq P(S)$.

Now, choose two distinct paths $Q_k \in \mathcal{Q}$ and $Q_l \in \mathcal{Q}$ with minimum number of combined H-boundary edges and $V[Q_k] \cap V[Q_l] = \{v_i\}$. 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 from t_r up to the last vertex still in H_{t_r}. Suppose that Q_k has a vertex $v_q \in V(S)$ in common with a Q_r'. Consequently, $Q_l \cap Q_r = \{v_i\}$, because S is cycle-free. Furthermore, according to the preceding observations, Q_k and Q_r have to contain a joint subpath including v_i and v_q. But this implies that Q_k contains at least one additional H-boundary edge (in order to be able to reach t_k, which is by definition not in H_{t_r}). Therefore, and due to $V[Q_l] \cap V[Q_r] = \{v_i\}$, the path 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 vertex in common with any Q'_r. Conclusively, the paths Q_k, Q_l have only the vertex v_i in common and all paths Q'_r are vertex disjoint and also do not have any vertex in common with both Q_k, Q_l. Using their combined weight, one can obtain an upper bound on the weight of S by:

$$\begin{aligned}
P(S) &= \sum_{v \in V(S)} p(v) \\
&\leq \Bigg(\sum_{Q_r \in \mathcal{Q}^-} P(Q'_r) \Bigg) + P(Q_k) + P(Q_l) - p(v_i) \\
&\leq \sum_{t \in T} r_H^+(t) - \min\{r_H^+(t) + r_H^+(t') \mid t, t' \in T,\ t \neq t'\} + P(Q_k) + P(Q_l) - p(v_i) \\
&\leq \sum_{t \in T} r_H^+(t) - \min\{r_H^+(t) + r_H^+(t') \mid t, t' \in T,\ t \neq t'\} + \overline{d}(v_i, \overline{v}_{i,1}) + \overline{d}(v_i, \overline{v}_{i,2}) \\
&\quad - p(v_i).
\end{aligned}$$

The first inequality follows from above discussed properties of the paths in Q^- and the paths Q_k and Q_l. The second inequality uses the fact that the weight of each path $Q_r \in Q^-$ can be bounded from above by $r_H^+(t_r)$. Finally, the third inequality exploits that the paths Q_k and Q_l do not contain any intermediate vertices of positive weight and that there weight can therefore be bounded by using the distance function $\overline{d}$. Consequently, the proposition is proven. □

A.2 Proof of Proposition 2

Proof. Let S be an optimal solution to P_{MW} (and, as before, a tree) such that $v_i \in V(S)$ and $|V(S) \cap T| \geq 2$. Define $\mathcal{Q}$ as in the proof A.1 and note that $\mathcal{Q} \neq \emptyset$ (because of $|V(S) \cap T| \geq 2$). Next, choose a path $Q_k \in \mathcal{Q}$ with a minimum number of H-boundary edges. Further, define $\mathcal{Q}^- := \mathcal{Q} \setminus \{Q_k\}$. As before, for all $Q_r \in \mathcal{Q}^-$, denote by Q'_r the subpath of Q_r from t_r up to the last vertex still in H_{t_r}. As in proof A.1, one validates that the Q'_r are pairwise vertex disjoint and that Q_k has no vertex in common with any Q'_r. One goes on to obtain an upper bound on the weight of S:

$$\begin{aligned}
P(S) &= \sum_{v \in V(S)} p(v) \\
&\leq \Bigg(\sum_{Q_r \in \mathcal{Q}^-} P(Q'_r) \Bigg) + P(Q_k) \\
&\leq \sum_{t \in T \setminus \{v_i\}} r_H^+(t) - \min\{r_H^+(t) \mid t \in T \setminus \{v_i\}\} + P(Q_k) \\
&\leq \sum_{t \in T \setminus \{v_i\}} r_H^+(t) - \min\{r_H^+(t) \mid t \in T \setminus \{v_i\}\} + \overline{d}(v_i, \overline{v}_{i,1}).
\end{aligned}$$

These inqualities conclude the proof of the proposition. □

A.3 Proof of Relation $v_{LP}(TransCut) \leq v_{LP}(Cut_r)$

Proof. Let $P_{RMW} = (V, A, R, p, r)$ be a directed RMWCSP and let $P' = (V', A', T', c', r')$ be the corresponding SAP obtained from applying Transformation 2. Furthermore, define $D := (V, A)$ and $D' := (V', A')$. Let $x \in [0,1]^{|A'|}$ be a solution to the

LP-relaxation of the $TransCut$ formulation such that the following minimality condition holds: none of the values $x(a)$ for $a \in A'$ can be reduced without making the solution infeasible (this condition is satisfied for at least one optimal LP solution). In the following it will be demonstrated that there exists an LP solution $y \in [0,1]^{|V|}$ to the Cut_r formulation to P_{RMW} of value $p^T y = \sum_{v \in V: p(v)>0} p(v) - {c'}^T x$.

To this end, define y as follows: For each $v \in V$ set $y(v) := x(\delta^-_{D'}(v))$. Next, let $v_i \in V$ such that $y(v_i) > 0$ and let N be an (r, v_i)-separator. One needs to verify that $y(N) \geq y(v_i)$ and that $y(v_i) \leq 1$. In order to acknowledge these properties, first define $S_{v_i} := \{v \in V' \mid N \text{ is } (r,v)\text{-separator}\}$.

To facilitate notation, δ will in the following be used as an equivalent for $\delta_{D'}$ (and δ^-, δ^+ correspondingly). In this way, denote by Δ the set of all $a \in \delta^-(v_i)$ with $x(a) > 0$. Let $W_1, ..., W_u \subset V'$ such that

1. for $q = 1, ..., u$:
   (a) $W_q \cap T' \neq \emptyset$, $r' \notin W_q$, $x(\delta^-(W_q)) = 1$
   (b) $\Delta_q := \Delta \cap \delta^-(W_q) \setminus \bigcup_{j=1}^{q-1} \delta^-(W_j) \neq \emptyset$
2. $\bigcup_{q=1}^{u} \Delta_q = \Delta$

The existence of such sets follows from the initial minimality assumption on x. Moreover, set $W_{u+1} := S_{v_i}$. In the following, we will define a finite series of arcs sets $(\overline{A'}_q)$ that converge to a subset of $\delta^-(S_{v_i})$ such that the sum of its x values is at least $x(\Delta)$. From there it is a short way to proving the proposition.

Define the sets X_q, A_q, A'_q and $\overline{A'}_q$ iteratively for $q = 1, 2, ..., u$. Initially, set $\overline{A'}_0 := \emptyset$ and $A_q := \delta^-(W_q) \cap \overline{A'}_{q-1}$ for $q \in \{1, ..., u\}$. Let X_q be the set of all vertices $v \in (\cap_{j=q+1}^{u+1} W_j) \setminus W_q$ such that there is a directed path in $(\cap_{j=q+1}^{u+1} W_j) \setminus W_q$ from v to the tail of an arc in $A_q \cup \Delta_q$. Furthermore, set

$$A'_q := (\bigcup_{j=q+1}^{u+1} \delta^-(W_j)) \ \cap \ (\delta^-(X_q) \cup A_q \cup \Delta_q). \tag{36}$$

and

$$\overline{A'}_q := (\bigcup_{j=1}^{q} A'_j) \ \cap \ (\bigcup_{j=q+1}^{u+1} \delta^-(W_j)). \tag{37}$$

Next, we iterate from $q = 0$ to $q = u$, and show that in each iteration $q \in \{0, 1, ..., u\}$ the invariant

$$x(\overline{A'}_q) \geq x(\bigcup_{j=1}^{q} \Delta_j) \tag{38}$$

holds. For $q = 0$ the invariant is trivially satisfied (as $x(\emptyset) = 0$), so consider $1 \leq q \leq u$. Due to $x(\delta^-(X_q \cup W_q)) \geq 1$ and $X_q \cap W_q = \emptyset$ it holds that

$$x(\delta^-(W_q)) \leq \delta^-(X_q \dot\cup W_q) = (\delta^-(W_q) \setminus \delta^+(X_q)) \dot\cup (\delta^-(X_q) \setminus \delta^+(W_q)). \tag{39}$$

This equation implies that

$$x(\delta^+(X_q) \cap \delta^-(W_q)) \leq x(\delta^-(X_q) \setminus \delta^+(W_q)) \tag{40}$$

$$\leq x(\delta^-(X_q) \cap \delta^-(\bigcup_{j=q+1}^{u+1} W_j)), \tag{41}$$

where (41) follows from $\delta^-(X_q) \subseteq \delta^+(W_q) \cup \delta^-(\cup_{j=q+1}^{u+1} W_j)$. One can go on to conclude that

$$x(A_q) + x(\Delta_q) = x(A_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) + x(A_q \cap \delta^+(X_q)) + x(\Delta_q) \tag{42}$$

$$\leq x(A_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) + x(\delta^-(X_q) \cap \delta^-(\bigcup_{j=q+1}^{u+1} W_j)) + x(\Delta_q) \tag{43}$$

$$= x(A'_q). \tag{44}$$

Equality (42) follows from the construction of X_q, and inequality (43) follows from $A_q \subseteq \delta^-(W_q)$ and from the inequalities (40)-(41). Finally, equality (44) holds because all three sets are disjoint; in particular, Δ_q and A_q are disjoint because $\Delta_q \cap \cup_{j=1}^{q-1} \delta^-(W_j) = \emptyset$ and therefore $\Delta_q \cap \overline{A'}_{q-1} = \emptyset$. Using definition (37), one obtains that

$$x(\overline{A'}_q) = x((\bigcup_{j=1}^{q-1} A'_j \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) \ \cup \ (A'_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j))) \tag{45}$$

$$\geq x(\overline{A'}_{q-1} \setminus (\delta^-(W_q) \cap \overline{A'}_{q-1}) \cup A'_q) \tag{46}$$

$$= x((\overline{A'}_{q-1} \setminus A_q) \cup A'_q) \tag{47}$$

$$\geq x(\overline{A'}_{q-1}) - x(A_q) + x(A'_q) \tag{48}$$

$$\geq x(\overline{A'}_{q-1}) + x(\Delta_q). \tag{49}$$

First, equality (45) follows from the definition of $\overline{A'}_q$. Inequality (46) follows from the definition of $\overline{A'}_q$ and A'_q. Similarly, equality (47) follows from the definition of A_q. Inequality (48) follows from $\overline{A'}_{q-1} \cap A'_q \subseteq A_q$. Finally, inequality (49) follows from the system (42)-(44).

Consequently, by an induction argument it can be concluded that invariant (38) is satisfied. Finally, at the end of the iterations (at $q = u$) it holds that $\overline{A'}_u \subseteq (\delta^-(S_{v_i}))$, which implies that

$$x\big(\delta^-(S_{v_i})\big) \geq x(\bigcup_{j=1}^{u} \Delta_j) = x(\Delta). \tag{50}$$

Due to $\delta^-(S_{v_i}) \subseteq \delta^-(N)$, one can moreover verify that $x\big(\delta^-(N)\big) \geq x(\Delta)$ is satisfied. Finally, because of $x(\delta^-(N)) \leq \sum_{v \in N} \delta^-(v) = y(N)$ it holds that

$$y(N) \geq x(\Delta) = y(v_i). \tag{51}$$

It remains to be shown that $y(v_i) \leq 1$. Suppose $y(v_i) > 1$. This would imply with the argumentation above that for each $W \subseteq V' \setminus \{r'\}$ with $v_i \in W$ it holds that $x(\delta^-(W)) > 1$, which would contradict the initial minimality assumption on x.

Comparing the objective value of y with that of x, one observes that

$$\begin{aligned}
p^T y &= \sum_{v \in V} \big(p(v) \sum_{a \in \delta_D^-(v)} x(a)\big) \\
&= \sum_{v \in V, p(v) > 0} \big(p(v) \sum_{a \in \delta_D^-(v)} x(a)\big) + \sum_{v \in V, p(v) \leq 0} \big(p(v) \sum_{a \in \delta_D^-(v)} x(a)\big) \\
&= \sum_{v \in V : p(v) > 0} p(v) - \sum_{a \in D' \setminus D} c'(a)x(a) - \sum_{v \in V, p(v) \leq 0} \big(\sum_{a \in \delta_D'(v)} c'(a)x(a)\big) \\
&= \sum_{v \in V : p(v) > 0} p(v) - \sum_{a \in D' \setminus D} c'(a)x(a) - \sum_{a \in D} c'(a)x(a) \\
&= \sum_{v \in V : p(v) > 0} p(v) - \sum_{a \in D'} c'(a)x(a) \\
&= \sum_{v \in V : p(v) > 0} p(v) - {c'}^T x.
\end{aligned}$$

This establishes the required relation. □