Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

DANIEL REHFELDT, THORSTEN KOCH

Generalized preprocessing techniques for Steiner tree and maximum-weight connected subgraph problems

ZIB Report 17-57 (June 2020)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +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

Generalized preprocessing techniques for Steiner tree and maximum-weight connected subgraph problems

Daniel Rehfeldt, Thorsten Koch

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

TU Berlin, Chair of Software and Algorithms for Discrete Optimization,
Str. des 17. Juni 135, 10623 Berlin, Germany

June 15, 2020 (revised version)

Abstract

This article introduces new reduction techniques for the Steiner tree problem in graphs (SPG) and one of its most popular relatives, the maximum-weight connected subgraph problem. Several of the techniques generalize previous results from the literature. In particular, we introduce a generalization of the Steiner bottleneck distance—the arguably most important reduction concept for SPG. While several methods require to solve NP-hard problems, relaxations allow for a strong practical efficiency of these techniques. Initial computational tests with the exact Steiner tree solver SCIP-Jack show a significant improvement of the preprocessing strength.

1 Introduction

The Steiner tree problem in graphs (SPG) is one of the most studied problems in combinatorial optimization. Furthermore, the SPG is among the classic $\mathcal{NP}$-hard problems [8]. Also, many relatives of the SPG have been extensively discussed in the literature, often propelled by practical applications. One of the most popular of these is the *maximum-weight connected subgraph problem* (MWCSP). For both SPG and MWCSP sophisticated exact solvers exist, see e.g. [5, 11, 14]. An essential component of all these approaches is preprocessing. This article introduces a series of new preprocessing technique for SPG and MWCSP (also commonly called *reduction techniques*) that are provably stronger than previous ones.

Preliminary computational tests with the exact Steiner tree solver SCIP-Jack show a significant improvement of the reduction strength. If integrated into the branch-and-cut algorithm of SCIP-Jack, the run-time for exact solution can also be notably improved. E.g., many non-trivial SPG instances from the literature can be solved more than twice as fast.

1.1 Notation

For both SPG and MWCSP we denote the underlying graph by $G := (V, E)$, with vertices V and (undirected) edges E. We set $n := |V|$ and $m := |E|$. While for the SPG T denotes the set of terminals, for the MWCSP we define $T := \{v \in V \mid p(v) > 0\}$. Furthermore, we use the notation $T = \{t_1, ..., t_s\}$ with $s := |T|$. For any subgraph $S \subseteq G$ (e.g., a Steiner tree) we denote its vertices by $V(S)$ and its edges by $E(S)$ (note the difference to the notation $E[U]$ defined in (1) for a set $U \subseteq V$ of vertices). For a walk W we likewise denote the set of vertices and the set of edges it contains by $V(W)$ and $E(W)$. For any $U \subseteq V$ we define

$$E[U] := \big\{\{v, w\} \in E \mid v, w \in U\big\}. \tag{1}$$

For $U \subseteq V$ define the induced *edge cut* as $\delta(U) := \{\{u, v\} \in E \mid u \in U, v \in V \setminus U\}$. We also write δ_G to distinguish the underlying graph. For a single vertex v we use the short-hand notation $\delta(v) := \delta(\{v\})$. We define the *neighborhood* of any $v \in V$ as

$$N(v) := \big\{w \in V \mid \{v, w\} \in E\big\}. \tag{2}$$

Given edge costs $c : E \mapsto \mathbb{Q}_{\geq 0}$, the triplet (V, E, c) is referred to as *network*. By $d(v, w)$ we denote the cost of a shortest path (with respect to c) between vertices $v, w \in V$. For any (distance) function $\tilde{d} : \binom{V}{2} \mapsto \mathbb{Q}_{\geq 0}$, and any $U \subseteq V$ we define the *$\tilde{d}$-distance graph* on U as the network

$$D_G(U, \tilde{d}) := (U, \binom{U}{2}, \tilde{c}), \tag{3}$$

with $\tilde{c}(\{v, w\}) := \tilde{d}(v, w)$ for all $v, w \in U$. If $\tilde{d}$ is the standard distance (i.e. $\tilde{d} = d$), we write $D_G(U)$ instead of $D_G(U, d)$. If a given $\tilde{d} : \binom{V}{2} \mapsto \mathbb{Q}_{\geq 0}$ is symmetric, we occasionally write, with a slight abuse of notation, $\tilde{d}(e)$ instead of $\tilde{d}(v, w)$ for an edge $e = \{v, w\}$.

2 Preprocessing for Steiner tree problems in graphs

Given an undirected, connected graph $G = (V, E)$, costs (or weights) $c : E \to \mathbb{Q}_{\geq 0}$ and a set $T \subseteq V$ of *terminals*, the Steiner tree problem in graphs (SPG) asks for a tree $S \subseteq G$ such that

1. $T \subseteq V(S)$ holds,
2. $c(E(S))$ is minimized.

A tree that satisfies condition 1 is called *Steiner tree*; a tree that additionally satisfies condition 2 is called *minimum Steiner tree*. The sum in 2 is called the *weight* or *cost* of the Steiner tree S. The vertices in $V \setminus T$ are referred to as *Steiner nodes*.

The SPG is a classic optimization problem, being the subject of hundreds of research articles, and can also be found in real-world applications (although applications for variations of the SPG are far more prevalent [5]). The by far strongest preprocessing techniques for the SPG are described in [12, 13], which also form the basis of the until today strongest exact SPG solver. This section generalizes some of these techniques and suggests new ones. In the following it will be assumed that an SPG denoted by $I_{SPG} = (V, E, T, c)$ is given.

2.1 Bottleneck distance based techniques

In the context of Steiner tree problems, alternative-based reduction methods attempt to prove that a specified part of the problem graph is not contained in at least one optimal solution [2].

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. This section describes new alternative based reduction techniques. In particular, it introduces a generalization of the most important distance concept for alternative-based reductions.

2.1.1 The bottleneck Steiner distance

Let P be a simple path with at least one edge. The *bottleneck length* [4] of P is

$$bl(P) := \max_{e \in E(P)} c(e). \tag{4}$$

Let $v, w \in V$. Let $\mathcal{P}(v, w)$ be the set of all simple paths between v and w. The *bottleneck distance* [4] between v and w is defined as

$$b(v, w) := \inf\{bl(P) \mid P \in \mathcal{P}(v, w)\}, \tag{5}$$

with the common convention that $\inf \emptyset = \infty$. Note that $b(v, w)$ is equal to the bottleneck length of the path between v and w on any minimum spanning tree of (G, c) [1].

Now consider the distance graph $D := D_G(T \cup \{v, w\})$. Let b_D be the bottleneck distance in D. Define the *bottleneck Steiner distance* or *special distance* [4] between v and w as

$$s(v, w) := b_D(v, w). \tag{6}$$

The bottleneck Steiner distance is arguably the most important reduction concept for SPG, with various applications. The arguably best known one is the following criterion, which allows for edge deletion [4].

Theorem 1. *Let $e = \{v, w\} \in E$. If $s(v, w) < c(e)$, then no minimum Steiner tree contains e.*

Note that bottleneck Steiner distances can be computed in polynomial time, but in practice (heuristic) approximations are used. See [12] for a state-of-the-art algorithm.

2.1.2 A stronger bottleneck concept

In the following we describe a generalization of the bottleneck Steiner distance. Initially, for an edge $e = \{v, w\}$ define the *restricted bottleneck distance* $\bar{b}(e)$ [12] as the bottleneck distance between v and w on $(V, E \setminus \{e\}, c)$.

The basis of the new bottleneck Steiner concept is formed by a node-weight function that we introduce in the following. For any $v \in V \setminus T$ and $F \subseteq \delta(v)$ define

$$p^+(v, F) = \max\big\{0, \sup\{\bar{b}(e) - c(e) \mid e \in \delta(v) \cap F, e \cap T \neq \emptyset\}\big\}. \tag{7}$$

We call $p^+(v, F)$ the *F-implied profit* of v. Note that $p^+(v, e) = \infty$ for an $e \in \delta(v)$ if and only if all Steiner trees contain e. The following observation motivates the subsequent usage of the implied profit. Assume that $p^+(v, \{e\}) > 0$ for an edge $e \in \delta(v)$. If a Steiner tree S contains v, but not e, then there is a Steiner tree S' with $e \in E(S')$ such that $c(E(S')) + p^+(v, \{e\}) \leq c(E(S))$.

Let $v, w \in V$. Consider a finite walk $W = (v_1, e_1, v_2, e_2, ..., e_r, v_r)$ with $v_1 = v$ and $v_r = w$. We say that W is a (v, w)-walk. For any $k, l \in \mathbb{N}$ with $1 \leq k \leq l \leq r$ define the subwalk $W(k, l) := (v_k, e_k, v_{k+1}, e_{k+1}, ..., e_l, v_l)$. W will be called *Steiner walk* if $V(W) \cap T \subseteq \{v, w\}$ and v, w are contained exactly once in W (the latter condition could be omitted, but has been added for ease of presentation). The set of all Steiner walks from v to w will be denoted by $\mathcal{W}_T(v, w)$.

With a slight abuse of notation we define $\delta_W(u) := \delta(u) \cap E(W)$ for any walk W and any $u \in V$. Define the *implied Steiner cost* of a Steiner walk $W \in \mathcal{W}_T(v, w)$ as

$$c_p^+(W) := \sum_{e \in E(W)} c(e) - \sum_{u \in V(W) \setminus \{v,w\}} p^+\big(u, \delta(u) \setminus \delta_W(u)\big). \tag{8}$$

Further, set

$$P_W^+ := \{u \in V(W) \mid p^+\big(u, \delta(u) \setminus \delta_W(u)\big) > 0\} \cup \{v, w\}. \tag{9}$$

Define the *implied Steiner length* of W as

$$l_p^+(W) := \max\{c_p^+(W(v_k, v_l)) \mid 1 \leq k \leq l \leq r,\ v_k, v_l \in P_W^+\}. \tag{10}$$

Define the *implied Steiner distance* between v and w as

$$d_p^+(v, w) := \min\{l_p^+(W) \mid W \in \mathcal{W}_T(v, w)\}. \tag{11}$$

Note that $d_p^+(v, w) = d_p^+(w, v)$. At last, consider the distance graph $D^+ := D_G(T \cup \{v, w\}, d_p^+)$. Let b_{D^+} be the bottleneck distance in D^+. Define the *implied bottleneck Steiner distance* between v and w as

$$s_p(v, w) := b_{D^+}(v, w). \tag{12}$$

Note that $s_p(v, w) \leq s(v, w)$ and that the inequality can be strict. Indeed, $\frac{s(v,w)}{s_p(v,w)}$ can become arbitrarily large. Thus, the following result is a stronger analogue to Theorem 1.

Theorem 2. *Let $e = \{v, w\} \in E$. If $s_p(v, w) < c(e)$, then no minimum Steiner tree contains e.*

Proof. Assume $s_p(v, w) < c(e)$ and let S be a Steiner tree with $e \in E(S)$. We will show the existence of a Steiner tree S' with $e \notin E(S')$ such that $c(E(S')) \leq c(E(S))$, which concludes the proof. First, remove e from S to obtain a new subgraph $\tilde{S}$, which consists of exactly two connected components. Assume that each connected component contains at least one terminal (otherwise the proof is already finished). Consider a (v, w)-path P in D^+ such that $bl_{D^+}(P) = b_{D^+}(v, w)$. Let $\{t, u\}$ be an edge on P such that t and u are in different connected components of $\tilde{S}$ (where t and u are considered in the original SPG). Let $\tilde{S}^t$ and $\tilde{S}^u$ be the connected components of $\tilde{S}$ such that $t \in V(\tilde{S}^t)$ and $u \in V(\tilde{S}^u)$. By the definition of the bottleneck length it holds that

$$d_p^+(t, u) \leq s_p(v, w). \tag{13}$$

Let $W \in \mathcal{W}_T(t, u)$ such that

$$l_p^+(W) = d_p^+(t, u). \tag{14}$$

Assume that W is given as $W = (v_1, e_1, ..., e_r, v_r)$. Define $b := \min\{k \in \{1, ..., r\} \mid v_k \in V(\tilde{S}^u)\}$ and $a := \max\{k \in \{1, ..., b\} \mid v_k \in V(\tilde{S}^t)\}$. Further, define $x := \max\{k \in \{1, ..., a\} \mid v_k \in P_W\}$ and $y := \min\{k \in \{b, ..., r\} \mid v_k \in P_W\}$. By definition, $x \leq a < b \leq y$ and furthermore:

$$\sum_{e \in E(W(a,b))} c(e) - \sum_{v \in V(W(a,b)) \setminus \{v_x, v_y\}} p^+\big(v, \delta(v) \setminus \delta_{W(x,y)}\big) \leq c_p^+(W(x, y)). \tag{15}$$

Reconnect $\tilde{S}^t$ and $\tilde{S}^u$ by $W(a, b)$, which yields a connected subgraph S_0' with $T \subseteq V(S_0')$. Assume that S_0' is a tree (otherwise remove any redundant edges).[1] It holds that

$$\sum_{e \in E(S_0')} c(e) \leq \sum_{e \in E(S)} c(e) + \sum_{e \in E(W(a,b))} c(e) - c(\{v, w\}). \tag{16}$$

[1] Because we assume all edges to be of positive cost, S_0' will in fact always be a tree.

Let $v_1^+, v_2^+, ..., v_z^+$ be the vertices in $P_{W(a,b)} \setminus \{v_a, v_b\}$. Choose for each $i = 1, ..., z$ an edge $e_i^+ \in \delta(v_i^+) \setminus \delta_{W(x,y)}(v_i^+)$ such that $e_i^+ \cap T \neq \emptyset$ and

$$\overline{b}(e_i^+) - c(e_i^+) = p^+(v, \delta(v) \setminus \delta_{W(x,y)}). \tag{17}$$

Note that all e_i^+ are pairwise disjoint (just as the v_i^+).

We will construct Steiner trees S_i' for $i \in \{1, ..., z\}$ that satisfy

$$\sum_{e \in E(S_i')} c(e) \leq \sum_{e \in E(S_0')} c(e) - \sum_{k=1}^{i} p^+(v_k^+, \delta(v) \setminus \delta_{W(x,y)}). \tag{18}$$

and

$$\bigcup_{k=i+1}^{z} \{e_k^+\} \cap E(S_i') = \emptyset, \tag{19}$$

and

$$V(S_i') = V(S_0'). \tag{20}$$

One readily verifies that S_0' satisfies (18)-(20). Let $i \in \{1, ..., z\}$ and assume that (18)-(20) hold for S_{i-1}'. Thus, $e_i^+ \notin E(S_{i-1}')$. Let P_i be the (unique) path in S_{i-1}' between v_i^+ and the terminal t_i with $\{t_i\} = e_i^+ \cap T$. Choose any $\tilde{e}_i \in E(P)$ with $c(\tilde{e}_i) = bl(P_i)$. Define the tree S_i' by $V(S_i') := V(S_{i-1}')$ and $E(S_i') := \big(E(S_{i-1}') \setminus \{\tilde{e}_i\}\big) \cup \{e_i^+\}$. We claim that S_i' satisfies (18)-(20). Equality (19) follows from the fact that all e_i^+ are disjoint. And (20) follows from the construction of S_i'. For (18), observe that by definition of the bottleneck distance it holds that $c(\tilde{e}_i) \geq \overline{b}(e_i^+)$ and therefore

$$\overline{b}(e_i^+) - c(e_i^+) \leq c(\tilde{e}_i) - c(e_i^+). \tag{21}$$

Thus, equation (17) implies that S_i' satisfies (18).

Finally, set $S' := S_z'$. Because of (20) it holds that $T \subseteq V(S')$. Furthermore, one obtains:

$$\sum_{e \in E(S')} c(e) \overset{(18)}{\leq} \sum_{e \in E(S_0')} c(e) - \sum_{k=1}^{z} p^+(v_k^+, \delta(v_k^+) \setminus \delta_{W(x,y)}) \tag{22}$$

$$\overset{(16)}{\leq} \sum_{e \in E(S)} c(e) + \sum_{e \in E(W(a,b))} c(e) - c(\{v,w\}) - \sum_{k=1}^{z} p^+(v_k^+, \delta(v_k^+) \setminus \delta_{W(x,y)}) \tag{23}$$

$$\overset{(15)}{\leq} \sum_{e \in E(S)} c(e) - c(\{v,w\}) + c_p^+(W(x,y)) \tag{24}$$

$$\overset{(14)}{\leq} \sum_{e \in E(S)} c(e) - c(\{v,w\}) + l_p^+(W) \tag{25}$$

$$\overset{(13)}{\leq} \sum_{e \in E(S)} c(e) - c(\{v,w\}) + s_p(v,w) \tag{26}$$

$$\leq \sum_{e \in E(S)} c(e), \tag{27}$$

where the last inequality follows from the initial assumptions. □

Furthermore, we define the *restricted implied bottleneck Steiner distance* $\overline{s}_p(v,w)$ between any $v, w \in V$ as the implied bottleneck Steiner distance between v and w in the SPG $(V, E \setminus \big\{\{v,w\}\big\}, c)$. One obtains the following corollary.

Corollary 3. *Let $e = \{v, w\} \in E$. If $\bar{s}_p(v, w) \leq c(e)$, then at least one minimum Steiner tree does not contain e.*

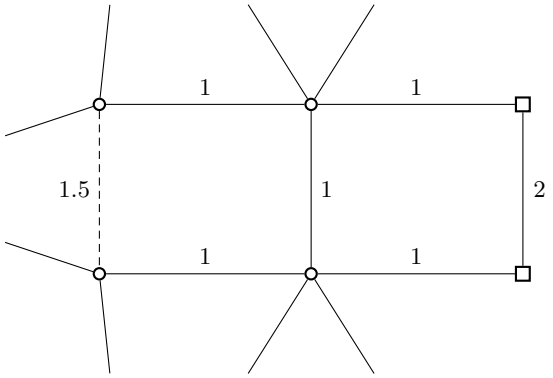


Figure 1: Segment of a Steiner tree instance. Terminals are drawn as squares. The dashed edge can be deleted by employing Theorem 2.

Figure 1 shows a segment of an SPG instance for which Theorem 2 allows for the deletion of an edge, but Theorem 1 does not. The implied bottleneck Steiner distance between the endpoints of the dashed edge is 1—the edge can thus be deleted. In contrast, the (standard) bottleneck Steiner distance between the endpoints is 1.5 (corresponding to the edge itself). In a sense, the implied bottleneck Steiner distance can also be seen as a generalization of the walk-based distance concept for the prize-collecting Steiner tree problem recently introduced by the authors [16]. Thus, it does not come as a surprise that already computing the implied Steiner distance is hard, as the following proposition shows.

Proposition 4. *Computing the implied Steiner distance is $\mathcal{NP}$-hard.*

Nevertheless, one can readily devise heuristics that provide upper bounds on s_p that are at least as strong as those used for s, and that are empirically often stronger.

2.1.3 Replacement techniques

This section starts with a reduction criterion based on the standard bottleneck Steiner distance. Besides being a new technique, this result also serves to highlight the complications that arise if one attempts to formulate similar conditions based on the implied bottleneck Steiner distance.

Proposition 5. *Let $D := D_G(T, d)$. Let Y be a minimum spanning tree in D. Write its edges $\{e_1^Y, e_2^Y, ..., e_{|T|-1}^Y\} := E(Y)$ in non-ascending order with respect to their weight in D. Let $v \in V \setminus T$. If for all $\Delta \subseteq \delta(v)$ with $|\Delta| \geq 3$ it holds that:*

$$\sum_{i=1}^{|\Delta|-1} d(e_i^Y) \leq \sum_{e \in \Delta} c(e), \tag{28}$$

then there is at least one minimum Steiner tree S such that $|\delta_S(v)| \leq 2$.

If the conditions (28) are satisfied for a vertex $v \in V \setminus T$ one can *pseudo-eliminate* [3] v, i.e., delete v and connect any two vertices $u, w \in N(v)$ by a new edge $\{u, w\}$ of weight $c(\{v, u\}) + c(\{v, w\})$.

The SPG depicted in Figure 2 exemplifies why Proposition 5 cannot be formulated by using the implied Steiner distance. The weight of the minimum spanning tree Y for $D_G(T, d)$ is 4, but the weight of a minimum spanning tree with respect to the implied bottleneck Steiner distance is 2. Similarly also the BD_m reduction technique [3] cannot be directly formulated by using the implied bottleneck distance. Still, it is possible to formulate a similar criterion that makes use of the implied bottleneck distance. Unfortunately, both the result and the corresponding proof are rather involved. Thus, we omit the details here. The important point is to make sure that the selected Steiner walks do not overlap at vertices with a positive implied profit.

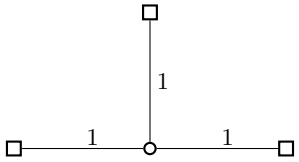


Figure 2: SPG instance. Terminals are drawn as squares

2.2 Bound-based techniques

Bound-based reductions techniques are preprocessing methods that identify edges and vertices for elimination by examining whether they induce an lower bound that exceeds a given upper bound [12, 17]. In this section a bound-based reduction concept is introduced that generalizes the Voronoi-regions concept from [12]. Note that the bounding technique described in this section can be seen as a special case of the bound-based reduction technique for the prize-collecting Steiner tree problem that have been recently published by the authors in [16]. Thus, no proofs are provided for any of the results in this section.

The base of the reduction technique is the following new concept: a *terminal-regions decomposition* of I_{SPG}—with underlying graph (V, E)—is a partition $H = \{H_t \subseteq V \mid T \cap H_t = \{t\}\}$ of V such that for each $t \in T$ the subgraph $(H_t, E[H_t])$ is connected. Each of the H_t will be called a *region* of H. Define for all $t \in T$

$$r_H(t) := \min\{d(t, v) \mid v \notin H_t\}. \tag{29}$$

In [12] a special terminal-regions decomposition called *Voronoi-regions decomposition* is used. The more general results presented here allow to improve on the Voronoi preprocessing methods introduced in [12]. However, it will also turn out that finding an optimal terminal-regions decomposition is $\mathcal{NP}$-hard. The following three propositions not only improve on the results from [12] by using a more general decomposition, but also by making use of the following distance function. Given vertices $v_i, v_j \in V$ define $\underline{d}(v_i, v_j)$ as the length of a shortest path between v_i and v_j without intermediary terminals. In [2] an $O(m + n \log n)$ algorithm was introduced to compute for each non-terminal v_i a constant number of $\underline{d}$-nearest terminals $v_{i,1}, v_{i,2}, ..., v_{i,k}$ (if existent) along with the corresponding paths. In the remainder of this section it will be assumed that a terminal-regions decomposition H is given. Moreover, for ease of presentation it will be assumed that the terminals of I_{SPG} are ordered such that $r_H(t_1) \leq r_H(t_2) \leq ... \leq r_H(t_k)$ with $k := |T|$. The following three propositions can be proved similarly to the Voronoi reduction techniques from [12].

Proposition 1. *Let $v_i \in V \setminus T$ and set $k := |T|$. If there is a minimum Steiner tree S such that $v_i \in V(S)$, then*

$$\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \sum_{q=1}^{k-2} r_H(t_q) \tag{30}$$

is a lower bound on the weight of S.

Each vertex $v_i \in V \setminus T$ such that the affiliated lower bound stated in Proposition 1 exceeds a known upper bound can be eliminated. Moreover, if a solution S corresponding to the upper bound is given and v_i is not contained in it, the latter can already be eliminated if the lower bound stated in Proposition 1 is equal to the cost of S. A similar proposition holds for edges in a minimum Steiner tree:

Proposition 2. *Let $\{v_i, v_j\} \in E$ and set $k := |T|$. If there is minimum Steiner tree S such that $\{v_i, v_j\} \in E(S)$, then L defined by*

$$L := c(\{v_i, v_j\}) + \underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,1}) + \sum_{q=1}^{k-2} r_H(t_q) \tag{31}$$

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

$$L := c(\{v_i, v_j\}) + \min\{\underline{d}(v_i, v_{i,1}) + \underline{d}(v_j, v_{j,2}), \underline{d}(v_i, v_{i,2}) + \underline{d}(v_j, v_{j,1}\} + \sum_{q=1}^{k-2} r_H(t_q) \tag{32}$$

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

The following proposition allows to pseudo-eliminate [2] vertices, i.e., to delete a vertex and connect all its adjacent vertices by new edges.

Proposition 3. *Let $v_i \in V \setminus T$. If there is a minimum Steiner tree S such that $\delta_S(v_i) \geq 3$, then*

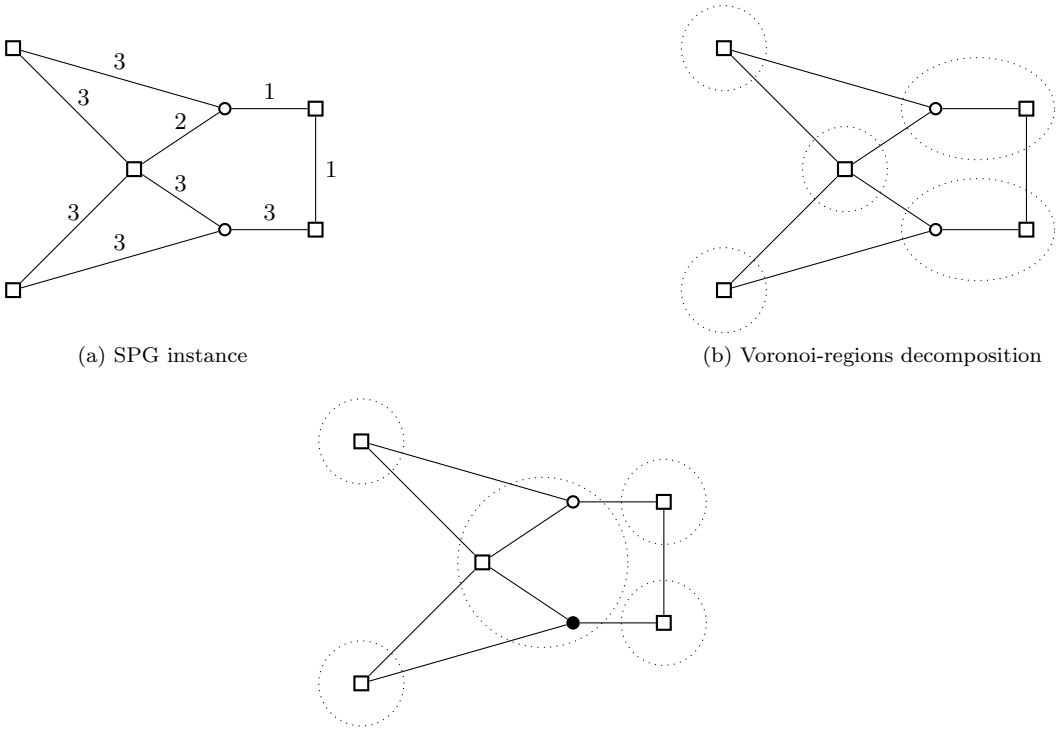
$$\underline{d}(v_i, v_{i,1}) + \underline{d}(v_i, v_{i,2}) + \underline{d}(v_i, v_{i,3}) + \sum_{q=1}^{k-3} r_H(t_q) \tag{33}$$

with $k := |T|$ is a lower bound on the weight of S.

To efficiently apply Proposition 1, one would like to maximize (30)—and for Proposition 2 and Proposition 3 to minimize (31) and (33), 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, connected graph with edge cost $c : E \to \mathbb{N}$. Furthermore, set $T_0 := \{v \in V_0 \mid p(v) > 0\}$, and assume that $\alpha < |T_0|$. For each terminal-regions decomposition H_0 of G_0 define $T_0' \subsetneq T_0$ such that $|T_0'| = \alpha$ and $r_{H_0}(t') \geq r_{H_0}(t)$ for all $t' \in T_0'$ and $t \in T_0 \setminus T_0'$. Let:

$$C_{H_0} := \sum_{t \in T_0 \setminus T_0'} r_{H_0}(t). \tag{34}$$

We now define the α terminal-regions decomposition problem as follows: Given a $k \in \mathbb{N}$, is there a terminal-regions decomposition H_0 such that $C_{H_0} \geq k$? In the following proposition it is shown that this problem is $\mathcal{NP}$-complete, which forthwith establishes the $\mathcal{NP}$-hardness of finding a terminal-regions decomposition that minimizes (30), (31), (32), or (33)—which corresponds to $\alpha = 2$ and $\alpha = 3$, respectively.



(a) SPG instance

(b) Voronoi-regions decomposition

(c) terminal-regions decomposition

Figure 3: Illustration of a Steiner tree instance (a), a Voronoi-decomposition (b), and a second terminal-regions decomposition (c). Terminals are drawn as squares. If an upperbound less than 11 is known, the vertex drawn filled in (c) can be deleted by means of the terminal-regions decomposition depicted in (c), but not by means of the Voronoi-regions decomposition.

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

Figure 3 depicts an SPG, a corresponding Voronoi-regions decomposition as described in [12], and an alternative terminal-regions decomposition. The second terminal-regions decomposition yields a stronger lower bound than the Voronoi-regions decomposition and indeed allows to eliminate a vertex if an upper bound that is sufficiently close to the optimal solution value is known. Computational experiments for this article have shown that it is in most cases easily possible to improve on the bound provided by the Voronoi-regions decomposition and allow for significantly stronger graph reductions.

2.3 Preliminary computational results

This section provides computational results for new implied bottleneck Steiner distance tests. The results should be regarded as preliminary, since the development and implementation of the reduction methods is still work in progress. We compare against a development version of the SCIP-Jack solver [5]. Note that this version of SCIP-Jack is significantly stronger than the latest released one [7].

We use four benchmark sets from the literature; two from [10], and two from the SteinLib [9]. Table 1 shows in the first column the name of the test set, followed by its number of instances.

The next columns show the percentual average number of nodes and edges of the instances after the preprocessing of SCIP-Jack and after the additional use of the new methods. We also report the percentual relative change for the instances that are not already solved to optimality by the SCIP-Jack preprocessing. Roughly speaking, the relative change shows the additional impact of the new reduction techniques relative to the size of the already reduced instances. More specifically, the relative change with respect to either edges or vertices for k instances is defined as follows. Let $p_i \in (0,1]$ (SCIP-Jack) and $q_i \in [0,1]$ (new) be the relative size of instance i after preprocessing, with $i = 1, ..., k$. E.g., if m_i is the number of edges before and m'_i the number of edges after presolving with SCIP-Jack, then the relative edge size of instance i is $\frac{m'_i}{m_i}$. The relative change on all k instances (either with respect to the edges or the vertices) is given as

$$\frac{1}{k} \sum_{i=1}^{k} \frac{q_i - p_i}{\max\{p_i, q_i\}}.$$

While the overall run-time is not reported, we note that the run time of the new tests in their current form is neglectable. In fact, the overall run-time of the preprocessing even decreases, since the tests are called early in the process. As they substantially reduce the instances already, the run-time of the subsequently executed preprocessing methods is notably smaller.

Table 1: Average remaining nodes and edges after preprocessing.

		SCIP-Jack		+new techniques		relative change	
Test set	#	nodes [%]	edges [%]	nodes [%]	edges [%]	nodes [%]	edges [%]
vienna-s	85	14.6	14.2	12.2	11.8	**-18.9**	**-19.4**
WRP3	63	54.5	56.1	52.6	53.2	**-7.1**	**-8.7**
GEO-org	23	10.6	12.3	10.1	11.5	**-6.1**	**-7.0**
ES1000FST	15	52.1	57.7	43.9	49.0	**-15.9**	**-15.1**

It can be seen that the average instance size on all test sets is notably decreased. This reduction is quite significant, given the large number of (often far more time-consuming) preprocessing tests already present in SCIP-Jack, including a subset of the highly intricate extending reduction techniques [13]. Note that the all test sets come already in a preprocessed form—the impact of these original reductions is not given here.

We also note that the new reduction techniques significantly reduce the overall run time on the above benchmark sets. E.g. for for more than 80 % of the instances from the vienna-s set the run time is at least halved. On this test set the new run times are now comparable with those of the solver from [12] (see also [14] for the latest results), which has remained out of reach for any other SPG solver from the literature until now. The WRP3 test set can even be solved significantly faster than by [12].

3 Preprocessing for maximum-weight connected subgraph problems[2]

Given an undirected graph $G = (V, E)$ and vertex weights $p : V \to \mathbb{R}$, the maximum-weight connected subgraph problem is to find a connected subgraph $S \subseteq G$ such that its *weight* $\sum_{v \in V(S)} p(v)$ is maximized. Throughout this section we consider a MWCSP $P_{MW} = (V, E, p)$

[2]Most of this section has been published in [15]

with the property that at least one vertex is assigned a negative and one a positive weight (otherwise the problem can be solved trivially).

The most comprehensive collections of reduction techniques for the MWCSP can be found in [17]. While (arguably) interesting in their own right, reduction techniques also form the backbone of the most successful solvers for the MWCSP. In the following, results from [17] are generalized.

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 $U \subset V \setminus X$. Then X will be said to *MWCS-dominate* U if

$$\big\{v \in V \setminus U \mid \exists \{v, w\} \in E, w \in U\big\} \subseteq \big\{v \in V \mid \exists \{v, w\} \in E, w \in X\big\} \cup X.$$

Importantly, one can remove U 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 U, 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 [17].

Proposition 6. *Let $U \subseteq V \setminus T$ and $X \subseteq V \setminus U$ such that X MWCS-dominates U and assume*

$$\sum_{u \in U} p(u) \leq \sum_{u \in X : p(u) < 0} p(u). \tag{35}$$

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

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

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

Next, remove U 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 $v \in \Delta_S$. 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 U and because each connected component contains a $v \in \Delta_S$, 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:

$$\sum_{u \in V(S''')} p(u) \geq \sum_{u \in V(S)} p(u) - \sum_{u \in U} p(u) + \sum_{u \in X : p(u) < 0} p(u) \overset{(35)}{\geq} \sum_{u \in V(S)} p(u).$$

This concludes the proof. □

While Proposition 6 guarantees that set U is not part of at least one optimal solution, the same may not be true for subsets of U. Therefore, one cannot just eliminate U in general. However, in the case of $|U| = 1$ one can forthwith eliminate U, and in the case of $|U| = 2$ with $U = \{v, w\} \in E$ one can eliminate the edge $\{v, w\}$. Figure 4 shows an MWCSP instance for which an edge can be eliminated by means of the criterion formulated in Proposition 6. The

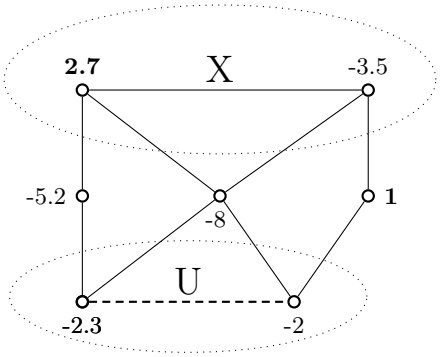


Figure 4: An MWCSP instance. Considering the vertices enclosed by the upper dotted ellipse as the set X and those enclosed by the lower one as U, one can verify with Proposition 6 that the dashed edge can be deleted.

vertices of the dashed edge have a summed weight of -4.3, smaller than the weight of the (sole) negative vertex in the MWCS-dominating set X marked by the upper dotted ellipse (which is -3.5).

In contrast to Proposition 6, 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 7. *Let $U \subseteq V \setminus T$ and $X \subseteq V \setminus U$ such that X MWCS-dominates U and assume*

$$\max_{w \in U} p(w) \leq \sum_{u \in X : p(u) < 0} p(u). \tag{36}$$

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

Proof. Let S be a feasible solution with $U \cap V(S) \neq \emptyset$. Further, define Δ_S as in the proof of Proposition 6. Remove $U \cap V(S)$ 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 $v \in \Delta_S$. These components can therefore be reconnected as in the proof of Proposition 6 to obtain a connected subgraph S''' with $U \cap V(S''') = \emptyset$. Because of (36) 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 5 shows an MWCSP instance that can be reduced by using Proposition 7.

For the special case of $|U| = 1$ a vertex set X max-weight MWCS-dominates a vertex set U if and only if X all-weights MWCS-dominates U. 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. 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 $v \in V_0$ with $p_0(v) \leq 0$ the single-weight MWCS-domination problem is to determine whether a subset of $V \setminus \{v\}$ exists that single-weight MWCS-dominates v.

Proposition 8. *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. Hence, the single-weight MWCS-dominating decision problem is in $\mathcal{NP}$.

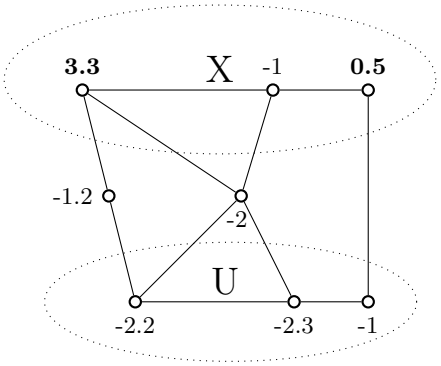


Figure 5: An MWCSP instance. Considering the vertices enclosed by the upper dotted ellipse as set X and the ones enclosed by the lower one as U, one can verify with Proposition 7 that all vertices of U (and incident edges) can be deleted.

In the following it will be demonstrated that the ($\mathcal{NP}$-complete [6]) 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 $p_0(v) := -1$ for all $v \in V_0$. In the next step replace each edge $e_l = \{v, w\} \in E_0$ by a vertex v'_l of weight $p_0(v'_l) := -(k+1)$ and the two edges $\{v, v'_l\}$ and $\{w, v'_l\}$. Moreover, add edges $\{v, w\}$ for each pair of distinct vertices $v, w \in V_0 \cap V_{cov}$ 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 $\{v_0^\star, v\}$ for all $v \in V_0 \setminus (V_{cov} \cup \{v_0^\star\})$.

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 $v \in V_0 \setminus (V_{cov} \cup \{v_0^\star\})$ there is an adjacent vertex $w \in V_0 \cap V_{cov}$ with $w \in X$. The latter condition is satisfied if and only if there is a vertex cover in G_{cov} of cardinality at most k. □

3.1 Bottleneck distances

Bottleneck Steiner distances are a classic concept for SPG reduction techniques [4], and has been generalized to PCSTP [18] and MWCSP [17]. Initially, this section translates (in a straightforward way) a recent generalization of this concept for PCSTP [16] to MWCSP. Let $v, w \in V$. A finite walk $W = (v_1, e_1, v_2, e_2, ..., e_r, v_r)$ with $v_1 = v$ and $v_r = w$ will be called *positive-weight constrained* (v, w)*-walk* if no $v \in T \cup \{v, w\}$ is contained more than once in W. For any $k, l \in \mathbb{N}$ with $1 \leq k \leq l \leq r$ define the subwalk $W(v_k, v_l) := (v_k, e_k, v_{k+1}, e_{k+1}, ..., e_l, v_l)$. In the following, let W be a positive-weight constrained (v, w)-walk. Define the *interior cost* of W as:

$$C^-(W) := \sum_{v \in V(W) \setminus \{v_k, v_l\}} p(v), \tag{37}$$

where the convention that the empty sum equals 0 is assumed, so the interior cost of an edge is likewise 0. Furthermore, define the *positive-weight constrained length* of W as:

$$l_{pw}(W) := \min\{C^-(W(v_k, v_l)) \mid 1 \leq k \leq l \leq r,\ v_k, v_l \in T \cup \{v, w\}\}. \tag{38}$$

Note that $l_{pw} \leq 0$ holds, because the interior cost of an edge is 0. Denote the set of all positive-weight constrained (v, w)-walks by $\mathcal{W}_{pw}(v, w)$ and define the *positive-weight constrained distance* between v and w as

$$d_{pw}(v, w) := \max\{l_{pw}(W) \mid W \in \mathcal{W}_{pw}(v, w)\}. \tag{39}$$

Note that it is $\mathcal{NP}$-hard to compute d_{pw}.

3.2 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 9. *Let $U \subseteq V \setminus T$ and define*

$$\Delta := \big\{v \in V \setminus U \mid \exists \{v, w\} \in E, w \in U\big\}$$

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

$$\Delta_1 := \Delta \cap \big(\big\{v \in V \setminus X \mid \exists \{v, w\} \in E, w \in X\big\} \cup X\big)$$

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

$$C_1 := \sum_{u \in X : p(u) < 0} p(u). \tag{40}$$

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

$$C_2 := \sum_{v_k \in \Delta_2} d_{pw}(v_k, v'_k). \tag{41}$$

If

$$C := C_1 + C_2 > \sum_{u \in U} p(u), \tag{42}$$

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

Proof. Let S be a feasible solution with $U \subsetneq V(S)$. 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 U and satisfies $P(S''') \geq P(S)$.

Let S' be the subgraph obtained from S by removing U 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

$$P(S') = P(S) - \sum_{u \in U} p(u). \tag{43}$$

If $\Delta_1^S \neq \emptyset$, let S'' be the vertex-induced subgraph of $X \cup V(S')$. Otherwise set $S'' := S'$. In both cases, it holds for S'' that

$$P(S'') \geq P(S') + C_1 \overset{(43)}{=} P(S) - \sum_{u \in U} p(u) + C_1. \tag{44}$$

Moreover, all vertices of 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)-walk W^k (with v'_k as defined in the statement of this proposition) such that $l_{pw}(W^k) = d_{pw}(v_k, v'_k)$. If v_k and v'_k are in different connected components of S''', there exist $v_q \in V(W^k)$ in the connected components of v_k and $v'_q \in V(W^k)$ in the connected component of v'_k such that $V(W^k(v_q, v'_q)) \cap V(S''') = \{v_q, v'_q\}$. Add (the subgraph corresponding to) $W^k(v_q, v'_q)$ to S'''. Because of condition (42) there is at least on vertex of U that is not contained in any of these newly added paths—otherwise it would hold that $C_2 \leq \sum_{u \in U} p(u)$ and therefore also $C \leq \sum_{u \in U} p(u)$. Moreover, because of condition (41) the overall procedure reduces the weight of S'' by at most $|C_2|$. Hence, it holds for the new (now connected) subgraph S''' that

$$P(S''') \geq P(S'') + C_2 \overset{(44)}{\geq} P(S) - \sum_{u \in U} p(u) + C_1 + C_2. \tag{45}$$

Finally, $U \nsubseteq V(S''')$ holds and due to (42) it follows from (45) that

$$P(S''') > P(S). \tag{46}$$

Hence the proposition is proven. □

Corollary 10. *Assume that the conditions of Proposition 9 hold, but instead of* (42) *assume*

$$C_1 + C_2 > \max_{u \in U} p(u). \tag{47}$$

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

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 9. S''' is connected and it holds that $P(S''') > P(S)$, so only the equation $U \cap V(S''') = \emptyset$ needs to be verified. By construction all vertices of S''' are in one of the three sets: $(V(S) \setminus U)$, X, and the set of vertices that are part of a (v_k, v'_k)-walk W^k with $l_{pw}(W^k) = d_{pw}(v_k, v'_k)$ and $v_k \in \Delta_2^S$. By definition the first two of these sets cannot contain any vertices of U. Furthermore, because of (47), none of the walks W^k can contain a vertex of U since otherwise it would hold that $l_{pw}(W^k) \leq \max_{u \in U} p(u)$—which is a contradiction because of $C_1 + C_2 \leq C_2 \leq l_{pw}(W^k)$. Thus, $U \cap V(S) = \emptyset$. □

Once again, for the special case of $|U| = 1$, corollary and proposition coincide. Figure 6 shows an MWCSP for which a vertex can be deleted by means of this special case. Consider the upper two encircled vertices as the set X. The right neighbor (forming the set Δ_2) of the filled vertex can be connected by a walk of positive-weight constrained length -1 to X, so $C_2 \geq -1$. Since $C_1 = -1$ and all other neighbors (Δ_1) of the filled vertex are also neighbors of X, one can delete the vertex.

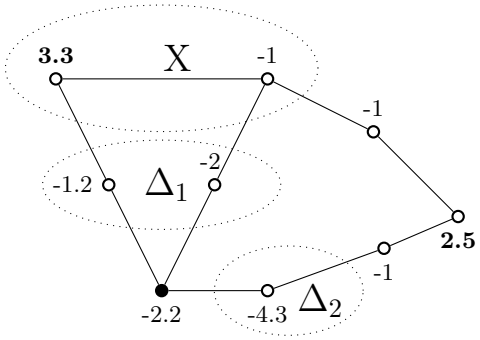


Figure 6: An MWCSP instance. Consider the vertices enclosed by the upper dotted ellipse as the set X, the lower left one as Δ_1, the lower right ones as Δ_2, and let U be the set that only contains the bottom left (filled) vertex. One can verify with Proposition 9 that the bottom left vertex can be deleted.

4 Conclusion

This paper has introduced various new preprocessing (or reduction) techniques for both the Steiner tree problem in graphs and the maximum-weight connected subgraph problem. Initial computational experiments with the Steiner tree solver SCIP-Jack [5] have already revealed a considerable potential of the SPG methods for reducing the problem size, and furthermore for strengthening exact solving. We plan to further extend the realization of the new techniques and fully implement them into SCIP-Jack. As to MWCSP, the reductions methods have already been fully integrated into SCIP-Jack and will be made publicly available as part of its next release.

5 Acknowledgements

This work was supported by the BMWi project *Realisierung von Beschleunigungsstrategien der anwendungsorientierten Mathematik und Informatik fur optimierende Energiesystemmodelle - BEAM-ME* (fund number 03ET4023DE). 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] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

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

[3] C. W. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19(5):549–567, 1989.

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

[5] 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.

[6] 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.

[7] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The scip optimization suite 6.0. Technical Report 18-26, ZIB, Takustr. 7, 14195 Berlin, 2018.

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

[9] Thorsten Koch, Alexander Martin, and Stefan Voß. SteinLib: An updated library on Steiner tree problems in graphs. In D.-Z. Du and X. Cheng, editors, *Steiner Trees in Industries*, pages 285–325. Kluwer, 2001.

[10] M. Leitner, I. Ljubic, M. Luipersbeck, M. Prossegger, and M. Resch. New Real-world Instances for the Steiner Tree Problem in Graphs. Technical report, ISOR, Uni Wien, 2014.

[11] Markus Leitner, Ivana Ljubic, 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.

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

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

[14] Tobias Polzin and Siavash Vahdati-Daneshmand. The Steiner Tree Challenge: An updated Study. Unpublished manuscript at http://dimacs11.cs.princeton.edu/downloads.html, 2014.

[15] D. Rehfeldt and T. Koch. Combining NP-Hard Reduction Techniques and Strong Heuristics in an Exact Algorithm for the Maximum-Weight Connected Subgraph Problem. *SIAM Journal on Optimization*, 29(1):369–398, 2019.

[16] Daniel Rehfeldt and Thorsten Koch. On the exact solution of prize-collecting steiner tree problems. Technical Report 20-11, ZIB, Takustr. 7, 14195 Berlin, 2020.

[17] Daniel Rehfeldt, Thorsten Koch, and Stephen J. Maher. Reduction techniques for the prize collecting Steiner tree problem and the maximum-weight connected subgraph problem. *Networks*, 73(2):206–233, 2019.

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