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 September 2017)

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 [2, 10, 20] seems to be the predominant application field for the MWCSP, one also encounters the problem in other, disparate, areas such as wildlife conservation [9] and computer vision [8].

The MWCSP has been discussed in various publications, see e.g. [4, 10, 17]. Several articles have contributed to a solid theoretical understanding of the problem [7, 29] while others have addressed exact solving approaches [6, 14, 15]. Recently, also the scope of preprocessing techniques has been considerably enhanced [26].

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 a superimposed 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 4. This section will also

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

discuss the incorporation of the new 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 [15, 21], which allows the reuse of several solving components such as the powerful native separation routine.

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 (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, given a subgraph $G' \subseteq G$ its edges and vertices will be denoted by $V[G']$ and $E[G']$, respectively. The weight for a given solution $S = (V[S], E[S])$ to an MWCSP 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 $T := \{t_1, ..., t_s\}$.

Throughout this article paths are invariably assumed to be simple, i.e. without cycles. Furthermore, this article introduces the distance function $\overline{d}(v_i, v_j)$, defined as the weight of a maximum-weight path between v_i and v_j without intermediary vertices of positive weight—this definition is similar to a distance function defined for the Steiner tree problem in graphs for which no intermediary terminals are allowed [11]. To each vertex v_i of non-positive weight, the k $\overline{d}$-nearest vertices of positive weight will be denoted by $v_{i,1}, v_{i,2}, ..., v_{i,k}$.

Finally, the following general graph notation will be used. The subpath of a path Q between two vertices $v_r, v_s \in V[Q]$ will be denoted by $Q(v_r, v_s)$. Moreover, 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 $G' = (V', E') \subseteq G$ and $W' \subseteq V'$ define $\delta_{G'}(W') := \{\{u, v\} \in E' \mid u \in W', v \in V' \setminus W'\}$. 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. [11, 24]. For the MWCSP the first ground was broken in the course of the 11th DIMACS Challenge, with two articles [3, 13] 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 [26] and a dual-ascent based branch-and-bound algorithm with strong reduction properties was described in [19]. The reductions techniques introduced in the following continue the work started in [26] 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 exceeds a given lower bound (or vice versa) [23, 26]. 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 [23] and the prize-collecting Steiner tree problem [26]—in this way the proof technique for the followings three propositions is similar to the Voronoi reduction proofs in [23] and [26] 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, two more definitions are necessary. First, define for all $t_i \in T$

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

and

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

This definition allows to establish the following bound-based reduction criterion (which is proved in Appendix A.1).

Lemma 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,

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

is an upper bound on the weight of S.

A vertex of non-positive weight can be eliminated if the associated upper bound U in (4) exceeds a known lower bound (e.g. the weight of a given solution). This test is exemplified in Figure 1 for a simple MWCSP instance.

The following proposition can be used to moreover eliminate vertices of positive weight. It can be proven similarly to Proposition 1.

Lemma 2. *Let H be a positive-vertex decomposition of P_{MW} and assume that $|T| \geq 2$. Furthermore, let $t_i \in T$ and assume that an optimal solution S exists such that $t_i \in V[S]$ and $t_j \in V[S]$ for a $t_j \in T \setminus \{t_i\}$. Define*

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

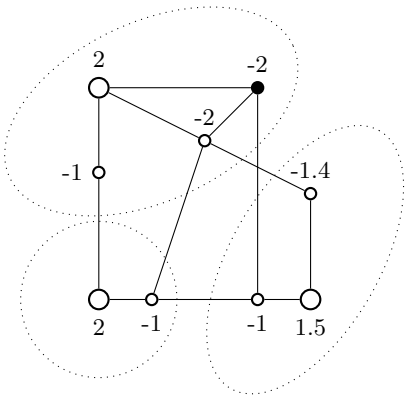


Figure 1: A positive-vertex decomposition of an MWCSP instance with regions marked by dotted ellipses. Proposition 1 guarantees that the (upper right) filled vertex can be deleted if a lower bound higher than $U_2 = 2.5$ is given.

Then

$$U := U_1 + \overline{d}(t_i, t_{i,1}) \tag{6}$$

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.

Lemma 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 each optimal solution S to P_{MW} it holds that $\delta_S(v_i) \geq 3$ (and in particular $v_i \in V[S]$). Finally, let*

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

Thereupon,

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

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 define $T_0' \subsetneq T_0$ 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.

Lemma 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 [16]) 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* [11]: 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.

First, an exclusion concept introduced in [26] will be revisited. Define the *interior cost* of a subpath $Q(v_k, v_l)$ (of a path Q) in (V, E) as:

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

and the *vertex weight bottleneck length* of Q as:

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

The two preceding definitions pave the way for a distance function between two distinct vertices $v_i, v_j \in V$. 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 [26] a reduction test based on the vertex weight bottleneck distance to eliminate vertices of up to degree 5 was employed. Building on the information obtainable from Proposition 3, one can formulate a reduction condition that complements the bottleneck reduction test in [26].

Lemma 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 $v_k, v_l \in \Delta$ it holds that $\hat{d}(v_k, v_l) > p(v_i)$, then there is at least one optimal solution that does not contain v_i.*

Proof. Let S be a tree such that v_i is of degree $k \in \{1, 2\}$ in S. If $k = 1$, 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 $k = 2$, let v_i' and v_i'' be the adjacent vertices of v_i in S 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 $v_j, v_k \in V[Q] \cap V[S']$ such that $V[Q(v_j, v_k)] \cap V[S'] = \{v_j, v_k\}$. Adding $Q(v_j, v_k)$ to S' one obtains a tree that does not contain v_i and is of higher weight than S. □

The corresponding reduction test will be referred to as **Extended Non-Positive Vertex** (**ENPV**) test. In this article it is applied to all negative vertices of degree smaller than or equal to 6—the number which gave the best results in preliminary tests—that have been verified by the PVD test as being eligible.

In both [26] and this article heuristics (with worst-case complexity of $O(1)$ [26]) 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 [28].

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 [28].

Lemma 6. *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 performance of the reduction test ENPV associated with Proposition 5.

In addition to the ENPV test, a more intricate approach encompassing vertex weight bottleneck distances can be devised. However, this approach requires a second component, namely a reduction technique introduced in [26]. This technique is based on:

Lemma 7. *Let $v_i \in V \setminus T$ and $W \subseteq V \setminus \{v_i\}$, $W \neq \emptyset$ such that $(W, E[W])$ is connected and $\sum_{w \in W : p(w) < 0} p(w) \geq p(v_i)$ holds. If*

$$\big\{v \in V \setminus W \mid \{v_i, v\} \in E\big\} \subseteq \big\{v \in V \setminus W \mid \{w, v\} \in E, w \in W\big\} \tag{13}$$

is satisfied, then there is at least one optimal solution that does not contain v_i. The set W will be referred to as MWCS-dominating to v_i.

As will be shown in the following, the problem of finding an MWCSP-dominating set for a non-negative vertex is $\mathcal{NP}$-hard.

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 MWCS-domination problem is to determine whether an MWCS-dominating set to v_i exists.

Lemma 8. *The MWCS-domination problem is $\mathcal{NP}$-complete.*

Proof. Given a vertex subset W 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 MWCS-dominating decision problem is in $\mathcal{NP}$.

In the following it will be demonstrated that the ($\mathcal{NP}$-complete [16]) vertex cover problem can be reduced to the 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 E_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 an MWCS-dominating set W 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 $v_j \in W$. The latter condition is satisfied if and only if there is a vertex cover in G_{cov} of cardinality at most k. □

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

Lemma 9. *Let $v_i \in V$ such that $p(v_i) < 0$ and $\delta(v_i) > 0$, and let Δ be the set of all adjacent vertices of v_i. Furthermore, let $\Delta_1 \subseteq \Delta$ such that $\Delta_1 \neq \emptyset$. Assume that $W \subseteq V \setminus \{v_i\}$ exists such that $(W, E[W])$ is connected and*

$$\Delta_1 \subseteq \big\{v \in V \setminus W \mid \{w, v\} \in E, w \in W\big\} \cup W. \tag{14}$$

Define

$$U_1 := \sum_{w \in W : p(w) < 0} p(w). \tag{15}$$

Let $\Delta_2 := \Delta \setminus \Delta_1$ and choose for each $v_k \in \Delta_2$ an, arbitrary, $v_k' \in W$. Further, let

$$U_2 := \sum_{v_k \in \Delta_2} \hat{d}(v_k, v_k'). \tag{16}$$

If

$$U := U_1 + U_2 > p(v_i), \tag{17}$$

then v_i cannot be contained in any optimal solution to P_{MW}.

Proof. Initially, note that both

$$U_1 \leq 0 \text{ and } U_2 \leq 0. \tag{18}$$

Let $S = (V[S], E[S])$ be a tree that contains v_i. In the following it will be demonstrated how to construct a new tree S''' that does not contain v_i and is of higher weight than S (which implies that S is not optimal). Start with $S' := \{V[S] \setminus v_i, E[S] \setminus \delta_S(v_i)\}$ to obtain a subgraph consisting of $|\delta_S(v_i)|$ disjoint subtrees. One readily verifies that

$$P(S') = P(S) - p(v_i) > P(S).$$

First, let $\Delta_1^S := \Delta_1 \cap V[S]$. Then, add W to S'. This procedure yields a new subgraph S'' with

$$P(S'') \geq P(S') + U_1 = P(S) - p(v_i) + U_1.$$

Moreover, add for each $v_j \in \Delta_1^S \setminus V[S']$ an edge $\{v_j, w_k\}$ for a $w_k \in W$ to S''. Second, let $\Delta_2^S := \Delta_2 \cap V[S]$. For each $v_k \in \Delta_2^S \setminus V[S'']$ add the path corresponding to $\hat{d}(v_k, v_k')$ (with v_k' as defined in the statement of this proposition). Because of condition (17) and (18) none of these

paths contain v_i. Moreover, because of condition (16) the overall procedure reduces the weight of S'' by at most $|U_2|$. Hence, it holds for the new (now connected) subgraph S''' that

$$P(S''') \geq P(S'') + U_2 \geq P(S) - p(v_i) + U_1 + U_2.$$

Finally, S''' can be transformed to a tree of the same weight by computing a spanning tree on its vertices and edges. This tree does not contain v_i and due to (17):

$$P(S''') > P(S).$$

Hence, the proposition is proven. □

One can replace condition (17) by $U \geq p(v_i)$ if none of the paths corresponding to $\hat{d}(v_k, v'_k)$ contains v_i. Associated with Proposition 9 the subsequently sketched reduction test will be used. For each vertex $v_j \in V$ the union of v_j and all non-negative adjacent vertices of v_j is considered as the set W in Proposition 9. Thereupon, one attempts to eliminate neighboring vertices of W by using the conditions 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.

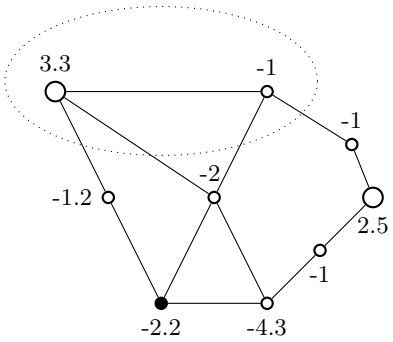


Figure 2: An MWCSP instance. Considering the vertices marked by the dotted ellipse as the set W, the AVS tests yields that the (bottom left) filled vertex can be deleted.

The reduction method will in the following be referred to as **Alternative Vertex Set** (**AVS**) test. It is exemplified for a small MWCSP instance in Figure 2. As in the ENPV test, only lower bounds for the bottleneck distance $\hat{d}$ are computed for AVS. However, despite the twofold use of non-exact algorithms, AVS is empirically the most powerful of the reduction tests introduced so far and often able to delete more than 70 percent of all vertices.

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 4.1.

Besides the PVD, ENPV, and AVS tests, the combined reduction package incorporates three routines described in [26]: 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 six 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, without contraction of edges, is used as a 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 [26] are empirically already notably successful, the combination with the new reduction methods not only enhances the strength of the overall reduction package, but also allows to reiterate the reduction loop less often, which leads to a significantly lower total execution time.

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 [19, 26]. In both [26] 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 [30]:

Formulation 1. *Directed Cut Formulation*

$$\begin{aligned} &min \quad c^T x && && (19)\\ &x(\delta^-(W)) \;\geq\; 1 && \text{for all } W \subset V,\ r \notin W, W \cap T \neq \emptyset, && (20)\\ &x(a) \;\in\; \{0,1\} && \text{for all } a \in A. && (21) \end{aligned}$$

In [30] 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 $O(|A| \min\{|V||T|, |A|\})$ [22]. 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 [25]. 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 [26] together with dual-ascent to eliminate both edges and vertices of an MWCSP. In this article, Transformation 1 is also used in an antipodal way: To show that a vertex is part of at least one optimal solution.

First, it should be noted that in the exact branch-and-cut solver SCIP-Jack 1.0 a transformation for the MWCSP similar to Transformation 1 is used [15]. However, 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). Fortunately, one can obtain the following information from dual-ascent: 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 [5].

For the RMWCSP we devised 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 [5]: 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

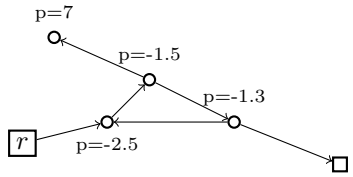
1. $R \subseteq V[S]$,
2. each $v_i \in V[S]$ can be reached from r on a directed path in S,
3. $\sum_{v \in V[S]} p(v)$ is maximized.

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.

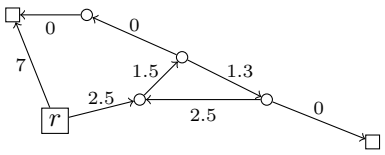
Transformation 2 (Directed RMWCSP to SAP)**.**
Input: *A directed RMWCSP* $P_{RMW} = (V, A, R, p, r)$
Output: *An SAP* $P' = (V', A', T', c', r')$

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 3. 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 3: Illustration of a directed RMWCSP instance with root r (left) and the equivalent SAP obtained by Transformation 2 (right). Terminals are drawn as squares.

Lemma 10 (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* $S' = (V'[S'], A'[S'])$ *to* P' *can be mapped to a solution* $S = (V[S], A[S])$ *to* P_{RMW} *defined by:*

$$V[S] := V \cap V'[S'], \tag{22}$$
$$A[S] := A \cap A'[S'] \tag{23}$$

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

$$p(V[S]) = \sum_{v \in V : p(v) > 0} p(v) - c(A'[S']). \tag{24}$$

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:

$$v(TransCut) := \sum_{v \in V : p(v) > 0} p(v) - {c'}^T x. \tag{25}$$

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 [18]. For an SAP (V, A, T, c, r) this formulation is defined as the Directed Cut Formulation plus the constraints

$$x(\delta^-(v)) \leq x(\delta^+(v)), \qquad \text{for all } v \in V \setminus T \tag{26}$$

$$x(\delta^-(v)) \geq x(a), \qquad \text{for all } a \in \delta^+(v), v \in V \setminus T \tag{27}$$

Empirically, $TransCut$ together with (26) and (27) proves to be considerably (and consistently) stronger than the formulation used by SCIP-Jack [15]. 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 [5] 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 [5] 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 [5]:

Formulation 2. *Node Separator Formulation*

$$max \quad p^T y \tag{28}$$

$$y(N) \geq y(v) \qquad \textit{for all } v \in V \setminus (\{r\} \cup D^+(r)),\ N \in \mathcal{N}(r, v), \tag{29}$$

$$y(v) = 1 \qquad \textit{for all } v \in R, \tag{30}$$

$$y(v) \in \{0,1\} \qquad \textit{for all } v \in V. \tag{31}$$

The Node Separator Formulation is also referred to as Cut_r in [5] and its optimal objective value will be denoted by $v(Cut_r)$.

From (24) 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 two propositions show that $v_{LP}(TransCut) \leq v(Cut_r)$ is always true, but that $v(Cut_r) \leq v_{LP}(TransCut)$ does not in general hold—i.e., the LP relaxation of $TransCut$ is *strictly stronger* [12] than the LP relaxation of Cut_r. Consequently, also the LP relaxation of

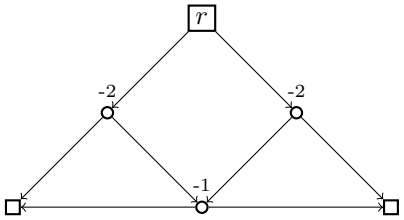


Figure 4: Illustration of a directed RMWCSP instance with vertices R drawn as squares. For this instance $v_{LP}(Cut_r) = -2.5$ and $v_{LP}(TransCut) = v(TransCut) = -3$ holds.

$TransCut$ together with the constraints (26) and (27) is strictly stronger than the LP relaxation of Cut_r

Lemma 11. *The relation $v_{LP}(Cut_r) \leq v_{LP}(TransCut)$ does not in general hold.*

Proof. To prove the proposition, it is sufficient to show that there is a directed RMWCSP instance for which $v_{LP}(Cut_r) > v_{LP}(TransCut)$. Such an instance is depicted in Figure 4. 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 4 in particular $v_{LP}(TransCut) = v(TransCut)$ holds.

Lemma 12. *The relation $v_{LP}(TransCut) \leq v_{LP}(Cut_r)$ is always true.*

The (technical and rather intricate) proof of the proposition can be found in Appendix A.2.

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 [26]. 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 Solving to Optimality

With several solving 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?

4.1 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 [21]. 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 [18]. A modified version of the algorithm has recently

been implemented for SCIP-JACK by the authors of this article and has proved to be on average more than ten times faster than the original used in [18].

Another component instrumental for empirically successful exact solving is constituted by heuristics. Except for dual-ascent, only primal heuristics have been designed and implemented. These include three constructive heuristics. First, a greedy approach has been implemented that is similar to the classical repetitive shortest path heuristic for the Steiner tree problem in graphs [27]. It is used as a subroutine for the other two constructive algorithms and moreover, with adapted vertex weights, during the branch-and-cut whenever a new LP has been solved. Second, the positive-vertex decomposition concept introduced in this article has been extended to a heuristic: The lower bound originally defined by the best-known solution is repeatedly increased such that in each iteration of the heuristic a fixed proportion of edges and vertices is eliminated. Thereupon, all exact reduction 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. A similar algorithm was applied for the Steiner tree problem in graphs in [23] and called *prune* heuristic. The same name will be used for the just described algorithm. Third, the *ascend-reduce* heuristic is based on Transformation 1 or 2 and the dual-ascent heuristic for the SAP (see Section 3). Consider an MWCSP or RMWCSP P and the graph constituted by the undirected edges in P corresponding to paths with reduced cost 0 from the root to all additional terminals in the SAP P' obtained from Transformation 1 or 2. On this subgraph a solution is computed by first employing Reduce-Basic (see Section 2.3) and then executing the prune heuristic and the subsequently introduced local search algorithms. The idea of using the graph obtained by dual-ascent for finding a primal solution goes back to [30], where it was used for the Steiner tree problem in graphs.

Furthermore, three local heuristics have been designed. First, an extension of the greedy construction approach is used. Second, an algorithm that attempts to add neighboring vertices to a given solution to allow the removal of vertices of lower weight has been developed. It works by computing a spanning tree on a given solution and looking for chains of vertices of degree two (in the spanning tree) that can be deleted when adding a new vertex to the solution. The third local heuristic attempts to delete vertices (and edges) from a given solution to increase its weight: Given a solution S, consider the connected subgraph $G_S := (V[S], (V[S] \times V[S]) \cap E)$. Thereupon, the heuristic employs the reduction package Reduce-Basic on G_S and executes several of the above heuristics to find a new solution.

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. Notably, with any of these three components—heuristics, reduction techniques, or new IP formulation—being deactivated, the performance of the solver considerably deteriorates.

4.2 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 [20] 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. Many of these instances proved to be intractable for the solvers participating in the 11th DIMACS Challenge and could only recently be solved to optimality [19]. 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 [4].
SHINY	39	232-3828	202-4494	solved	Real-world instances from network enrichment analysis in computational biology [20].
ACTMOD	8	2034-5226	3335-93394	solved	Real-world instances derived from from integrative biological network analysis [10].
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 [31]. For all computations a time limit of one hour was set.

The results for the first three tests sets reveal a strong performance of both SCIP-Jack 1.0 and the new solver. The latter shows a run time improvement of factor more than three for most instances, solving all instances in less than half a second. However, also SCIP-Jack 1.0 solves all instances within two seconds. The bester other results for these test sets are reported in [19], for ACTMOD and JMPALMK, and in [20], 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 [19] and more than two orders of magnitude faster than the solver described in [20]. Furthermore, the maximum run time in [20] (on the SHINY test set) is more than a thousand seconds, while it is less than 0.1 seconds for the new solver. The computational environment for the experiments was reported as Intel Xeon CPU with 2.5 GHz and 64 GB memory [19] and AMD Opteron 6380 CPU with 2.5 GHz [20], respectively.

The results for the two hardest test sets, HANDBD and HANDBI, are reported in Table 2. 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 1.0, 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 1.0 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 1.0 leaves three instances—those that have also been intractable for the approach in [19]—unsolved, while the new solver reaches optimality for one of them and significantly

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

	SCIP-Jack 1.0 [15]			New Solver		
Instance	Primal	Gap [%]	Time [s]	New Primal	Gap [%]	Time [s]
handbd01	728.963591	–	22.1	–	–	9.8
handbd02	296.496486	–	202.5	–	–	32.2
handbd03	135.070605	–	89.3	–	–	34.8
handbd04	1813.95916	–	340.6	–	–	27.4
handbd05	105.474688	–	91.7	–	–	39.1
handbd06	1528.76544	–	248.9	–	–	39.5
handbd07	77.861959	–	93.3	–	–	38.3
handbd08	1368.16677	–	56.7	–	–	24.3
handbd09	62.71716	–	99.3	–	–	39.2
handbd10	1137.42973	–	33.3	–	–	15.7
handbd11	46.772533	–	96.2	–	–	34.2
handbd12	321.204744	–	75.7	–	–	13.9
handbd13	13.20885	0.82	> 3600.0	13.196987	0.14	> 3600.0
handbd14	4379.10424	–	142.9	–	–	2.8
handbi01	1358.56338	–	20.5	–	–	9.0
handbi02	531.810883	–	77.0	–	–	27.3
handbi03	243.134201	–	81.4	–	–	37.3
handbi04	3202.18574	0.02	> 3600.0	3202.18574	–	46.7
handbi05	184.467331	–	75.9	–	–	33.6
handbi06	2921.54472	–	139.5	–	–	36.5
handbi07	150.974258	–	70.9	–	–	41.5
handbi08	2270.28462	–	38.8	–	–	21.0
handbi09	107.768806	–	88.4	–	–	40.5
handbi10	1874.29296	–	17.3	–	–	11.3
handbi11	68.944709	–	74.6	–	–	33.1
handbi12	138.257023	–	211.9	–	–	8.8
handbi13	4.268146	1.60	> 3600.0	4.251	0.13	> 3600.0
handbi14	7881.76874	–	228.7	–	–	2.9

Table 2: Computational comparison of SCIP-Jack 1.0 and the new MWCSP solver.

reduces the primal-dual gap for the other two. The previously best known gaps [19] 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 first LP-solve—further experiments showed that CPLEX takes more than a day to only solve the first LP.

Furthermore, as compared to SCIP-Jack 1.0 also the run times are notably reduced—all instances are solved faster, and all but three more than twice as fast. A prominent example for the improved run time is the instance handbi14 for which the difference in run time is almost two orders of magnitude.

Table 3 provides further information on preprocessing for the three hardest instances handbd13, handbi04 and handbi13. In the columns two and three the number of vertices and edges after the preprocessing of SCIP-Jack 1.0 is given. Column four shows the time SCIP-Jack 1.0 spent in preprocessing. The next three columns provide the corresponding information for the new solver. Finally, the last column signifies whether the new solver could transform the instance to RMWCSP.

The results show that both the strength of the reductions and the preprocessing time have significantly improved with the new solver. Furthermore, for two of the three instances Transformation 2 could be applied, which allowed further reductions in preprocessing for both handbi04 and handbi13 and significantly sped up the main solving process for handbi04. Overall, the results for the three hard instances exemplify the interplay of the new reduction techniques,

	SCIP-Jack 1.0 [15]			New Solver			
Instance	$\|V'\|$	$\|E'\|$	Time [s]	$\|V'\|$	$\|E'\|$	Time [s]	RMWCSP
handbd13	165990	330502	536.1	122868	243897	374.5	no
handbi04	16647	63528	97.2	9723	16115	32.2	yes
handbi13	158151	315094	403.1	123319	243278	273.9	yes

Table 3: Presolving statistics of hard instance for SCIP-Jack 1.0 and the new solver. The nodes and edges of the presolved problem are denoted by V' and E'. Column RMWCSP signifies whether Transformation 2 could be applied.

heuristics and the new formulation and its practical success.

5 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 less than half 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.

6 Acknowledgements

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

This work was supported by the BMWi project *Realisierung von Beschleunigungsstrategien der anwendungsorientierten Mathematik und Informatik für optimierende Energiesystemmodelle - BEAM-ME* (fund number 03ET4023DE). This work was supported by DFG in the framework of the Collaborative Research Centre CRC/Transregio

154, *Mathematical Modelling, Simulation and Optimization Using the Example of Gas Networks*. This work was supported by the ICT COST Action TD1207 *Mathematical Optimization in the Decision Support Systems for Efficient and Robust Energy Networks*. The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

References

[1] 11th DIMACS Challenge. `http://dimacs11.zib.de/`. Accessed: August 10. 2017.

[2] 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.

[3] Ernst Althaus and Markus Blumenstock. Algorithms for the Maximum Weight Connected Subgraph and Prize-collecting Steiner Tree Problems. Unpublished manuscript at http://dimacs11.cs.princeton.edu/workshop.html, 2014.

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] Marcus T. Dittrich, Gunnar W. Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. In *ISMB*, pages 223–231, 2008.

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

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

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

[14] Matteo Fischetti, Markus Leitner, Ivana 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*, pages 1–27, 2016.

[15] 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.

[16] 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.

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

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

[19] M. Leitner, M. Luipersbeck, I. Ljubic, and M. Sinnl. A dual-ascent-based branch-and-bound framework for the prize-collecting Steiner tree and related problems. Technical report, Department of Statistics and Operations Research, 2016.

[20] 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.

[21] 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.

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

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

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

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

[26] 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.

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

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

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

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

[31] 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 $v_i \in V[S]$. Denote the (unique) path in S between v_i and a $t_j \in V[S] \cap T$ by Q_j and the set of all such paths by $\mathcal{Q}$. First, note that $|\mathcal{Q}| \geq 2$, because if $\mathcal{Q}$ just contained one path, say Q_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 $Q_l \in \mathcal{Q}$ such that their combined number of H-boundary edges is minimal and $V[Q_k] \cap V[Q_l] = \{v_i\}$ holds. Further, define $\mathcal{Q}^- := \mathcal{Q} \setminus \{Q_k, Q_l\}$. For all $Q_r \in \mathcal{Q}^-$, denote by Q_r' the subpath of Q_r 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 cost, one can obtain an upper bound on the cost of S by:

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

Consequently, the proposition is proven. □

A.2 Proof of Proposition 12

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

$$A'_q := (\bigcup_{j=q+1}^{u+1} \delta^-(W_j)) \ \cap \ (\delta^-(X_q) \cup A_q \cup \Delta_q). \tag{32}$$

and

$$\overline{A'}_q := (\bigcup_{j=1}^{q} A'_j) \ \cap \ (\bigcup_{j=q+1}^{u+1} \delta^-(W_j)). \tag{33}$$

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

$$x(\overline{A'}_q) \geq x(\bigcup_{j=1}^{q} \Delta_j) \tag{34}$$

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

$$x(\delta^-(W_q)) \leq \delta^-(X_q \dot\cup W_q) = (\delta^-(W_q) \setminus \delta^+(X_q)) \dot\cup (\delta^-(X_q) \setminus \delta^+(W_q)). \tag{35}$$

This equation implies that

$$x(\delta^+(X_q) \cap \delta^-(W_q)) \leq x(\delta^-(X_q) \setminus \delta^+(W_q)) \tag{36}$$

$$\leq x(\delta^-(X_q) \cap \delta^-(\bigcup_{j=q+1}^{u+1} W_j)), \tag{37}$$

where (37) 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

$$x(A_q) + x(\Delta_q) = x(A_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) + x(A_q \cap \delta^+(X_q)) + x(\Delta_q) \tag{38}$$

$$\leq x(A_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) + x(\delta^-(X_q) \cap \delta^-(\bigcup_{j=q+1}^{u+1} W_j)) + x(\Delta_q) \tag{39}$$

$$= x(A'_q). \tag{40}$$

Equality (38) follows from the construction of X_q, and inequality (39) follows from $A_q \subseteq \delta^-(W_q)$ and from the inequalities (36)-(37). Finally, equality (40) 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 (33), one obtains that

$$x(\overline{A'}_q) = x((\bigcup_{j=1}^{q-1} A'_j \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j)) \ \cup \ (A'_q \cap \bigcup_{j=q+1}^{u+1} \delta^-(W_j))) \tag{41}$$

$$\geq x(\overline{A'}_{q-1} \setminus (\delta^-(W_q) \cap \overline{A'}_{q-1}) \cup A'_q) \tag{42}$$

$$= x((\overline{A'}_{q-1} \setminus A_q) \cup A'_q) \tag{43}$$

$$\geq x(\overline{A'}_{q-1}) - x(A_q) + x(A'_q) \tag{44}$$

$$\geq x(\overline{A'}_{q-1}) + x(\Delta_q). \tag{45}$$

First, equality (41) follows from the definition of $\overline{A'}_q$. Inequality (42) follows from the definition of $\overline{A'}_q$ and A'_q. Similarly, equality (43) follows from the definition of A_q. Inequality (44) follows from $\overline{A'}_{q-1} \cap A'_q \subseteq A_q$. Finally, inequality (45) follows from the system (38)-(40).

Consequently, by an induction argument it can be concluded that invariant (34) 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

$$x\big(\delta^-(S_{v_i})\big) \geq x(\bigcup_{j=1}^{u} \Delta_j) = x(\Delta). \tag{46}$$

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

$$y(N) \geq x(\Delta) = y(v_i). \tag{47}$$

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. □