The bi-objective prize-collecting Steiner tree problem

Markus Leitner
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
leitner@ads.tuwien.ac.at

Ivana Ljubić, Markus Sinnl
Department of Statistics and Operations Research
University of Vienna, Austria
{ivana.ljubic,markus.sinnl}@univie.ac.at

May 27, 2013

Abstract

We consider the bi-objective prize-collecting Steiner tree problem, whose goal is to find a subtree considering the conflicting objectives of minimizing the edge costs for building that tree, and maximizing the collected node revenues. From a practical perspective, the problem is important whenever node revenues and edge costs cannot be easily expressed using the same (e.g., monetary) units. For example, in wildlife corridor design, nodes correspond to land parcels and their revenues correspond to their associated wildlife benefit. We consider five iterative mixed integer programming frameworks that identify the complete Pareto front, i.e., one efficient solution for every point on the Pareto front. More precisely, the following methods are studied: an ϵ-constraint method, a two-phase method, a binary search in the objective space, a weighted Chebyscheff norm method and a method of Sylva and Crema. We also investigate how to exploit and recycle information gained during these iterative MIP procedures in order to accelerate the solution process. We consider (i) additional strengthening valid inequalities, (ii) procedures for initializing feasible solutions, (iii) procedures for recycling violated cuts using cut pools, and (iv) guiding the branching process by previously detected Pareto optimal solutions. This work is a first study on exact approaches for solving the bi-objective prize-collecting Steiner tree problem. Standard benchmark instances from the literature are used to assess the efficacy of the proposed methods.

1 Introduction

Many problems arising in the design of telecommunication, district heating, or water distribution networks, in forestry planning or even in image processing and system biology can be modeled as variants of the *prize-collecting Steiner tree*

problem (PCSTP) (see e.g., [1] for a recent survey). The problem is defined as follows: Given an undirected graph $G = (V, E)$, with node revenues $r : V \to \mathbb{N}_0$, and edge costs $c : E \to \mathbb{N}$, find a subtree $G' = (V', E'), V' \subseteq V, E' \subseteq E$, such that $\sum_{v \in V'} r_v - \sum_{e \in E'} c_e$ is maximized. If node revenues and edge costs are measured using the same (e.g., monetary) units, this objective function corresponds to the net-worth maximization problem. Difficulties arise, however, if "node revenues" are used to model aspects that cannot be so easily expressed using the same units. For instance, in wildlife corridor design (see [8]), the goal is to determine a subset of land parcels that connect areas of biological significance for a given species. In this case, nodes correspond to land parcels and their revenues correspond to habitats' suitability. Typically, in such applications, the problem is modeled as a budget-constrained prize-collecting Steiner tree problem (see [8]), where the budget limits are provided by decision makers, and the collected revenues are maximized subject to the given budget. From the perspective of a decision maker it would be preferable to consider both objectives, namely, the cost minimization and the revenue maximization, as part of the optimization process. In that case, the decision maker would be able to easily perform a sensitivity analysis regarding the variations of the budget and the corresponding deviations of the collected revenue. Therefore, for dealing with tree problems in which the tree cost has to be minimized, on the one hand, and the collected revenue should be maximized, on the other hand, we propose to solve the PCSTP as a bi-objective optimization problem.

Problem Formulation

The set of all subtrees of G will be modeled using a rooted Steiner arborescence model originally proposed by [12], see also [23]. Let T be the set of *terminal nodes*, (i.e., nodes with positive revenue), and $V \backslash T$ be the set of potential *Steiner nodes* (i.e., nodes with zero revenue). We create a directed graph $(V \cup \{0\}, A)$ with a root 0 as follows: $A = \{(i, j), (j, i) \mid \forall e = \{i, j\} \in E\} \cup \{(0, t) \mid \forall t \in T\}$, set $c_{ij} = c_{ji} = c_e$, for all $e = \{i, j\}$ and $c_{0t} = 0$, for all $t \in T$. Using binary variables x_a, for all $a \in A$ to indicate whether an arc is part of a solution and binary variables y_v, for all $v \in V$ to indicate if a vertex v is part of a solution, the following inequalities are used to model the set of all feasible arborescences of G rooted at 0:

$$\mathcal{P} = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{|A|+|V|} \mid x(\delta^-(i)) = y_i, \forall i \in V, \;\; x(\delta^+(0)) = 1, \\ x(\delta^-(W)) \geq y_k, \forall W \subseteq V, k \in W\},$$

where $\delta^+(W)$ and $\delta^-(W)$ denote the outgoing and incoming cutset, respectively, for any $W \subseteq V$, and $x(A') = \sum_{a \in A'} x_a$ for a given $A' \subseteq A$. Connectivity constraints $x(\delta^-(W)) \geq y_k$ make sure that there is a directed path between the root and any other node of the solution, and the root-outdegree constraint $x(\delta^+(0)) = 1$ ensures that the subgraph obtained after removing node 0 from the solution is connected. The bi-objective prize-collecting Steiner tree problem (BOPCSTP) can now be defined as follows:

$$(\min \sum_{a \in A} c_a x_a, \max \sum_{t \in T} r_t y_t) \text{ subject to } (\mathbf{x}, \mathbf{y}) \in \mathcal{P}. \tag{1}$$

Thereby, the two objective functions need to be simultaneously optimized, i.e., no objective is more significant than the other one. Figure 1 shows an instance

of the BOPCSTP, two Pareto optimal solutions and the Pareto front, cf. Section 2.1.

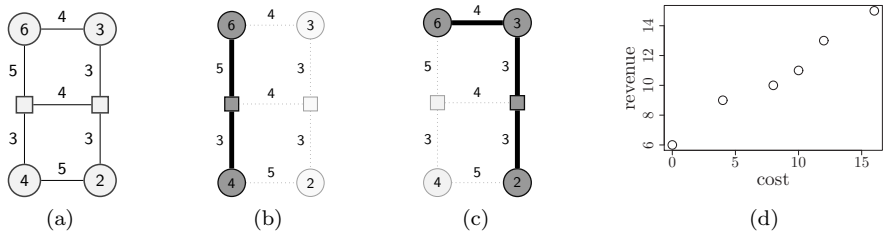


Figure 1: (a) An instance of the BOPCSTP. The costs of the edges and the revenues of the terminals are indicated by the numbers on the edges and nodes; the squares are Steiner nodes. (b), (c) Two Pareto optimal solutions with cost eight and revenue ten, and cost ten and revenue eleven, respectively. (d) Pareto front for this instance.

Let $\boldsymbol{\sigma} = (\mathbf{x}, \mathbf{y})$ denote the vector representing a feasible BOPCSTP solution. Notice that replacing the maximization problem of the second objective with the minimization problem $\min \sum_{t \in T} r_t(1 - y_t)$, yields the same solution $\boldsymbol{\sigma}$. This follows from the fact that the difference between the two objective values is $\sum_{t \in T} r_t$, which is a constant. From this point of view, the node revenues can be interpreted as penalties, which have to be paid, if a node is not part of the solution tree and the second objective becomes penalty minimization. Therefore, in the remainder of this paper, we will refer to the BOPCSTP as the bi-objective minimization problem $\min_{\boldsymbol{\sigma} \in \mathcal{P}}(z_1(\boldsymbol{\sigma}), z_2(\boldsymbol{\sigma}))$ where

$$z_1(\boldsymbol{\sigma}) = \sum_{a \in A} c_a x_a \text{ and } z_2(\boldsymbol{\sigma}) = \sum_{t \in T} r_t(1 - y_t).$$

Our goal is to find one efficient solution for every point on the Pareto front.

Scientific Contribution. In this paper we consider five *iterative mixed integer programming (MIP) frameworks* for bi-objective combinatorial optimization problems applied to the BOPCSTP. We solve the BOPCSTP using: an ϵ-constraint method, a two-phase method, a binary search in objective space, for which we also propose a small improvement, a method based on a weighted Chebyscheff norm and a method of Sylva and Crema [38]. Moreover, we show how to recycle the information gained during these iterative procedures: We accelerate the solution process by defining various strengthening inequalities, exploiting methods for generating feasible solutions, re-using violated cutting planes, and by guiding the branching process. This paper is a first study on exact approaches for solving the bi-objective prize-collecting Steiner tree problem.

Outline of the Paper. The rest of the paper is organized as follows: We first review previous work on bi- and multi-objective spanning and Steiner tree problems. We also briefly review related literature on the single objective PCSTP in this section. In Section 2 solution methods are described, while in Section 3

details about used acceleration techniques are given. Section 4 provides the results of our computational study and describes all computational details of the *iterative branch-and-cut approaches*. Finally, some conclusions are drawn in Section 5.

Previous and Related Work

Multicriteria Optimization on Trees. In the earliest works on bi- and multi-objective tree problems, several variants of the minimum spanning tree problem with more than one set of edge costs are considered (see e.g., Hamacher and Ruhe [15], Ramos et al. [31], Steiner and Radzik [37]). These algorithms for bi- and multi-objective spanning trees use combinatorial arguments and are based on efficient algorithms for the single-objective problem (see also the survey by Ruzika and Hamacher [33]). Moreover, Sourd and Spanjaard [36] introduced a general multi-objective branch-and-bound algorithm and applied it to the bi-objective minimum spanning tree problem. Vujošević and Stanojević [41] introduced a bi-objective variant of the Steiner tree problem with edge capacities and edge costs. They presented heuristic approaches that aim to maximize the capacity and minimize the total cost of a solution. Levin and Nuiriakhmetov [22] proposed a heuristic to solve a Steiner tree problem with more than one set of edge costs. Another bi-objective variant of the Steiner tree problem was presented by Martins and Ferreira [25], who proposed a heuristic aiming to simultaneously minimize the total solution costs and the number of intermediate (i.e., Steiner) nodes.
The BOPCSTP differs from these bi-objective spanning/Steiner tree problems, since feasible solutions for these problems are only spanning trees or Steiner trees spanning the given set of terminals, whereas in our problem, even a single-node subtree is a feasible solution.

PCSTP. Segev [35] introduced the *node-weighted Steiner tree problem* in graphs. In this modification of the classical Steiner tree problem, non-positive weights are associated to nodes and the objective is to find a subtree that minimizes the sum of the edge-cost and node-weights. This problem (as well as its maximization counterpart) is not approximable within any constant ratio. Bienstock et al. [5] considered a different minimization problem, to which they refer as the *prize-collecting Steiner tree problem*: The goal is to find a subtree that minimizes edge-cost plus node-weights of those nodes not connected by that tree. For this version of the problem (with non-negative node weights) Bienstock et al. [5] presented a factor three approximation algorithm and were the first to call the problem PCSTP. Other approximation algorithms with better approximation ratios were presented by Goemans and Williamson [13], Johnson et al. [17], Feofiloff et al. [11] and Archer et al. [2], which achieves the best known approximation ratio of $(2 - \epsilon)$.

MIP approaches to the PCSTP based on single- and multi-commodity flow formulations were introduced by Segev [35]. Lucena and Resende [24] proposed a cutting plane algorithm to find lower bounds and da Cunha et al. [7] a relax and cut algorithm to find primal and dual bounds. Haouari et al. [16] considered Lagrangian dual approaches for a generalized variant of the PCSTP with additional quota constraints. Ljubić et al. [23] proposed a branch-and-cut

approach using connectivity inequalities for directed Steiner trees previously studied by Fischetti [12].

2 Solution Methods

In the last decades, many algorithmic approaches were proposed for solving bi-objective combinatorial optimization problems (see e.g., the survey by Ehrgott and Wiecek [9]). In this study on the BOPCSTP we will concentrate on *iterative MIP approaches* that repeatedly solve single-objective problems (possibly augmented with some additional constraints) with a modified objective function. One of our major goals is to computationally compare these methods and to study how to recycle the information gained through solving a series of MIPs. For this computational study we select five iterative MIP approaches proposed in the literature in the last decades. For one of them, namely, the binary search in the objective space, we also present slight modifications that reduce the number of iterations and improve the performance. The chosen approaches explore the Pareto front in different ways, which is why they are chosen as representatives of iterative MIP approaches for bi-objective combinatorial optimization.

Since the main focus is on iterative MIP frameworks, we do not consider methods like bi-objective branch-and-bound or bi-objective branch-and-cut algorithms, see, e.g., Jozefowiez et al. [18], Mavrotas and Diakoulaki [26], Sourd and Spanjaard [36]. Note finally that all considered methods could be parallelized by partitioning the problem into intervals like described in, e.g., Lemesre et al. [21]. Hybridizations of the considered methods are also possible, and are expected to further improve the performance of these iterative frameworks. Study of these parallelizations and/or hybridizations, however, remains out of scope of this paper.

2.1 Preliminaries

In the following we recall the basic concepts about efficiency and nondominance needed to understand and solve bicriteria optimization problems (see, e.g., Przybylski et al. [28] for more details). Note that our problem is a bi-objective integer programming problem and as such it is more difficult to solve than bi-objective linear programming problems (the latter ones are much better studied and understood).

Let $\mathbf{z}(\boldsymbol{\sigma}) = (z_1(\boldsymbol{\sigma}), z_2(\boldsymbol{\sigma}))$ be a vector of two minimization objective functions, $\boldsymbol{\sigma}$ a feasible solution, $\mathcal{P}$ be the feasible region and $Z = \{\mathbf{z}(\boldsymbol{\sigma}) : \boldsymbol{\sigma} \in \mathcal{P}\}$ be the set of objective vectors. $\mathcal{P}$ is called the *decision space* and Z is called the *objective space*. A solution $\boldsymbol{\sigma} \in \mathcal{P}$ is *efficient* (*Pareto optimal*), iff there is no $\boldsymbol{\sigma}' \in \mathcal{P}$, s.t. $z_i(\boldsymbol{\sigma}') \leq z_i(\boldsymbol{\sigma})$ for $i = 1, 2$ with at least one strict inequality. The set of efficient solutions is denoted by $\mathcal{P}_E$ and the corresponding vectors $\mathbf{z}(\boldsymbol{\sigma})$ are called *non-dominated* (ND). The set of these solutions is denoted by Z_N and called *non-dominated frontier* or *Pareto front*.

Note that $|Z_N| \leq |\mathcal{P}_E|$, since different efficient solutions can have the same non-dominated vectors. For the BOPCSTP, we can derive an upper bound on the number of non-dominated vectors as follows: Note that any point (z_1, z_2), which lies on the Pareto front, must have unique coordinates in both objective functions. Thus the maximum number of points is bounded by the minimum

of the maximum values, which both objective functions can attain. Since z_1 is at most the cost of a minimum Steiner tree connecting all terminals (denote this cost by C) and z_2 is at most $\sum_{t\in T} r_t$, the number of vectors on the non-dominated frontier is at most $\min(C/\Delta_c, (\sum_{t\in T} r_t)/\Delta)$, where Δ_c and Δ are the greatest common divisors for edge costs and node revenues, respectively.

The set $\mathcal{P}_E$ is the disjoint union of *supported efficient solutions* (*SE*) and *non-supported efficient solutions* (*NSE*). Supported efficient solutions are all efficient solutions, where the associated point lies on the convex hull of all solutions in the objective space. They can be found using the weighted sum method, cf. Section 2.3. Non-supported efficient solutions are all remaining efficient solutions, i.e., their associated non-dominated points lie in the interior of this convex hull. Supported efficient solutions that correspond to extreme points of the convex hull of all solutions in the objective space are called *extreme efficient solutions*. The corresponding subsets in the objective space are called *supported*, *non-supported* and *extreme non-dominated points*, respectively.

Important concepts which will be used by all of the following methods are the ideal point and the Nadir point.

Definition 1 (Ideal point and Nadir point)**.** *Given a bi-objective combinatorial optimization problem with objectives $z_1(\boldsymbol{\sigma})$ and $z_2(\boldsymbol{\sigma})$, the ideal point $\mathbf{z}^{\mathrm{I}}$ is defined as $\mathbf{z}^{\mathrm{I}} = (z_1^{\mathrm{I}}, z_2^{\mathrm{I}}) = (\min_{\boldsymbol{\sigma}\in\mathcal{P}} z_1(\boldsymbol{\sigma}), \min_{\boldsymbol{\sigma}\in\mathcal{P}} z_2(\boldsymbol{\sigma}))$. The Nadir point $\mathbf{z}^{\mathrm{N}}$ is defined as $\mathbf{z}^{\mathrm{N}} = (z_1^{\mathrm{N}}, z_2^{\mathrm{N}}) = (\min_{\boldsymbol{\sigma}\in\mathcal{P}}\{z_1(\boldsymbol{\sigma}) : z_2(\boldsymbol{\sigma}) \leq z_2^{\mathrm{I}}\}, \min_{\boldsymbol{\sigma}\in\mathcal{P}}\{z_2(\boldsymbol{\sigma}) : z_1(\boldsymbol{\sigma}) \leq z_1^{\mathrm{I}}\})$.*

Obviously the points $(z_1^{\mathrm{I}}, z_2^{\mathrm{N}})$ and $(z_1^{\mathrm{N}}, z_2^{\mathrm{I}})$ are the boundary points of the non-dominated frontier. In the BOPCSTP, the point $(z_1^{\mathrm{I}}, z_2^{\mathrm{N}})$ corresponds to a solution consisting of a single terminal with the highest revenue and $(z_1^{\mathrm{N}}, z_2^{\mathrm{I}})$ corresponds to a minimum Steiner tree connecting all terminals. To simplify notation, we will sometimes slightly abuse the notation in the following sections and refer to some $\mathbf{z} = (z_1, z_2)$ not only as solution value, but also as feasible solution $\boldsymbol{\sigma} \in P$ such that $\mathbf{z}(\boldsymbol{\sigma}) = \mathbf{z}$ or as a tree corresponding to this solution. Moreover, we will sometimes use the terms non-dominated point and its associated Pareto optimal solution interchangeably. For two distinct vectors $\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}} \in Z$ such that $z_1^a < z_1^b$, by $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ we will denote the interval between the two vectors in the objective space. Finally, some methods will also find weakly Pareto optimal solutions (we will point out how to deal with this in the description of the respective methods).

Definition 2 (Weakly Pareto Optimal Solutions)**.** *A solution $\boldsymbol{\sigma} \in \mathcal{P}$ is weakly efficient/Pareto optimal, iff there is no $\boldsymbol{\sigma}' \in \mathcal{P}$, s.t. $z_1(\boldsymbol{\sigma}') < z_1(\boldsymbol{\sigma})$ and $z_2(\boldsymbol{\sigma}') < z_2(\boldsymbol{\sigma})$.*

2.2 ϵ-Constraint Method

In the ϵ-constraint method for bi-objective optimization, one objective function is kept, while an upper bound is imposed on the other objective function by adding an additional constraint (for a default description of this approach for multi-criteria optimization, see, e.g., Ehrgott and Wiecek [9]).

We now present an approach combining the ϵ-constraint method with a branch-and-cut framework, similar to the one of Bérubé et al. [4] used for solving the traveling salesman problem with profits. In our approach, the second

objective is relaxed and added as constraint. Hence, for a given $\epsilon \geq z_2^{\mathrm{I}}$, we obtain the following ϵ-constraint problem $P(\epsilon)$ for the BOPCSTP:

$$P(\epsilon): \qquad \min\{\sum_{a\in A} c_a x_a \mid (\mathbf{x},\mathbf{y}) \in \mathcal{P},\ \sum_{t\in T} r_t(1-y_t) \leq \epsilon\}$$

Note that every optimal solution for a given value of ϵ is (weakly) Pareto optimal. In the ϵ-constraint method, this fact, together with the integrality of the input, is used to systematically find one solution for every point on the Pareto front by successively decreasing ϵ by Δ (recall that Δ is the greatest common divisor of all revenues r_t, $t \in T$). Since we replace the second objective by an ϵ-constraint, the starting boundary point is the Pareto optimal point with the minimum possible value for the first objective, i.e., $\epsilon = z_2^{\mathrm{N}}$. This point corresponds to a solution containing no edges in which a terminal with maximum node revenue $r^* = \max_{t\in T} r_t$ is selected (ties are broken randomly).

We therefore initialize $\epsilon = \sum_{t\in T} r_t - r^* - \Delta$ and set $\epsilon = z_2(\boldsymbol{\sigma}^*) - \Delta$ in each iteration after computing the optimal solution $\boldsymbol{\sigma}^*$. The algorithm terminates when the Pareto optimal point with the minimum possible value of the second objective is reached, i.e., when ϵ becomes less than or equal to z_2^{I}. This point corresponds to a solution which is the minimum cost Steiner tree connecting all terminals. An overview of this approach is given in Algorithm 1. The Pareto optimal solutions are kept in the set *Sol*.

Algorithm 1 ϵ-Constraint Method for the BOPCSTP

$z_1^{\mathrm{I}} \leftarrow 0$; $z_2^{\mathrm{I}} \leftarrow 0$; $r^* \leftarrow \max_{t\in T} r_t$; $z_2^{\mathrm{N}} \leftarrow r^*$
Sol $\leftarrow \{(z_1^{\mathrm{I}}, z_2^{\mathrm{N}})\}$; $\epsilon \leftarrow \sum_{t\in T} r_t - r^* - \Delta$
while $\epsilon \geq z_2^{\mathrm{I}}$ **do**
  $\boldsymbol{\sigma}^* \leftarrow$ argmin $P(\epsilon)$
  Sol $\leftarrow$ *Sol* $\cup \{\boldsymbol{\sigma}^*\}$
  $\epsilon \leftarrow \mathbf{z}_2(\boldsymbol{\sigma}^*) - \Delta$
Remove weakly Pareto optimal points from *Sol*

Choosing the cost of the solution tree as objective function and the lost revenue as ϵ-constraint has two main advantages: First, the ϵ-constraint is of knapsack-type, so it implies further strengthening inequalities that are described in Section 3.1. Second, there is no need to compute the ideal and Nadir points separately to find the boundary points of the non-dominated frontier, since these points are found in the first and last iteration of the algorithm as described above.

Since we are only minimizing the first objective, the set *Sol* may contain solutions with the same cost but different penalties, i.e., the proposed algorithm may also find weakly Pareto optimal solutions. We can easily resolve this problem in pseudo-polynomial time: In a post-processing phase, among all solutions with the identical z_1 value, we only keep the single solution with the minimal z_2 value. Alternatively, one can also use a modified objective function (see, e.g. Schweigert and Neumayer [34]): $\mathbf{z}'(\boldsymbol{\sigma}) = \gamma z_1(\boldsymbol{\sigma}) + z_2(\boldsymbol{\sigma})$, with γ chosen s.t. $\gamma \geq z_2^{\mathrm{N}} + 1$. This way, only Pareto optimal solutions are found, since both objectives get minimized. However, our preliminary tests showed that modifying the objective function deteriorates the performance, so we will not consider this variant in the computational results given in Section 4.

2.3 Two-Phase Method

The two-phase method was introduced by Ulungu and Teghem [39] for the bi-objective assignment problem and was subsequently used to solve various bi-objective variants of problems such as the knapsack problem [40] or the integer minimum cost flow problem [29]. It is the most-popular method to solve bi-objective combinatorial optimization problems, when the associated single-objective counterpart is easily solvable, i.e., by a specialized, (pseudo) polynomial time algorithm. This stems from the fact that using this method, in the first phase, no constraints need to be added to the problem at hand, but only the coefficients of the objective function have to be modified. Hence any single-objective algorithm for the considered problem can be recycled. In our case, however, there is no efficient single-objective algorithm (the PCSTP is NP-hard), so there are no benefits at hand when comparing the two-phase method with the other iterative MIP approaches.

In the *first phase* of this approach, the following weighted sum problem, denoted by $P(\lambda_1, \lambda_2)$, is repeatedly solved for different combinations of nonnegative weight parameters λ_1, λ_2:

$$P(\lambda_1, \lambda_2): \qquad \min\{\lambda_1 \sum_{a \in A} c_a x_a + \lambda_2 \sum_{t \in T} r_t(1 - y_t) \mid (\mathbf{x}, \mathbf{y}) \in \mathcal{P}\}$$

This weighted sum problem is embedded into an interval-based algorithm (see Algorithm 2), where we set $\lambda_1 = z_2^{\mathrm{a}} - z_2^{\mathrm{b}}$ and $\lambda_2 = z_1^{\mathrm{b}} - z_1^{\mathrm{a}}$ for a given interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ in the objective space. The new objective function is parallel to the segment connecting $\mathbf{z}^{\mathrm{a}}$ and $\mathbf{z}^{\mathrm{b}}$. Each optimal solution of $P(\lambda_1, \lambda_2)$ corresponds to a supported non-dominated point that lies in the efficient frontier between $\mathbf{z}^{\mathrm{a}}$ and $\mathbf{z}^{\mathrm{b}}$ (including $\mathbf{z}^{\mathrm{a}}$ and $\mathbf{z}^{\mathrm{b}}$). If the line parallel to $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ belongs to the convex hull of the efficient frontier, then multiple supported non-dominated solutions exist on this line. Therefore, in the default implementation of this algorithm, it is assumed that *all* optimal solutions of $P(\lambda_1, \lambda_2)$ are determined. Among these optimal solutions, we define point $\tilde{\mathbf{z}}$ to be the most left one (according to the z_1-coordinate) and point $\hat{\mathbf{z}}$ to be the most right point. Now, if $\tilde{\mathbf{z}} \neq \mathbf{z}^{\mathrm{a}}$ and $\hat{\mathbf{z}} \neq \mathbf{z}^{\mathrm{b}}$, the starting interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ is divided into two new intervals $[\mathbf{z}^{\mathrm{a}}, \tilde{\mathbf{z}}]$ and $[\hat{\mathbf{z}}, \mathbf{z}^{\mathrm{b}}]$ and the process is repeated until all intervals are proceeded and no further optimal solutions are found. At the beginning, this interval-based algorithm is initialized with a single interval connecting the two boundary points of the efficient frontier. At the end, the set *SE* contains all supported efficient non-dominated points. For further details, see, e.g. Przybylski et al. [28].

In the *second phase*, all non-supported non-dominated points are found. For this purpose we apply the ϵ-constraint method (cf. Section 2.2) to the intervals between consecutive supported non-dominated points. Algorithm 3 provides a detailed overview of phase two, where $P'(\epsilon, \mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$ denotes the previously defined problem $P(\epsilon)$ additionally augmented with the following two constraints:

$$\lambda_1 \sum_{a \in A} c_a x_a + \lambda_2 \sum_{t \in T} r_t(1 - y_t) \geq \lambda_1 z_1^{\mathrm{a}} + \lambda_2 z_2^{\mathrm{a}} \tag{2}$$

$$\sum_{t \in T} r_t(1 - y_t) \geq z_2^{\mathrm{b}} + \Delta \tag{3}$$

Inequality (2) states that the non-supported non-dominated points are above

Algorithm 2 Two-Phase Method: Phase One

Compute the ideal point $\mathbf{z}^{\mathrm{I}}$ and the Nadir point $\mathbf{z}^{\mathrm{N}}$
$I \leftarrow \{[(z_1^{\mathrm{I}}, z_2^{\mathrm{N}}), (z_1^{\mathrm{N}}, z_2^{\mathrm{I}})]\}$; $SE \leftarrow \{(z_1^{\mathrm{I}}, z_2^{\mathrm{N}}), (z_1^{\mathrm{N}}, z_2^{\mathrm{I}})\}$
while $I \neq \emptyset$ **do**
  select an interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ and remove it from I
  $\lambda_1 \leftarrow z_2^{\mathrm{a}} - z_2^{\mathrm{b}}$; $\lambda_2 \leftarrow z_1^{\mathrm{b}} - z_1^{\mathrm{a}}$
  $Opt \leftarrow$ all argmin $P(\lambda_1, \lambda_2)$
  $SE \leftarrow SE \cup \{\mathbf{z}(\boldsymbol{\sigma})\}_{\boldsymbol{\sigma} \in Opt}$
  if $\{\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}\} \cap \{\mathbf{z}(\boldsymbol{\sigma})\}_{\boldsymbol{\sigma} \in Opt} = \emptyset$ **then**
    $\tilde{\boldsymbol{\sigma}} \leftarrow \mathrm{argmin}_{\boldsymbol{\sigma} \in Opt}\{z_1(\boldsymbol{\sigma})\}$; $\tilde{\mathbf{z}} \leftarrow \mathbf{z}(\tilde{\boldsymbol{\sigma}})$
    $\hat{\boldsymbol{\sigma}} \leftarrow \mathrm{argmax}_{\boldsymbol{\sigma} \in Opt}\{z_1(\boldsymbol{\sigma})\}$; $\hat{\mathbf{z}} \leftarrow \mathbf{z}(\hat{\boldsymbol{\sigma}})$
    $I \leftarrow I \cup \{[\mathbf{z}^{\mathrm{a}}, \tilde{\mathbf{z}}], [\hat{\mathbf{z}}, \mathbf{z}^{\mathrm{b}}]\}$

the line segment connecting the current interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ in the objective space, while constraint (3) ensures that the solution differs from $\mathbf{z}^{\mathrm{b}}$.

Algorithm 3 Two-Phase Method: Phase Two

$Sol \leftarrow SE$ found in phase one
let $z^1, \ldots, z^{|SE|}$ be the supported non-dominated points sorted in increasing order of z_1
for $i = 1, \ldots |SE| - 1$ **do**
  $\mathbf{z}^{\mathrm{a}} \leftarrow \mathbf{z}^i, \quad \mathbf{z}^{\mathrm{b}} \leftarrow \mathbf{z}^{i+1}, \quad \epsilon \leftarrow z_2^{\mathrm{a}} - \Delta$
  while $P'(\epsilon, \mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$ has at least one feasible solution **do**
    $\boldsymbol{\sigma}^* \leftarrow$ argmin $P'(\epsilon)$
    $Sol \leftarrow Sol \cup \{\boldsymbol{\sigma}^*\}$
    $\epsilon \leftarrow \mathbf{z}_2(\boldsymbol{\sigma}^*) - \Delta$

Note that with the incorporation of the ϵ-constraint method in the second phase, there is *no need* to search for *all* optimal solutions of the problem $P(\lambda_1, \lambda_2)$ in the first phase. It suffices to find a single optimal solution (in which case $\hat{\mathbf{z}}$ and $\tilde{\mathbf{z}}$ are identical), and the remaining supported non-dominated points will be found in the second phase. This is an important performance enhancement since enumerating all optimal solutions with standard MIP solvers is rather slow. Thus, in our default implementation, we consider a variant in which only a single solution is found in the first phase, for each pair (λ_1, λ_2). The second phase then detects the remaining supported non-dominated points along with the non-supported ones. Note that in this variant at least the set of extreme non-dominated points is identified in the first phase.

2.4 Binary Search in the Objective Space

Riera-Ledesma and Salazar-González [32] proposed an algorithm that explores the objective space in a similar fashion as the two-phase method, but runs in a single phase and determines both supported- and non-supported points in the same run. We call their approach *binary search in the objective space*. The main idea behind this approach is to recursively divide the objective space into intervals and search for solutions inside of them.

Assume that $\mathbf{z}^{\mathrm{a}}$ and $\mathbf{z}^{\mathrm{b}}$ are two non-dominated points with $z_1^{\mathrm{a}} \leq z_1^{\mathrm{b}}$ that

specify the current interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$ in the objective space. The algorithm repeatedly solves the single-objective problem $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$, which, for the BOPCSTP, is defined as follows:

$$P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}): \qquad \min\{\lambda_1 \sum_{a \in A} c_a x_a + \lambda_2 \sum_{t \in T} r_t(1 - y_t) \mid (\mathbf{x}, \mathbf{y}) \in \mathcal{P},$$
$$\sum_{a \in A} c_a x_a \leq z_1^{\mathrm{b}} - \Delta_c,\ \sum_{t \in T} r_t(1 - y_t) \leq z_2^{\mathrm{a}} - \Delta, \lambda_1 = z_2^{\mathrm{a}} - z_2^{\mathrm{b}},\ \lambda_2 = z_1^{\mathrm{b}} - z_1^{\mathrm{a}}\}$$

The two added inequalities restrict the objective space to a region in which we search for a new efficient solution. An overview of the algorithm is given in Algorithm 4[1]. As in the two-phase method, the point $\mathbf{z}^* = \mathbf{z}(\boldsymbol{\sigma}^*)$ corresponding to the optimal solution of $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$, is then used to define two new intervals $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^*]$ and $[\mathbf{z}^*, \mathbf{z}^{\mathrm{b}}]$. Note that if there exists a solution of $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$, then the corresponding point $\mathbf{z}^*$ is always non-dominated. Due to the added two inequalities, we allow non-supported points to be found as optimal solutions of $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$ which is the main difference of this approach when compared to the two-phase method. Before choosing the next interval to investigate, we sort the interval set I according to z_1 and pick the topmost. This way we explore the objective space in a systematic way, which is beneficial for some of the acceleration methods, e.g., cut pool and feasible starting solutions (see Section 3).

Algorithm 4 Binary Search in the Objective Space

```
Compute the ideal point z^I and the Nadir point z^N
I ← {[(z_1^I, z_2^N), (z_1^N, z_2^I)]}; Sol ← {(z_1^I, z_2^N), (z_1^N, z_2^I)}
while I ≠ ∅ do
  select an interval [z^a, z^b] ∈ I and remove it from I
  if z_1^b − Δ_c ≥ z_1^a + Δ_c ∧ z_2^b + Δ ≤ z_2^a − Δ then
    σ* ← argmin P(z^a, z^b)
    if σ* ≠ ∅ then
      Sol ← Sol ∪ {z(σ*)}
      I ← I ∪ {[z^a, z(σ*)], [z(σ*), z^b]}
```

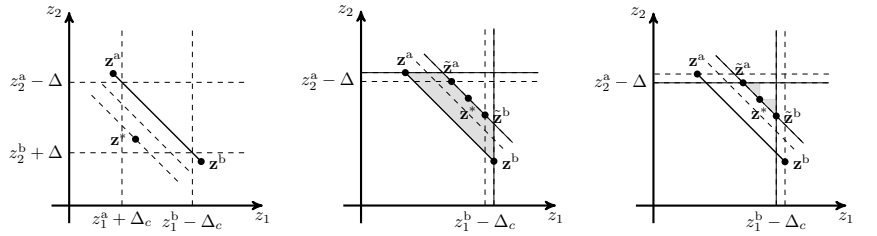
A drawback of this method is that even for the "empty intervals" (i.e., intervals containing no non-dominated points, except the two ones on its border), a MIP has to be executed until infeasibility is detected. On the other hand, the advantage of this method is that it does not find weakly Pareto optimal solutions.

2.4.1 A New Improvement of the Binary Search in the Objective Space

Notice that for a given interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$, we need to solve the MIP $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$ only if $z_1^{\mathrm{b}} - \Delta_c \geq z_1^{\mathrm{a}} + \Delta_c$ and $z_2^{\mathrm{b}} + \Delta \leq z_2^{\mathrm{a}} - \Delta$ holds (otherwise there will be no non-dominated points in this interval, see Figure 2a).

By the definition of the objective function $P(\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}})$, a newly discovered non-dominated point is the outmost point in the objective space, which lies on a line

[1] Note that due to small typographical errors in Riera-Ledesma and Salazar-González [32] our description is slightly different then the originally proposed one.



(a) Pareto optimal solution. (b) Interval tightening. (c) Interval tightening.

Figure 2: Illustration of effect of the chosen weight.

parallel to the line through $[\mathbf{z}^a, \mathbf{z}^b]$ (cf. Figure 2a). Notice that all non-dominated points lying below the $[\mathbf{z}^a, \mathbf{z}^b]$ segment are supported non-dominated points and those above it are non-supported ones. We now propose an improvement of the binary search method that computes better interval bounds for two new intervals defined *after a non-supported non-dominated point* $\mathbf{z}^*$ *is found.*

Proposition 1. *Let* $\mathbf{z}^*$ *be an optimal solution of* $P(\mathbf{z}^a, \mathbf{z}^b)$ *for a given interval* $[\mathbf{z}^a, \mathbf{z}^b]$. *If* $\mathbf{z}^*$ *is non-supported, then the two new intervals to be solved are defined as* $[\tilde{\mathbf{z}}^a, \mathbf{z}^*]$ *and* $[\mathbf{z}^*, \tilde{\mathbf{z}}^b]$, *where*

$$\tilde{\mathbf{z}}^a = (\tilde{z}_1^a, \tilde{z}_2^a) \quad \textit{with} \quad \tilde{z}_1^a = \lceil (\lambda_1 z_1^* + \lambda_2 z_2^* - \lambda_2 (z_2^a - \Delta))/\lambda_1 \rceil, \quad \tilde{z}_2^a = z_2^a$$

and $\Delta_c = 0$ *in the associated problem* $P(\tilde{\mathbf{z}}^a, \mathbf{z}^*)$. *Similarly,*

$$\tilde{\mathbf{z}}^b = (\tilde{z}_1^b, \tilde{z}_2^b) \quad \textit{with} \quad \tilde{z}_1^b = z_1^b, \quad \tilde{z}_2^b = \lceil (\lambda_1 z_1^* + \lambda_2 z_2^* - \lambda_1 (z_1^b - \Delta_c))/\lambda_2 \rceil$$

and $\Delta = 0$ *in the associated problem* $P(\mathbf{z}^*, \tilde{\mathbf{z}}^b)$.

Proof. Since z^* is a non-supported optimal solution, it is placed above the segment $[\mathbf{z}^a, \mathbf{z}^b]$. By the definition of the objective function (it is parallel to the segment $[\mathbf{z}^a, \mathbf{z}^b]$), there will be no further points in the shaded region given in Figure 2b, except on the line through z^* parallel to $[\mathbf{z}^a, \mathbf{z}^b]$. Thus, the intersection points $\tilde{\mathbf{z}}^a$ and $\tilde{\mathbf{z}}^b$ of this line segment with the upper z_1-boundary, i.e., $z_1^b - \Delta_c$, and upper z_2-boundary, i.e., $z_2^a - \Delta$ of the current interval are valid boundary points. Further Pareto optimal solutions can only be found in the shaded regions shown in Figure 2c. Finally, the values of the intersection points can be rounded up, since the objective value is always integer, which concludes the proof. □

This new result may reduce the overall number of MIP iterations. In particular, empty intervals can be detected earlier and discarded without running the underlying MIP model. The influence of this enhancement to the performance of the binary search method is studied in Section 4.

2.5 Chebyscheff Norm Method

Chebyscheff norm methods use the weighted Chebyscheff distance of both objectives to an ideal point as objective function, i.e., $\min\{\max\{\beta|(z_1(\boldsymbol{\sigma}) - z_1^I|, (1 -$

$\beta)|z_2(\boldsymbol{\sigma}) - z_2^{\mathrm{I}}|\}\}$, for some weight parameter β. Such a method was first introduced by Eswaran et al. [10] for general bi-objective problems. Based on this, some specialized methods for bi-criteria combinatorial optimization were presented in Neumayer and Schweigert [27], Ralphs et al. [30].

For the BOPCSTP the resulting optimization problem can be modeled as MIP $P(\beta)$ by introducing an auxiliary variable ξ in the following way:

$$P(\beta): \qquad \min\{\xi \mid \xi \geq \beta \sum_{a \in A} c_a x_a,\ \xi \geq (1-\beta) \sum_{t \in T} r_t(1-y_t),\ (\mathbf{x}, \mathbf{y}) \in \mathcal{P}\}$$

Depending on the value of β, one obtains so called level-lines, on which Pareto optimal solutions lie. We used the weighted Chebyscheff norm (WCN) algorithm as described in [30]; see Algorithm 5.

Algorithm 5 WCN Algorithm

compute the ideal point $\mathbf{z}^{\mathrm{I}}$ and the Nadir point $\mathbf{z}^{\mathrm{N}}$
$I = \{[(z_1^{\mathrm{I}}, z_2^{\mathrm{N}}), (z_1^{\mathrm{N}}, z_2^{\mathrm{I}})]\}$
$Sol = \{(z_1^{\mathrm{I}}, z_2^{\mathrm{N}}), (z_1^{\mathrm{N}}, z_2^{\mathrm{I}})\}$
while $I \neq \emptyset$ **do**
  select an interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}] \in I$ and remove it from I
  $\beta \leftarrow z_2^{\mathrm{a}}/(z_2^{\mathrm{a}} + z_1^{\mathrm{b}})$
  $\boldsymbol{\sigma}^* \leftarrow$ argmin $P(\beta)$
  if $z_1(\boldsymbol{\sigma}^*) \neq z_1^{\mathrm{a}} \wedge z_1(\boldsymbol{\sigma}^*) \neq z_1^{\mathrm{b}}$ **then**
    $Sol \leftarrow Sol \cup \mathbf{z}(\boldsymbol{\sigma}^*)$
    $I \leftarrow I \cup \{[\mathbf{z}^{\mathrm{a}}, \mathbf{z}(\boldsymbol{\sigma}^*)], [\mathbf{z}(\boldsymbol{\sigma}^*), \mathbf{z}^{\mathrm{b}}]\}$

The algorithm is similar to the binary search in the objective space, the main difference lies in the definition of the underlying MIP problem. Instead of adding constraints based on the current interval to find new Pareto optimal solutions, only the weight parameter β of the Chebyscheff norm is modified. Choosing $\beta = z_2^{\mathrm{a}}/(z_2^{\mathrm{a}} + z_1^{\mathrm{b}})$, one either obtains a new Pareto optimal solution or one of the interval defining solutions (see Ralphs et al. [30] for further details). If a new non-dominated point is found in an iteration, it must lie in the current interval defined by $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$. Thus, the two new resulting intervals are added to the interval set.

Using the weighted Chebyscheff norm as objective can result in finding dominated solutions, since solutions with the same value for z_1 and different value for z_2 (and vice versa) can lie on the same level line, i.e., it is possible that the algorithm finds points which are only weakly Pareto optimal. To deal with this, two ways are presented by Ralphs et al. [30]: Either one considers the augmented Chebyscheff norm, i.e., the term $\rho(z_1(\boldsymbol{\sigma}) + z_2(\boldsymbol{\sigma}))$ for a small $\rho > 0$ is added to the objective function, or one needs to enumerate all solutions of $P(\beta)$ for each considered value of β. Since enumeration of all optimal MIP solutions is a time consuming task, we implemented the first variant with $\rho = 0.00001$.

2.6 Method of Sylva and Crema [38]

This method, which is also suitable for multi-objective problems, starts with solving the weighted sum problem $P^1(\lambda_1, \lambda_2) = P(\lambda_1, \lambda_2)$ for arbitrary nonnegative weights λ_1, λ_2, where $P(\lambda_1, \lambda_2)$ is defined as in Section 2.3 ($\lambda_1 = \lambda_2 = 1$

in our default setting). The solution $\boldsymbol{\sigma}_1^*$ of $P^1(\lambda_1, \lambda_2)$ with objective values $\mathbf{z}^1(\boldsymbol{\sigma}_1^*) = (z_1(\boldsymbol{\sigma}_1^*), z_2(\boldsymbol{\sigma}_1^*))$ is an efficient solution. To find another efficient solution, a set of constraints with two new binary variables k_1^2 and k_2^2, cutting off $\mathbf{z}^1(\boldsymbol{\sigma}_1)$ and all solutions dominated by it is added, yielding problem $P^2(\lambda_1, \lambda_2, \boldsymbol{\sigma}_1^*)$:

$$P^2(\lambda_1, \lambda_2, \boldsymbol{\sigma}_1^*): \quad \min\{\lambda_1 \sum_{a \in A} c_a x_a + \lambda_2 \sum_{t \in T} r_t(1 - y_t) \mid k_1^2 + k_2^2 \geq 1,\ (\mathbf{x}, \mathbf{y}) \in \mathcal{P}$$
$$z_\ell(\boldsymbol{\sigma}) - (z_\ell(\boldsymbol{\sigma}_1^*) - \Delta_\ell)k_\ell^2 - z_\ell^{\mathrm{N}}(1 - k_\ell^2) \leq 0, \quad k_\ell^2 \in \{0, 1\}, \ell = 1, 2\}$$

with $\Delta_1 = \Delta_c$ and $\Delta_2 = \Delta$. The added constraints work as follows: A new Pareto optimal solution needs to have less cost or a lower penalty than $\boldsymbol{\sigma}_1^*$, i.e., $z_1(\boldsymbol{\sigma}) \leq z_1(\boldsymbol{\sigma}_1^*) - \Delta_c$ or $z_2(\boldsymbol{\sigma}) \leq z_2(\boldsymbol{\sigma}_1^*) - \Delta$ must hold. Variables k_2^1 and k_2^2 ensure that this holds for one of the inequalities and that the other inequality is still valid by subtracting the corresponding value of the Nadir point. Using the solution $\boldsymbol{\sigma}_2^*$ of $P^2(\lambda_1, \lambda_2, \boldsymbol{\sigma}_1^*)$, a new problem $P^3(\lambda_1, \lambda_2, \boldsymbol{\sigma}_1^*, \boldsymbol{\sigma}_2^*)$ with three more constraints and two more binary variables k_1^3, k_2^3 is defined in the same way as above. This process is repeated, until for some t, the problem $P^t(\lambda_1, \lambda_2, \boldsymbol{\sigma}_1^*, \ldots, \boldsymbol{\sigma}_{t-1}^*)$ becomes infeasible.

We now present a simplification of this method in the bi-objective case: In this case, for a solution $\boldsymbol{\sigma}_i^*$ found in the current iteration, a new Pareto optimal solution must either have less costs or a lower penalty than $\boldsymbol{\sigma}_i^*$. Therefore, we can can replace the inequality $k_1^i + k_2^i \geq 1$ with an equality and the new constraints can be rewritten as follows:

$$z_1(\boldsymbol{\sigma}) - (z_1^{\mathrm{N}} - z_1(\boldsymbol{\sigma}_i^*) + \Delta_c)k^i \leq z_1(\boldsymbol{\sigma}_i^*) - \Delta_c,$$
$$z_2(\boldsymbol{\sigma}) - (z_2^{\mathrm{N}} - z_2(\boldsymbol{\sigma}_i^*) + \Delta)(1 - k^i) \leq z_2(\boldsymbol{\sigma}_i^*) - \Delta.$$

Correctness of this new set of constraints can be easily checked by inserting the two possible values of k^i: If $k^i = 0$, the solution must have lower costs than the current solution, due to the first constraint, while the second constraint poses no restriction on the penalty, since every Pareto optimal solution has a penalty lower or equal to z_2^{N}. The case for $k^i = 0$ is analogous.

The method has the advantage that only a single MIP needs to be solved for each non-dominated point (and one infeasible MIP at the end). These MIPs, however, become more difficult to solve at every step, due to the added constraints and variables.

3 Acceleration Methods

Next, we discuss potential speed-up methods for the described solution approaches, which are based on exploiting and recycling the information contained in a set of already identified Pareto optimal solutions. Depending on the approach, the already identified solutions provide lower and/or upper bounds for the optimal value of the current iteration. Moreover, the "special constraints" of the underlying approaches (e.g., the interval defining constraints) also allow to derive some strengthening inequalities.

3.1 Cover and Lifted Cover Inequalities

The interval-based constraint $\sum_{t\in T} r_t(1-y_t) \geq \underline{r}$, which appears in the binary search in the objective space, the two-phase method and the Chebyscheff norm method (cf. Sections 2.3, 2.4, and 2.5), is a knapsack-type constraint, since it can be rewritten as $\sum_{t\in T} r_t y_t \leq \sum_{t\in T} r_t - \underline{r}$. It is well known that such knapsack constraints can be used to derive strengthening cover inequalities to a MIP (see, e.g., Kaparis and Letchford [19]). Let $R = \sum_{t\in T} r_t - \underline{r}$ and $C \subset T$ be a minimal cover, i.e., C is a minimal set s.t. $\sum_{t\in C} r_t > R$. Then the associated cover inequality is

$$y(C) \leq |C| - 1, \tag{4}$$

where $y(T') = \sum_{t\in T'} y_t$ for a given $T' \subseteq T$. This cover inequality means that at least one $t \in C$ must not be selected in a solution. By defining $C' = \{u \in T \setminus C : r_u \geq \max_{s\in C} r_s\}$, we can strengthen (4) and derive the extended cover inequality

$$y(C \cup C') \leq |C| - 1. \tag{5}$$

Separation of cover inequalities is a weakly NP-hard problem, however, once we detect a cover C, it is possible to further strengthen these inequalities by considering the family of lifted cover inequalities:

$$y(C) + \sum_{t\in C'} \pi_t y_t \leq |C| - 1. \tag{6}$$

We apply the procedure from Balas and Zemel [3], which was also used in Bérubé et al. [4] for the ϵ-constraint method. The lifting coefficients $\pi_t, \forall t \in C'$, are obtained in the following way: Assume that the elements from C are indexed by $1, \ldots, |C|$, and sorted in an non-increasing order according to their revenues. Then, the coefficients π_t are calculated as follows:

$$\pi_t = h \quad \text{where} \quad h = \text{argmin}_{k\in\{1,\ldots,|C|\}} \sum_{l=1}^{k} r_l \leq r_t < \sum_{l=1}^{k+1} r_l,$$

where $r_{|C|+1} = \infty$. See [3] for conditions under which these constraints are facet defining.

Likewise, the other interval-based constraint involving the second objective, i.e., the $\sum_{t\in T} r_t(1-y_t) \leq \bar{r}$, which appears in all methods except the one of Sylva and Crema, is also a knapsack-type constraint. If $D \subset T$ is a minimal cover, i.e., $V \setminus D$ is a maximal set s.t. $\sum_{t\notin D} r_t < \sum_{t\in V} r_t - \bar{r}$, we end up with the following cover inequality associated with D:

$$\sum_{t\in D} (1 - y_t) \leq |D| - 1, \tag{7}$$

which can be rewritten as $y(D) \geq 1$.

This inequality implies that at least one $t \in D$ must be selected in a solution. It can be extended and lifted in the same spirit as described above, i.e., if $D' = \{u \in T : r_u \geq \max_{s\in D} r_s\}$, we obtain

$$y(D \cup D') \geq |D'| + 1. \tag{8}$$

3.2 Visit Inequalities

Visit inequalities are another family of knapsack-type inequalities that cut off previously "visited" solutions. For example, Bérubé et al. [4] show how to use these inequalities in combination with the ϵ-constraint method. We extend this idea and show that these inequalities can also be used in combination with the binary search in the objective space. Visit inequalities can be derived from lower and upper bound solutions defined with respect to the second objective. Let T_S^{U} be the terminal set associated with a solution, which defines an upper bound w.r.t. the second objective in the current iteration. Since we need to find a solution with smaller penalty in the current iteration, at least one terminal not in T_S^{U} must be part of the solution. Thus we can add the inequality

$$y(T \setminus T_S^U) \geq 1. \tag{9}$$

Furthermore, when given a terminal set T_S^{L} associated with a solution defining a lower bound w.r.t. the second objective, we can add

$$y(T_S^L) \leq |T_S^L| - 1. \tag{10}$$

This inequality is valid since at least one of the terminals must not be selected. Both types of visit inequalities can be lifted in the same way as described in the previous subsection.

3.3 Cutset-Cover Inequalities

Consider again the cover inequalities (7). Since at least one of the terminals in D must be connected to the root in a solution, the following family of cutset-cover inequalities (see, e.g., Gollowitzer et al. [14]) is valid for the BOPCSTP:

$$x(\delta^-(W)) \geq 1 \quad \forall W \subseteq V, D \subseteq W. \tag{11}$$

The separation of these constraints during a single MIP iteration (branch-and-cut) will be discussed in Section 4.2. Cutset-cover inequalities are also applied to the visit inequalities (9).

3.4 Asymmetry Constraints

Since every feasible subtree can be represented in various ways as a rooted arborescence, our directed cutset model for the PCSTP introduces a lot of symmetries. To get rid of them, and to achieve a bijection between arborescences and PCSTP solutions, we use the following constraints (see also Ljubić et al. [23]):

$$x_{0j} \leq 1 - y_i \qquad \forall i < j, j \in T. \tag{12}$$

These inequalities make sure that the root of each arborescence is the terminal node with the smallest index.

3.5 Path Inequalities

The following family of inequalities are derived from upper bound constraints $\sum_{t \in T}(1-y_t)r_t \leq \bar{r}$. Consider a breadth-first-search tree of G rooted at a terminal $t \in T$, and denote with V_h^t the nodes h arcs away from t in it. Let h_t be the

index such that $\sum_{0\leq i\leq h_t}\sum_{j\in V_i^t} r_j \geq \sum_{j\in T} r_j - \bar{r}$ and $\sum_{0\leq i\leq h_t-1}\sum_{j\in V_i^t} r_j < \sum_{j\in T} r_j - \bar{r}$. Then, at least one arc crossing from $V_{h_t-1}^t$ to $V_{h_t}^t$ must be selected, if terminal t is chosen as root of the arborescence, which is stated as:

$$\sum_{(i,j):i\in V_{h_t-1}^t, j\in V_{h_t}^t} x_{ij} \geq x_{0t} \quad \forall t \in T. \tag{13}$$

Note that the asymmetry constraints can be used to strengthen the path inequalities: If terminal t is a root of the arborescence, each terminal t' with index smaller than t cannot be part of this solution. Thus, when constructing the sets V_h^t for the path inequalities, these terminals can be ignored, and also the coefficients next to arcs adjacent to such t' can be down-lifted to zero.

3.6 Feasible Starting Solutions

Further acceleration of a single MIP iteration can be achieved by providing a starting solution to the MIP model. Potential starting solutions are kept in a list L, which is updated during the solution process to contain solutions that are non-so-far dominated. L is initialized with all non-optimal incumbent solutions of the first iteration. In the following iterations, potential starting solutions are added to L in two ways: (i) For the Pareto optimal solution computed in the previous iteration, a terminal t not in this solution with minimum connection costs (i.e., a closest terminal to this Pareto optimal solution with respect to edge costs) is added. If there are several terminals with the same minimum connection costs, one with the highest revenue is chosen, in case of ties the terminal with minimal index is chosen. (ii) All so-far non-dominated incumbent solutions discovered in the current iteration are added to L.

The starting solution for an iteration is then chosen from L as follows: The elements of L are evaluated using the objective function of the current iteration and the one with the best objective value is chosen. Note that for the two-phase method, binary search in the objective space and the Chebyscheff norm method, the solution of the current iteration needs to lie in the current interval $[\mathbf{z}^{\mathrm{a}}, \mathbf{z}^{\mathrm{b}}]$, thus only solutions of L, which lie inside this interval are considered. For the ϵ-constraint method and the method of Sylva and Crema, we restrict the search to solutions with a cost value, which is at most $100\Delta_c$ larger than the solution of the previous iteration.

3.7 Guiding the Branching Process

For guiding the branching process, we consider two strategies in which different priorities are associated with decision variables. In the basic branching strategy, no information from the iterative ILP framework is used, i.e., branching is guided as for the single-objective PCSTP: The highest branching priority is associated with terminal-variables, followed by variables for potential Steiner nodes, followed by arc-variables.

In the advanced branching strategy, we collect information gained in the previous iterations to guide the branching process. In the first iteration, branching priorities are initialized as above. Every time a terminal or Steiner node occurs in a Pareto optimal solution of an iteration, the branching priority of the corresponding variable is increased for all following iterations.

3.8 Cut Pools

The idea of cut pools for bi-objective MIP approaches is to store cuts from previous iterations and reuse them [32]. We implemented a cut pool for directed cutset constraints in the following way: During the first branch-and-cut iteration, all violated cuts detected by the maximum-flow algorithm are stored in the current cut pool. In each following iteration, violated cutset constraints are separated as follows: (i) check, if there exist violated cuts in the cut pool created in the previous iteration (ii) add all these cuts and delete them from the current cut pool; in case no such cuts exist, call the separation routine (cf. Section 4.2). In either case, all detected violated cuts are inserted in the new cut pool that is used in the next iteration.

4 Computational Results

In this section, we describe further implementation details of the presented MIP approaches and provide a detailed analysis of our computational experiments. The computational results are obtained using a single core of an Intel Xeon X5500 with 2.67Ghz and 24GB RAM. CPLEX 12.4, which was used as MIP solver, was configured as follows: All CPLEX cuts were disabled and the dual simplex algorithm with steepest edge pricing, as recommended by Koch and Martin [20], was used.

4.1 Benchmark Instances

As test instances we used the *wildlife corridor instances* obtained using the instance generator of Dilkina and Gomes [8], which is also used in Álvarez Miranda et al. [1]. These instances consist of 4-grid graphs with costs and revenues on the nodes. The nodes describe land parcels, which can be purchased by paying the costs, and the revenues indicate the wildlife benefit of the parcel. The instances can be transformed into PCSTP instances by setting the cost of every incoming arc (i, v) of a node v to the cost of v. In every instance, there are also some land parcels $i \in L$, which must be part of every feasible solution. Thus, we add $y_i = 1, \forall i \in L$, to our formulation, and the node in L with the smallest index is used as root node. The obtained instances are rooted directed graphs with asymmetric arc weights. Hence, there is no artificial root node and the asymmetry constraints (12) are not valid in this case.

The instance sets will be referred to with 15-U-$\mathcal{T}$, which indicates the settings used in the instance generator of Dilkina and Gomes [8]. The underlying graphs are 15×15 grid graphs. Costs and revenues are uncorrelated, i.e., they are chosen uniformly at random between zero and ten. In every instance, there are three land parcels, which must be part of every solution, i.e., $|L| = 3$. When $\mathcal{T} = R$, random nodes are chosen as these parcels, and when $\mathcal{T} = F$, one of these parcels is randomly chosen out of all nodes, the other two are the upper left and the lower right node of the grid graph. Each instance set consists of twenty instances and individual instances will be referred to with an index number, e.g., 15-U-F-4 for the fourth instance from set 15-U-F.

Moreover, we also used the PCSTP instance sets C and D from Ljubić et al. [23]. Both sets contain 20 graphs with two settings for the revenues – these instances will be referred to as C1-A, C1-B and so on. Instances from set C

have $|V| = 500$, $625 \leq |E| \leq 12500$ and $5 \leq |T| \leq 250$, while $|V| = 1000$, $1250 \leq |E| \leq 25000$, and $5 \leq |T| \leq 500$, holds for instances of set D.

Preprocessing. Before running an iterative MIP framework, the instances are preprocessed using the basic preprocessing steps known for the PCSTP: The *least-cost* test, i.e., remove all edges (i, j), where c_{ij} is larger than the cost of the shortest path between i and j, the *degree-one* test, i.e., remove all potential Steiner nodes with degree one and their adjacent edge and the *degree-two* test, i.e., replace each potential Steiner node i of degree two and its two adjacent edges $(i, j), (i, k)$ by a new edge (j, k) with cost $c_{ij} + c_{ik}$. These reduction steps are applied iteratively, until no more nodes or edges get removed by them. Note that other known preprocessing techniques that delete or modify terminals can not be used in combination with iterative MIP frameworks for the BOPCSTP.

4.2 Solving a Single MIP iteration: Branch-and-Cut Algorithm

Each single iteration of the five MIP frameworks considered is solved by means of a branch-and-cut algorithm. The latter algorithm basically solves the underlying MIP problems defined above ($P(\epsilon)$, $P(\lambda_1, \lambda_2)$,...) by dynamically separating: connectivity constraints, cover inequalities, visit inequalities and cutset cover inequalities. Path inequalities are inserted at the MIP initialization phase. In this section we describe the main ingredients of a single branch-and-cut iteration.

Constraint Separation. For each fractional solution occurring during the course of a branch-and-cut tree, we identify violated connectivity constraints using the maximum flow algorithm of Cherkassky and Goldberg [6]. As suggested in Koch and Martin [20] and Ljubić et al. [23], we also use *nested*, *back* and *minimum cardinality cuts*. Note that we only search for violated constraints for terminals $t \in T$, if $y_t^* \geq 0.5$ in the current LP-solution $(\mathbf{x}^*, \mathbf{y}^*)$ since preliminary tests showed that this strategy yields computational advantages.

If the current LP-solution $(\mathbf{x}^*, \mathbf{y}^*)$ is integral, we identify possibly existing violated connectivity constraints as follows: By breadth-first-search, identify the node set V_0^* of nodes reachable from the root node 0 in the support graph $G^* = (V \cup \{0\}, (i, j) \in A : x_{ij}^* = 1)$. If there exists a terminal t with $y_t^* = 1$, but $t \notin V_0^*$, $\sum_{(i,j) \in A: i \in V_0^*, j \notin V_0^*} x_{ij} \geq y_t$ is a violated cutset constraint. We also add nested cuts by setting $x_{ij}^* = 1$ for all arcs (i, j) contained in an added cut and repeating the search.

The separation of a cover inequality amounts to solving a knapsack problem. Since this strategy is very costly, we use the heuristic described in Section 3.3 of Kaparis and Letchford [19]. The separation of the cutset-cover inequalities (11) can also be done with a maximum flow algorithm, but this time on a slightly modified graph: For each identified cover $D \subset T$, add a sink node t and arcs (t', t), $\forall t' \in D$. Let the capacity of arcs $(i, j) \in A$ be x_{ij}^* and the capacity of auxiliary arcs (t', t) be one. If the value of the maximum 0-t flow is less than one, the associated cut W defines a violated cutset-cover inequality $x(\delta^-(W)) \geq 1$. Again we apply nested cuts and back cuts. Visit inequalities are uniquely determined by the lower and upper bounding solution and lifted as described in Section 3.1. Path inequalities are determined in polynomial time

by running the breath-first search algorithm as described in Section 3.5 and are added at the initialization of the MIP. All other mentioned inequalities are separated in the following order: (i) violated cutset inequalities from the cut pool, (ii) the dynamically separated cutset inequalities, (iii) visit and visit-cover inequalities, (iv) cover and cutset-cover inequalities. If a violated inequality is found at some step, the LP is resolved before the remaining separation steps are performed. Moreover, when the current LP-solution is integer, only steps (i) and (ii) are performed, i.e., only cutset inequalities are considered.

Branch-and-Cut Configuration. We initialize each MIP model by adding flow-balance constraints (14), subtour elimination constraints for subtours of length two (15), as well as asymmetry constraints, cf. Section 3.4.

$$x(\delta^-(i)) \leq x(\delta^+(i)) \qquad \forall i \in V \setminus (T \cup \{0\}) \tag{14}$$

$$x_{ij} + x_{ji} \leq y_i \qquad \forall i \in V, (i,j) \in A \tag{15}$$

Since the wildlife corridor instances are 4-grid graphs, for them we also add all *4-cycle inequalities*

$$x(A(C)) \leq y(C \setminus i) \qquad \forall i \in C,\ \forall C \in \mathcal{C}_4,$$

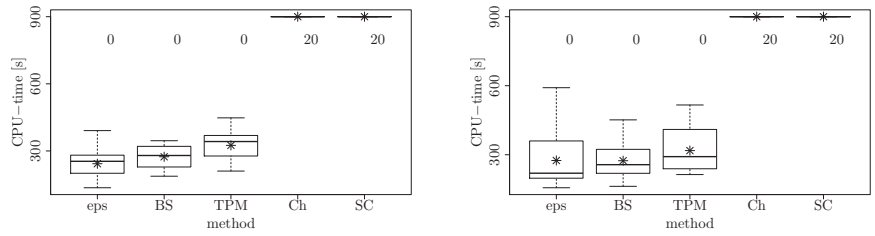
to our formulation, where $\mathcal{C}_4$ denotes the set of all 4-cycles C of graph G and $A(C)$ is the arc set associated with C.

4.3 Analysis of the Computational Results

In the following, we will abbreviate the acceleration methods described in Section 3 as follows: `V` stands for the visit and cover inequalities with the associated cutset inequalities, `P` for the path inequalities, `S` stands for using feasible starting solutions, `B` for the advanced branching strategy and `C` for the cutpool. If none of the methods is used, we will write `N`. For the binary search in the objective space, setting `N−` means `N` without using interval tightening, cf. Proposition 1.

At first, we concentrate on the wildlife instances and present a computational comparison of all solution methods described in Section 2. For all methods, the best possible combination of the acceleration methods was used (we will provide a more detailed analysis of the effects of the acceleration methods for the ϵ-constraint method and binary search in the objective space below). The combinations are as following: `SBC` for the ϵ-constraint method (*eps*) and the method of Sylva and Crema (SC), `FBCV` for the binary search in the objective space (BS), two-phase method (TPM) and the Chebyscheff norm method (Ch). Note that for the two-phase method and the Chebyscheff norm method, visit inequalities are not valid and thus `V` indicates only cover and cutset-cover inequalities. For the binary search in the objective space, we used the tightened intervals, cf. Proposition 1. Branching priorities are initialized with value two for terminals, value one for Steiner nodes and zero for arc variables. Priorities are increased by one each time a node appears in a Pareto optimal solution of an iteration.

Figures 3a and 3b show boxplots of the CPU-times for instance sets 15-U-F and 15-U-R, respectively. The stars in the boxes indicate the average CPU-times over 20 instances, and the numbers shown above the boxes explain for



(a) 15-U-F (b) 15-U-R

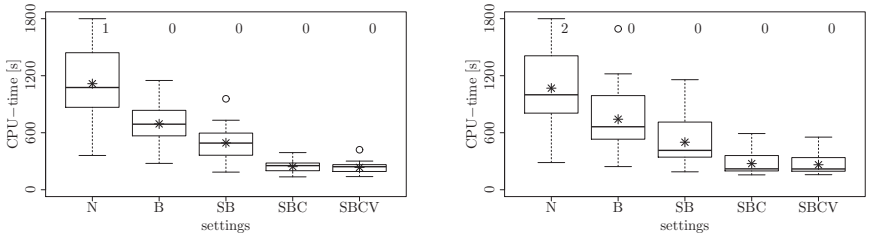
Figure 3: All five iterative MIP approaches with best settings (time limit set to 1 800 seconds)

how many out of the 20 instances, we were not able to discover the complete Pareto front within the given time limit of 1 800 seconds. BS and *eps* clearly exhibit better performance than the other three methods. Using Ch or SC, no instance could be solved within the given time limit (even when raising the limit to 7 200 seconds). For Ch, often a single ILP iteration requires an extremely high number of branch-and-bound nodes and therefore the whole time limit is exceeded. This may be due to the min-max objective function and/or potential numerical problems caused by the non-integrality of parameters β and ρ. In SC, every iteration takes more and more time, i.e., the need of adding constraints and variables in each iteration is manifested in the runtime. TPM does not suffer from either of these problems, however, it cannot exploit its strength of using a polynomial time algorithm in the first phase, since the PCSTP is NP-hard. Thus it ends up as a sort of hybrid of *eps* and BS with worse performance. In contrast to BS, for example, in order to prove emptiness of an interval, the same Pareto optimal point may be computed twice within the TPM.

A closer comparison of *eps* and BS methods is given in Figures 4 and 6, respectively. Figures 4a and 4b provide a more detailed analysis of the effect of the acceleration methods in combination with the ϵ-constraint method. With respect to the considered acceleration methods, we observe that branching priorities, feasible starting solutions and the cutpool all have a positive effect. On the other hand, using cover, visit and the respective cutset inequalities has no or negative effect on the computational performance. This can be partially explained by the fact that the runtime for an individual MIP is already very short when using the other acceleration methods. The fraction of weakly dominated solutions found by the *eps* method is around ten percent for all settings.

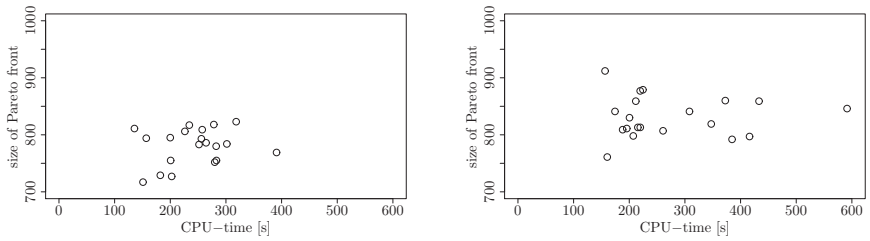
Figures 5a and 5b plot the runtime versus the size of the Pareto front (using *eps* with setting `SCB`). While for set 15-U-F, there seems to be a correlation between the size of the Pareto front and the runtime, the picture is not so clear for 15-U-R. For the set 15-U-F, instance 15-U-F-19 has the largest Pareto front consisting of 823 points and the largest front for set 15-U-R has 912 points and is obtained from instance 15-U-R-15. The highest computation times are needed for instances 15-U-F-20 and 15-U-R-20 with 381 seconds and 591 seconds, respectively.

Figures 6a and 6b detail the effects of the acceleration methods in combination with binary search in the objective space. Interestingly, for BS, using



(a) 15-U-F (b) 15-U-R

Figure 4: ϵ-constraint method and acceleration methods



(a) 15-U-F (b) 15-U-R

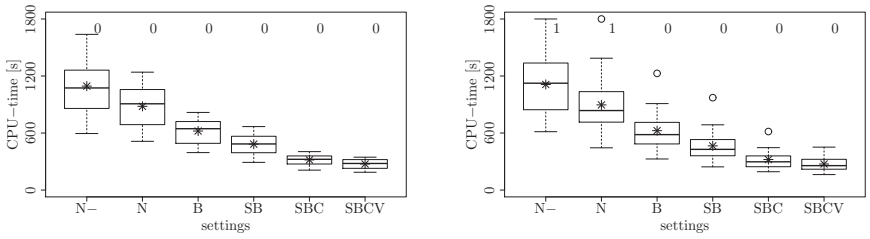
Figure 5: Runtime vs. size of Pareto front, ϵ-constraint method with setting `SCB`

visit, cover and the associated cutset inequalities has a larger effect than for *eps*. This may be explained by the fact that, in contrast to *eps*, cover (6) and visit inequality (9) can be defined from upper and lower interval bounds. The runtime for proving emptiness of an interval turned out to be relatively high. In some cases, it was four times higher than the average runtime needed to find a point on the Pareto front in an interval.

Figures 7a and 7b detail the performance of the solution methods for instance sets C and D. Time limits were 1 800 and 7 200 CPU-seconds, respectively. The general trends are similar to the one for the wildlife instances, i.e., *eps* and BS clearly outperform the other methods. For these instance sets, however, some instances with a rather small number of terminals could be solved with Ch and SC.

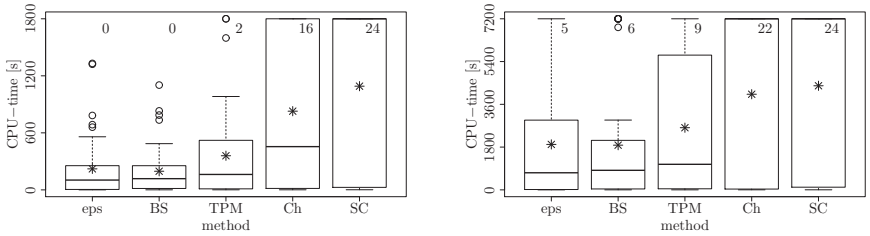
Figures 8a and 8b give a more in-depth look for *eps* in combination with acceleration methods for these instance sets, this time also the path inequalities are tested. However, they result in a performance loss. The effects of the other acceleration methods are similar to the effects observed for the wildlife instances, branching priorities, starting solutions and the cut pool improve the performance, while cover, visit and the associated cutset inequalities have little effect.

Figures 9a and 9b show a plot of the runtime vs. the size of the Pareto front using *eps* with setting `SCB`. In contrast to the wildlife instances, where the underlying graphs all had the same size, we see the influence of the graph attributes (i.e., density and number of terminals) on the runtime. For the set



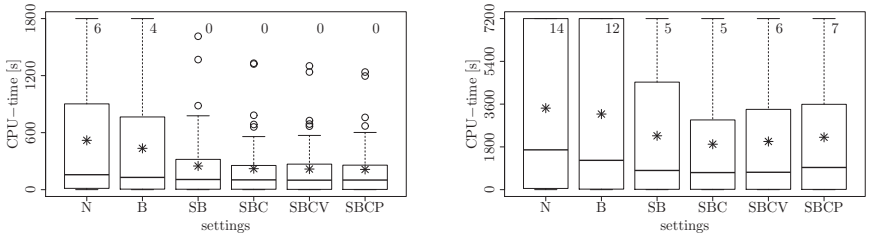
(a) 15-U-F (b) 15-U-R

Figure 6: Binary search in the objective space and acceleration methods



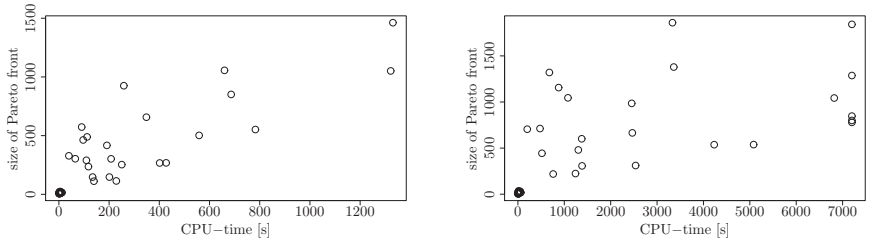
(a) Set C (b) Set D

Figure 7: All five approaches with best settings



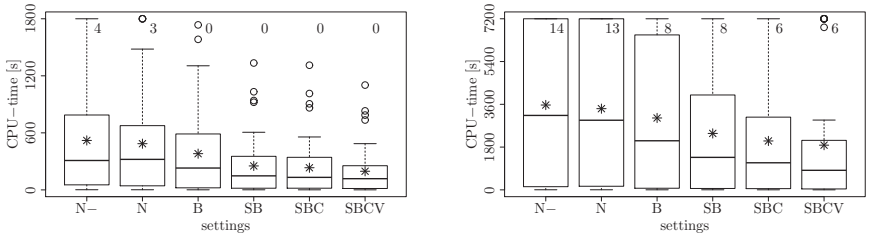
(a) Set C (b) Set D

Figure 8: ϵ-constraint method and acceleration methods



(a) Set C (b) Set D

Figure 9: Runtime vs. size of Pareto front, ϵ-constraint method with setting `SCB`



(a) Set C (b) Set D

Figure 10: Binary search in the objective space and acceleration methods

C, instance C5-B with 625 edges and 250 terminals has the largest Pareto front consisting of 1 462 points. The largest front for set D discovered within the time limit has 2 141 points and is obtained from instance D5-A, which has 1 250 edges and 500 terminals.

Figures 10a and 10b detail the effects of BS in combination with acceleration methods. Again, the same picture as for the wildlife instances emerges, every acceleration method has some positive effect on the runtime.

5 Conclusion

In this article, we introduced the bi-objective prize-collecting Steiner tree problem (BOPCSTP). As opposed to the single-objective variant where net-worth (i.e., the difference between revenues and costs) is maximized, in the BOPCSTP revenues and costs are considered as individual objective functions. The goal of the BOPCSTP is to find one solution for each point on the Pareto front. After formally introducing the problem we showed how to apply five iterative mixed-integer programming frameworks to the BOPCSTP. An important aspect of this work was to study acceleration methods that exploit and recycle information gained during the previous iterations and to apply them to the considered MIP frameworks.

A computational study was performed on instances borrowed from wildlife corridor design and from the single-objective PCSTP. Wildlife corridor design

is a problem for which the BOPCSTP is especially suitable, since costs and revenues are not given in the same units and thus net-worth maximization gives less insights into the nature of the problem. The obtained results show that the ϵ-constraint method and the binary search in the objective space clearly outperform all other considered methods, i.e., the two-phase method, the weighted Chebyscheff norm based approach, and the method by Sylva and Crema. In particular, the proposed acceleration methods turn our to be of huge importance for the practical performance, since they significantly speed-up the solution process and allow for solving more instances within the given time limits.

Acknowledgments

Markus Leitner is supported by the Austrian Science Fund (FWF) under grant I892-N23 and Ivana Ljubić is supported by the APART Fellowship of the Austrian Academy of Sciences. This support is greatly acknowledged.

References

[1] E. Álvarez Miranda, I. Ljubić, and P. Toth. Exact approaches for solving robust prize-collecting Steiner tree problems. *Eur. J. Oper. Res.*, 229(3): 599–612, 2013.

[2] A. Archer, M. H. Bateni, M. T. Hajiaghayi, and H. J. Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. *SIAM J. Comput.*, 40(2):309–332, 2011.

[3] E. Balas and E. Zemel. Facets of the knapsack polytope from minimal covers. *SIAM J. on Appl. Math.*, 34(1):119–148, 1978.

[4] J. F. Bérubé, M. Gendreau, and J. Y. Potvin. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *Eur. J. Oper. Res.*, 194(1):39–50, Apr. 2009.

[5] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Programming*, 59(1):413–420, 1993.

[6] B. V. Cherkassky and A. V. Goldberg. On implementing the push - relabel method for the maximum flow problem. *Algorithmica*, pages 390–410, 1997.

[7] A. S. da Cunha, A. Lucena, N. Maculan, and M. G. C. Resende. A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Appl. Math.*, 157(6):1198–1217, 2009.

[8] B. Dilkina and C. Gomes. Solving connected subgraph problems in wildlife conservation. In *CPAIOR 2010*, volume 6140 of *Springer Lecture Nodes in Comput. Sci.*, pages 102–116, 2010.

[9] M. Ehrgott and M. M. Wiecek. Mutiobjective Programming. In F. S. Hillier, editor, *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *Internat. Ser. in Oper. Res. & Management Sci.*, pages 667–708. Springer New York, 2005.

[10] P. K. Eswaran, A. Ravindran, and H. Moskowitz. Algorithms for nonlinear integer bicriterion problems. *J. of Optim. Theory and Appl.*, 63(2):261–279, 1989.

[11] P. Feofiloff, C. G. Fernandes, C. E. Ferreira, and J. C. de Pina. Primal-dual approximation algorithms for the prize-collecting Steiner tree problem. *Inf. Process. Lett.*, 103(5):195–202, 2007.

[12] M. Fischetti. Facets of two Steiner arborescence polyhedra. *Math. Programming*, 51:401–419, 1991.

[13] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 144–191, 1997.

[14] S. Gollowitzer, B. Gendron, and I. Ljubić. A cutting plane algorithm for the capacitated connected facility location problem. *Comp. Opt. and Appl.*, 2013. doi: 10.1007/s10589-013-9544-9.

[15] H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *A. of Oper. Res.*, 52(4):209–230, 1994.

[16] M. Haouari, S. B. Layeb, and H. D. Sherali. The prize collecting Steiner tree problem: models and Lagrangian dual optimization approaches. *Comp. Opt. and Appl.*, 40(1):13–39, 2008.

[17] D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: theory and practice. In D. B. Shmoys, editor, *SODA*, pages 760–769. ACM/SIAM, 2000.

[18] N. Jozefowiez, G. Laporte, and F. Semet. A generic branch-and-cut algorithm for multiobjective optimization problems: Application to the multi-label traveling salesman problem. *INFORMS J. on Comp.*, 24(1):554–564, 2012.

[19] K. Kaparis and A. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Math. Programming*, 124(1-2):69–91, 2010.

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

[21] J. Lemesre, C. Dhaenens, and E. G. Talbi. Parallel partitioning method (PPM): A new exact method to solve bi-objective problems. *Comput. & Oper. Res.*, 34(8):2450–2462, Aug 2007.

[22] M. S. Levin and R. I. Nuiriakhmetov. Multicriteria Steiner tree problem for communication network. *Inform. Technology in Engrg. Systems*, 3:199–209, 2009.

[23] I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Math. Programming*, 105:427–449, 2006.

[24] A. Lucena and M. G. C. Resende. Generating lower bounds for the prize collecting Steiner problem in graph. *Electronic Notes in Discrete Math.*, 7: 70–73, 2001.

[25] L. Martins and N. G. Ferreira. A bi-criteria approach for Steiner's tree problems in communication networks. In *Proc. of the 2011 Internat. Workshop on Modeling, Analysis, and Control of Complex Networks*, pages 37–44. ITCP, 2011.

[26] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *Eur. J. Oper. Res.*, 107 (3):530–541, June 1998.

[27] P. Neumayer and D. Schweigert. Three algorithms for bicriteria integer linear programs. *OR Spectrum*, 16(4):267–276, 1994.

[28] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *Eur. J. Oper. Res.*, 185(2):509–533, 2008.

[29] A. Raith and M. Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Comput. & Oper. Res.*, 36(6):1945–1954, 2009.

[30] T. K. Ralphs, M. J. Saltzman, and M. M. Wiecek. An improved algorithm for biobjective integer programming and its application to network routing problems. *A. of Oper. Res.*, 147(1):43–70, Sep 2006.

[31] R. M. Ramos, S. Alonso, J. Sicilia, and C. González. The problem of the optimal biobjective spanning tree. *Eur. J. Oper. Res.*, 111(3):617–628, 1998.

[32] J. Riera-Ledesma and J. J. Salazar-González. The biobjective travelling purchaser problem. *Eur. J. Oper. Res.*, 160(3):599–613, 2005.

[33] S. Ruzika and H. W. Hamacher. A survey on multiple objective minimum spanning tree problems. In J. Lerner, D. Wagner, and K. A. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 104–116. Springer, 2009. ISBN 978-3-642-02093-3. doi: 10.1007/978-3-642-02094-0_6.

[34] D. Schweigert and P. Neumayer. A reduction algorithm for integer multiple objective linear programs. *Eur. J. Oper. Res.*, 99(2):459–462, 1997.

[35] A. Segev. The node-weighted steiner tree problem. *Networks*, 17(1):1–17, 1987.

[36] F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS J. Comput.*, 20(3):472–484, 2008.

[37] S. Steiner and T. Radzik. Solving the biobjective minimum spanning tree problem using a k-best algorithm. Technical report, King's College London, 2003.

[38] J. Sylva and A. Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *Eur. J. Oper. Res.*, 158(1):46–55, 2004.

[39] E. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Comput. and Decision Sci.*, 20(2):149–165, 1995.

[40] M. Visée, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. *J. of Global Optim.*, 12(2):139–155, 1998.

[41] M. Vujošević and M. Stanojević. A bicriterion Steiner tree problem on graph. *Yugoslav J. Of Oper. Res.*, 13(1):25–33, 2003.