An algorithmic framework in the criterion space for bi-objective mixed integer linear problems

Lavinia Amorosi[1] and Marianna De Santis[2]

Abstract

We propose a flexible and general algorithmic framework for solving bi-objective mixed integer linear programming problems. The Pareto frontier of these problems can have a complex structure, as it can include isolated points, open, half-open and closed line segments. Therefore, its exact detection is an achievable though hard computational task. Operating in the criterion space, our algorithm features a simple structure that relies on an ordered exploration of the feasible region, making it relatively easy to implement. Our framework is oracle-based and iteratively solves two classes of subproblems: mixed integer and bi-objective linear programming problems. We introduce a family of cuts, called *extreme-inequalities*, to exclude dominated regions in the image space and improve the performance of the algorithm. Under specific assumptions, we show that the exact Pareto frontier can be detected in a finite number of iterations. Experimental results on a test-bed of instances and a comparison with the Triangle Splitting Method, a renowned criterion space method for bi-objective mixed integer problems, is presented, showing the notably good performance of our approach in terms of accuracy of the Pareto frontier detected and in terms of efficiency for medium size instances.

Keywords:
Bi-objective Programming, Mixed Integer Linear Programming, Criterion Space Algorithm.
Mathematics subject classifications (MSC 2010): 90C11, 90C29, 90C57.

[1]Dipartimento di Scienze Statistiche, Sapienza Università di Roma, Piazzale Aldo Moro 5, 00185, Roma, Italy, (lavinia.amorosi@uniroma1.it)

[2]Dipartimento di Ingegneria dell'informazione, Università di Firenze, Via di Santa Marta 3, 50139, Firenze, Italy, (marianna.desantis@unifi.it)

1. Introduction

Optimization problems arising from real world applications increasingly require the introduction of multiple criteria against which the possible solutions are evaluated to identify the best decision(s). In particular, under the sustainability paradigm that is increasingly spreading across all sectors, new problems of a multi-objective nature arise or classic problems in the mathematical programming literature are reformulated in multi-objective terms. Just to cite a few examples, in reverse logistics network design, both minimization of the total transportation cost and minimization of the total tardiness of returned products from the target delivery duration are optimized simultaneously, (Yıldız and Soylu, 2019). In the energy sector the design and management of renewable energy storage systems is performed by taking into account both installation costs and energy self-sufficiency (Amorosi et al., 2022). In the mobility sector, the planning of walking routes is done by minimizing travel times but also maximizing the air quality of the route (Wang et al., 2023). In supply chain, maximization of profit and minimization of CO2 emissions are simultaneously performed (Caramia and Pizzari, 2023). Consequently, from a methodological point of view, multi-objective programming is receiving increasing attention. Although there is still a considerable gap between the complexity of the models deriving from real world applications and the solution techniques available, research in this area has produced increasingly efficient algorithms capable of dealing with instances of increasing size (Halffmann et al., 2022).
Depending on the continuous, discrete or mixed nature of the decision variables, and on the absence or presence of a combinatorial structure underlying the problem, we can distinguish multi-objective optimization (MOO) problems from multi-objective combinatorial optimization (MOCO) problems. As regards the exact solution techniques capable of generating the entire Pareto frontier of the problem, we can distinguish between decision space algorithms and objective space (or criterion space) algorithms. Since the number of objectives is usually smaller than the number of variables, objective space algorithms have the advantage of working in a lower dimensional space. In the context of multi-objective optimization problems, the class of multi-objective linear programming (MOLP) problems has so far been the most studied and for which solution techniques have also been developed as extensions of those for the single objective case (Ehrgott, 2000). The class of multi-objective integer or binary linear optimization problems has also been

the focus of many works from the literature (Ehrgott and Gandibleux, 2000). In particular, solution algorithms have been developed for the bi- or multi-objective versions of classical combinatorial optimization problems such as the shortest path, the minimum spanning tree and the minimum cost flow problems (see e.g. (Raith and Ehrgott, 2009a; Steiner and Radzik, 2008; Amorosi and Puerto, 2022; Raith and Ehrgott, 2009b)). Recently, works dealing with bi- and multi-objective mixed integer nonlinear optimization problems have also been proposed (see e.g. (De Santis et al., 2020a, 2022, 2020b, 2025; Eichfelder and Warnow, 2023; de Castro and Wiecek, 2025)). In this work we address bi-objective mixed integer linear problems of the following form:

$$\begin{array}{ll} \min & z(x) = (z_1(x), z_2(x))^T \\ \text{s.t.} & Ax \leq b, \\ & x_i \in \mathbb{Z},\ i \in I \end{array} \tag{BOMILP}$$

where $z_1(x), z_2(x)$ are linear functions, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $I \subset \{1, \ldots, n\}$. We denote by $C = \{1, \ldots, n\} \setminus I$ the set of the continuous variables indices. We further denote by X the feasible set of the problem, subset of the so called *decision space*:

$$X = \{x \in \mathbb{Z}^{|I|} \times \mathbb{R}^{|C|} : Ax \leq b\}.$$

The image of X through the functions $z_1(x), z_2(x)$ is a subset of $\mathbb{R}^2$ and it is called the *feasible set in the image* or *feasible set in the criterion space*. To characterize the solutions of problem (BOMILP) we use the standard Pareto-optimality notion based on the dominance in the image space. In particular, given two feasible solutions $x', x'' \in X$, we say that x' dominates x'' and $z(x')$ dominates $z(x'')$ if $z_i(x') \leq z_i(x''),\ i = 1, 2$ and $z(x') \neq z(x'')$.

Definition 1.1 ((weakly) efficient solution and (weakly) non-dominated point)**.** *A feasible solution $x^* \in X$ is called an* efficient solution *for problem* (BOMILP) *if there is no feasible solution $x \in X$ such that*

$$z_i(x) \leq z_i(x^*) \ \text{ for } i = 1, 2 \ \text{ and } \ z_k(x) < z_k(x^*) \text{ for some } k \in \{1, 2\}.$$

The image $z(x^*)$ is called non-dominated point*. Moreover, a feasible solution $\hat{x} \in X$ is called a* weakly efficient solution *for problem* (BOMILP) *if there is no feasible solution $x \in X$ such that*

$$z_i(x) < z_i(\hat{x}) \quad \text{for } i = 1, 2.$$

The image $z(\hat{x})$ is called a weakly non-dominated point*.*

The set of all non-dominated points of a BOMILP is called the non-dominated set or also the Pareto frontier. We denote it by $\mathcal{Y}_N$. In Figure 1, we report the feasible set in the image space of an instance proposed in (Fattahi and Turkay, 2018). Since the feasible set in the image space of a BOMILP is a union of polyhedra in $\mathbb{R}^2$, its non-dominated set can contain isolated points as well as open, half-open, and closed line segments. This makes the exact detection of the non-dominated set of a BOMILP an achievable, though hard, computational task. The isolated points as well as the extremes of the line segments that form the non-dominated set of a BOMILP are called *extreme non-dominated points*. Each polyhedron in the criterion space is the image of feasible solutions having the integer variables fixed to specific feasible values. The continuous bi-objective linear subproblems obtained when the integer variables are fixed to specific integer values are called *slice problems* (Belotti et al., 2013). We will further refer to an *efficient integer assignment* as to a fixing of the integer variables in such a way that there exists a non-dominated point of (BOMILP) with exactly that fixing. Given (BOMILP), we denote with z_{id} its ideal point, namely $(z_{id})_i := \min_{x \in X} z_i(x)$, $i = 1, 2$ and we denote by $x^{id,1}, x^{id,2} \in X$, two feasible solutions such that $(z_{id})_1 = z_1(x^{id,1})$ and $(z_{id})_2 = z_2(x^{id,2})$. Given a vector $v \in \mathbb{R}^n$, we will denote by $\|v\|$ its euclidean norm.

The paper is organized as follows. In Section 1.1, we review some exact approaches for solving BOMILPs. In Section 2 we present our method: a scheme of the algorithm is reported and commented. In Section 2.1, we introduce the so called *extreme-inequalities* used within two different versions of our algorithm to cut dominated regions in the image space. An analysis of the algorithm is given in Section 3, where we show that under suitable assumptions, our method is able to detect the exact Pareto frontier of a BOMILP after a finite number of iterations. Computational results are reported in Section 4, where different strategies based on the use of the extreme-inequalities are explored and a comparison with the Triangle Splitting Algorithm (TSM) (Boland et al., 2015b) is shown. Section 5 concludes.

1.1. Literature Review

Since the first contribution by Mavrotas and Diakoulaki (Mavrotas and Diakoulaki, 1998), three main categories of algorithms have been proposed in the literature for solving multiobjective mixed integer linear problems

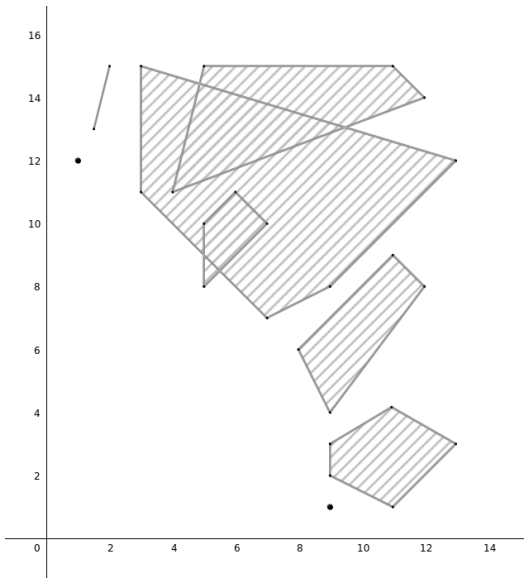


Figure 1: The feasible set in the image space of a BOMILP (Fattahi and Turkay, 2018). Its Pareto frontier contains isolated points, half-open and closed line segments.

(MOMILPs): branch-and-bound methods, criterion, or image space algorithms, and hybrid methods consisting in a combination of the previous two approaches. Branch-and-bound methods represent the first and most investigated approach to deal with MOMILPs (see e.g. (Mavrotas and Diakoulaki, 1998; Vincent et al., 2010, 2013; Bökler et al., 2024; Parragh and Tricoire, 2019; Forget and Parragh, 2024)) and in particular with BOMILPs (see (Belotti et al., 2013, 2016; Adelgren and Gupte, 2022)). However, as the image space usually has a smaller dimension than the decision space, also image space algorithms have been designed, by exploiting this advantage and focusing on the structure of the problem in the image space. The first algorithm belonging to this category is the *Triangle Splitting Method* by (Boland et al., 2015b) designed for BOMILPs. This algorithm starts by computing the lexicographically optimal images and building a rectangle from them. All local extreme supported non-dominated images are found and then used to split the rectangle in upper rectangular triangles. The hypotenuse of each triangle is then investigated by solving an auxiliary MIP. At this stage, two situations can occur: the whole hypotenuse is non-dominated or a part of it is non-dominated and an unsupported image is found. In this latter case the triangle is split into two new rectangles and the same procedure is re-

peated. Between iterations, the splitting direction is changed. At the end of the algorithm a post-processing procedure is adopted to represent the Pareto frontier via a minimal number of line segments. In (Soylu and Yıldız, 2016) the *ϵ-Tabu Constraint Algorithm* is introduced. As in (Boland et al., 2015b), it starts from the lexicographically optimal images. Then it computes the corresponding slice problem and the associated non-dominated set via dichotomic search. Thereafter the line segments of the slice problem are checked for dominance by using an auxiliary problem based on ϵ-constraints and no-good contraints. If a line segment is (partially) dominated, the procedure switches to the slice that dominates the current line segment and starts exploring the new slice. The algorithm ends when all line segments have been explored. We further cite (Boland et al., 2015a; Fattahi and Turkay, 2018; Perini et al., 2020; Pecin et al., 2024) as additional methods appeared in the literature to deal with BOMILPs. As regards hybrid methods, these are image space approaches that use methods from decision space algorithms (branch-and-bound algorithms) or vice versa. In this class we can mention (Soylu, 2018) and (Stidsen and Andersen, 2018). In the first, the authors present the Search and Remove Algorithm. At each step efficient slices are searched through a dichotomic search and, via Tabu constraints, excluded in the following steps. The algorithms continues until no more efficient slices can be found. Then, the non-dominated sets of all found slices are computed. The Pareto frontier of the original BOMILP is obtained by computing the upper/lower envelope of their Pareto frontiers. In (Stidsen and Andersen, 2018) the authors present a branch-and-bound method for BOMILP where continuous variables are allowed only in one objective function. The objective space is partitioned via lines starting from the origin and then used to parallelize the algorithm without the need of communications between processors. We also refer to (Halffmann et al., 2022) for a comprehensive overview on solution approaches for multiobjective mixed integer linear problems.

1.2. Our Contribution

We propose a method named `PADMe`, which serves as a flexible and general algorithmic framework for generating the Pareto frontier of a BOMILP. `PADMe` operates in the criterion space and offers several noteworthy features that contribute to the literature on exact approaches for bi-objective mixed integer optimization:

1. `PADMe` has a simple structure based on an ordered exploration of the

feasible region in the image space. This design makes `PADMe` relatively easy to implement compared to other methods.

2. `PADMe` is an oracle-based method that requires solving two classes of subproblems during its iterations: mixed integer programming problems and bi-objective programming problems. In this work, we focus on the linear case and present a method for transforming a solver for bi-objective linear programming problems into one capable of handling bi-objective mixed integer linear programming problems. The effectiveness and the accuracy in detecting the Pareto frontier of a BOMILP directly depend on the performance of the solver chosen to address the bi-objective linear programming problems. Therefore, the flexibility in selecting this solver allows users to prioritize either the precision of the retrieved Pareto frontier or the overall computational efficiency of the algorithm. In our implementation, we use BENSOLVE, an open-source solver for BOLPs (Löhne and Weißing, 2017). The experimental results obtained on a testbed of instances demonstrate that `PADMe` is able to produce Pareto frontiers that, at least partially, dominate the ones generated by the Triangle Splitting Method (TSM) (Boland et al., 2015b).
3. Specific cuts, referred to as `extreme-inequalities`, are introduced to avoid the exploration of dominated regions of the feasible set in the image space. We theoretically show that the use of `extreme-inequalities` does not hinder the algorithm's ability to detect the complete Pareto frontier of the BOMILP. As such, `extreme-inequalities` serve as a valuable tool that can be incorporated into the design of other criterion space algorithms for BOMILPs.
4. During the execution of the algorithm, the nondominated set is dynamically updated and filtered using the data structure proposed in (Adelgren et al., 2018). This, combined with the fact that `PADMe` generates the Pareto frontier of the BOMILP in an ordered manner, implies that - unlike other existing algorithms in the literature - no post-processing filtering procedure is required.
5. `PADMe` is an exact method. In our theoretical analysis we show that the method exactly identifies the Pareto frontier of a BOMILP.

2. PADMe: a criterion space method for the accurate detection of the Pareto frontier of BOMILPs

The method we present detects the Pareto frontier of a (BOMILP) alternating the solution of single-objective mixed integer linear and bi-objective linear subproblems. In particular, PADMe is based on the idea of detecting the efficient integer assignments of (BOMILP) and the related extreme non-dominated points, visiting the feasible set in the image space from top-left to bottom-right. At every iteration, a new integer assignment is detected by addressing a specific MILP, while potential non-dominated extreme points are detected by addressing specific BOLPs. The scheme of our method is reported in Algorithm 1. As a standard assumption, we assume that the ideal point of (BOMILP), or, equivalently, the minimum of each objective over the feasible set, exists:

Assumption 2.1. *Given* (BOMILP)*, we assume that the ideal objective values* $z_i^{id} := \min_{x \in X} z_i(x)$*,* $i = 1, 2$*, and thus the ideal point* $z^{id} := (z_1^{id}, z_2^{id})^T \in \mathbb{R}^2$*, exist.*

Note that this assumption guarantees the existence of the Pareto frontier of (BOMILP).

As a first step (see line 3 in Algorithm 1), we address the single-objective mixed integer linear problem having $z_1(x)$ as objective function and the feasible set of (BOMILP). Its solution $(x_I^0, x_C^0) = (x_I^{id,1}, x_C^{id,1})$ provides the integer assignment x_I^0 that identifies a first polyhedron in the image space or a first *slice problem*, as commonly called in the literature (see e.g. (Belotti et al., 2013)). In particular, since x_I^0 is obtained by minimizing $z_1(x)$, the first identified polyhedron/slice problem is located in the upper-left part of the feasible set in the image space. Its extreme non-dominated points, computed in step 4 of Algorithm 1 form the set Y^0 and are determined by addressing the bi-objective linear problem obtained from (BOMILP), with the integer variables fixed to x_I^0. Note that (x_I^0, x_C^0) may, in general, be only weakly efficient. Consequently, it may represent the integer assignment of a slice problem that the algorithm will need to discard in later iterations. After having addressed the slice problem identified by the integer assignment x_I^0, we enter in a loop (see step 10 in Algorithm 1), in order to explore the feasible set in the image space until the minimum with respect to $z_2(x)$, computed at step 9 in Algorithm 1, is reached. We recall that such minimum is the second component of the ideal vector, denoted by $(z_{id})_2$ and it is attained at

Algorithm 1 `PADMe` = `PA`areto `D`etection `Me`thod

1: **Input:** (BOMILP), $k = 0$, $\pi = \emptyset$, ID
2: **Output:** Pareto frontier $\mathcal{Y}_N$ of $(BOMILP)$
3: **Compute** $x^{id,1} = (x_I^0, x_C^0) \in \arg\min_{x \in X} z_1(x)$
4: **Compute** the set $Y^0 = \{y_1^0, \ldots, y_{p^0}^0\}$ of extreme non-dominated points of

$$\min_{x \in X, x_I = x_I^0} (z_1(x),\ z_2(x))^T$$

5: **for** $j = 1, \ldots, p^0 - 1$ **do**
6: $\quad \pi^* = \left(\left((y_j^0)_1, (y_j^0)_2 \right), (y_{j+1}^0)_1, (y_{j+1}^0)_2, \emptyset, \emptyset \right)$
7: $\quad$ `Insert`(π^*, π)
8: **end for**
9: **Compute** $x^{id,2} = (x_I^{id,2}, x_C^{id,2}) \in \arg\min_{x \in X} z_2(x)$
10: **while** $z_2((x_I^k, x_C^k)) > (z_{id})_2$ **and** $\mathcal{S}^k \neq \emptyset$ **do**
11: $\quad$ **Set** $k = k + 1$
12: $\quad$ **Compute** (x_I^k, x_C^k) by solving (MILP$_k$)
13: $\quad$ **Set** $Y^k = \emptyset$
14: $\quad$ **Set** F^k as in (1)
15: $\quad$ $(Y^k, \pi) =$ `SolveSP`$(x_I^k, F^k, p^{k-1}, \pi, ID)$
16: $\quad$ **if** $Y^k = \emptyset$ **then**
17: $\quad\quad$ **Set** $Y^k = Y^{k-1}$
18: $\quad$ **end if**
19: $\quad$ **Set** $p^k = |Y^k|$
20: **end while**
21: **Return** $\mathcal{Y}_N = \pi$

$x^{id,2} = (x_I^{id,2}, x_C^{id,2})$. At each iteration, we identify the new slice problem to solve and a corresponding potential efficient integer assignment as follows. We begin by solving a single-objective mixed-integer linear subproblem (see step 12 in Algorithm 1), where the first objective function $z_1(x)$ is minimized over the original feasible set X, intersected with two additional constraints that bound the values of $z_2(x)$ and $z_1(x)$, respectively, along with a set of so-called *no-good* (or Tabu) constraints (Soylu and Yıldız, 2016). The role of the no-good constraint is to exclude integer assignments that have already been explored. To be more precise, at each iteration the following problem

Algorithm 2 `Update`π (Y, p, π) (Adelgren et al., 2018)

if $p == 1$ **then**
  $\pi^* = (((y_1)_1, (y_1)_2), (y_1)_1, (y_1)_2, \emptyset, \emptyset)$
  `Insert`(π^*, π)
else
  for $j = 1, \ldots, p-1$ **do**
    $\pi^* = (((y_j)_1, (y_j)_2), (y_{j+1})_1, (y_{j+1})_2, \emptyset, \emptyset)$
    `Insert`(π^*, π)
  end for
end if

is addressed:

$$\begin{aligned} \min \quad & z_1(x) \\ \text{s.t.} \quad & x \in X, \\ & z_2(x) \leq z_2(x_I^{k-1}, x_C^{k-1}) \\ & z_1(x) \leq z_1(x^{id,2}) \\ & x_I \neq x_I^0, \ldots, x_I^{k-1}. \end{aligned} \tag{MILP$_k$}$$

As mentioned, the Tabu constraints $x_I \neq x_I^j$, $j = 0, 1, \ldots, k-1$ are included into (MILP$_k$) in order to exclude slice problems that have already been analyzed in a previous iteration. Such constraints take the form $\|x_I - x_I^j\|_1 \geq 1$, $j = 0, 1, \ldots, k-1$, where with $\|\cdot\|_1$ we denote the ℓ_1-norm. In the specific case of binary problems, they can be written as

$$\sum_{\substack{m \in I: \\ (x_I^j)_m = 1}} x_m - \sum_{\substack{m \in I: \\ (x_I^j)_m = 0}} x_m \leq \sum_{\substack{i \in I: \\ (x_I^j)_m = 1}} 1 - 1.$$

The Tabu constraints can also be linearized in the general integer case; for a detailed description, we refer the interested reader to Soylu and Yıldız (2016). Given the solution of (MILP$_{k-1}$), $\hat{x}^{k-1} = (x_I^{k-1}, x_C^{k-1})$ we restrict the search of non-dominated points of (BOMILP) in the image space below $z_2(\hat{x}^{k-1})$. Note that the constraint on z_2 in (MILP$_k$) excludes slice problems belonging to the image space above $z_2(\hat{x}^{k-1})$ and then dominated by $z(\hat{x}^{k-1})$. On the other hand, the constraint on $z_1(x)$ in (MILP$_k$) excludes slice problems dominated by $z(x^{id,2})$. Indeed, assuming that the ideal point of (BOMILP) exists, we have that the non-dominated set $\mathcal{Y}_N$ is contained in the box $[z_1^{id}, z_1(x^{id,2})] \times$

$[z_2^{id}, z_2(x^{id,1})]$. Therefore, we can exclude slice problems outside such box. In Algorithm 1, the feasible set of (MILP$_k$) is denoted by $\mathcal{S}^k$ for ease of notation. Once problem (MILP$_k$) is solved, we detect an approximation of the partial potential Pareto frontier of the slice problem related to $x_I = x_I^k$, by addressing one or a number of bi-objective linear problems (BOLPs). This is done calling the function `SolveSP` in step 15 of Algorithm 1. Clearly, there are many possibilities in addressing the task of solving a given slice problem and detect its extreme non-dominated points. In this work, we explore the possibility of using an oracle such as BENSOLVE (Löhne and Weißing, 2017). As far as we know, it is the first time that BENSOLVE is integrated within an algorithm for mixed integer problems.

To avoid exploring dominated regions, specific cuts in the image space are introduced. We propose two alternative strategies for applying these cuts and define the corresponding `SolveSP` function accordingly (see Section 2.1).

In order to efficiently store and filter the points and line segments detected at each iteration, we use the data structure developed in (Adelgren et al., 2018), called Bi-objective Tree (BoT). In particular, in our algorithm we keep updated the data structure π, where the new points and line segments are the input of the *insert* function from (Adelgren et al., 2018) (see Algorithm 2).

2.1. Using extreme-inequalities *to cut dominated regions in the image space*

At every iteration, our algorithm needs to address a slice problem. To avoid exploring dominated regions of the feasible set in the image space, we define specific cuts based on the extreme non-dominated points identified by addressing the previous slice problem. To be more precise, let (x_I^k, x_C^k) be the solution of (MILP$_k$) obtained at iteration $k \geq 1$ and let $Y^{k-1} \neq \emptyset$ be the list of extreme non-dominated points, whose cardinality is denoted by p^{k-1}, detected at iteration $k-1$ related to the $(k-1)$-th integer assignment x_I^{k-1} (or the $(k-1)$-th slice problem). The list of extreme non-dominated points is sorted in increasing order with respect to the first coordinate (or first objective). First, we define the set F^k as the one that identifies the region of the current slice problem, with $z_2(x) \leq (y_1^{k-1})_2$:

$$F^k = \{x \in X,\ z_2(x) \leq (y_1^{k-1})_2,\ x_I = x_I^k\}, \tag{1}$$

being $y_1^{k-1} \in Y^{k-1}$ the first extreme non-dominated point detected at the previous iteration. In order to check whether the current x_I^k is defining a potential efficient integer assignment and in order to exclude dominated region

of the image space, we use information from the previously addressed slice problem as follows. Given the pair of subsequent extreme non-dominated points $(y_i^{k-1}, y_{i+1}^{k-1})$, with $1 \leq i < p^{k-1}$, we define the i-th *extreme-inequality* as

$$(w_i)^T z(x) \leq (w_i)^T y_i^{k-1},$$

where w_i is

$$w_i = \frac{(v_1^i, v_2^i)}{\|v^i\|} \tag{2}$$

being

$$v_1^i = (y_i^{k-1})_2 - (y_{i+1}^{k-1})_2, \quad v_2^i = (y_{i+1}^{k-1})_1 - (y_i^{k-1})_1.$$

In case $i = p^{k-1}$, we set $w_i = (0,1)^T$. Note that $v_1^i \geq 0$ and $v_2^i \geq 0$ for all $i \in \{1, \ldots, p^{k-1}\}$. Loosely speaking, an *extreme inequality* identifies the half plane in the criterion space defined through the line connecting two subsequent extreme non-dominated points. See also the example showed in Figure 3 (b).

Within `PADMe`, once the set F^k is computed (step 14 in Algorithm 1), the function `SolveSP` is called (step 15 in Algorithm 1), giving as output the list of extreme non-dominated points of the k-th slice problem and an updated bi-objective tree. Clearly, there are several ways to define the `SolveSP` function and we propose two alternatives, depending on how the extreme-inequalities are used. The function `SolveSP` includes the parameter ID as input: If $ID = 0$, the extreme-inequalities are used to define a sequence of BOLPs (see section 2.1.1) where the feasible set of the slice problem is conveniently reduced. In case $ID = 1$, the extreme-inequalities are used to check whether the slice problem can be discarded or not, under a criterion guided by the ideal point of the slice problem. From our computational experience and our implementation of `PADMe`, the option $ID = 1$ is much faster.

2.1.1. Setting $ID = 0$: solve the slice problem by addressing a sequence of BOLPs with reduced feasible regions

A first and direct way of using extreme-inequalities is to avoid the exploration of dominated regions when addressing the slice problem. Within a loop on the index $i \in \{1, \ldots p^{k-1}\}$, we check whether the set F^k intersected with one inequality per time, is non-empty. In particular, we check if the following set:

$$W_i^k = F^k \cap \{x \in \mathbb{R}^n : (w_i)^T z(x) \leq (w_i)^T y_i^{k-1}\}, \tag{3}$$

with $i \in \{1, \dots, p^{k-1}\}$ is non-empty. In case $W_i^k \neq \emptyset$, we address the BOLP

$$\min_{x \in W_i^k} (z_1(x), z_2(x))$$

and we enrich the set Y^k with its extreme non-dominated points. Once all the sets W_i^k, $i \in \{1, \dots, p^{k-1}\}$ have been analyzed and the corresponding non-empty BOLPs have been addressed, the set Y^k is a collection of points that we denote as *potential extreme non-dominated points*, related to the integer fixing x_I^k. Such points define the so called *partial potential Pareto frontier* of the slice problem obtained when the integer variables are fixed to x_I^k. The name *partial potential Pareto frontier* wants to emphasize the fact that the non-dominated set of a slice problem may contribute only partially to the Pareto frontier of the entire BOMILP.

In case $W_i^k = \emptyset$ for every $i = 1, \dots, p^{k-1}$, we have that the extreme non-dominated points of the current slice problem are dominated by the previously identified non-dominated points and line segments (see Proposition 3.1), so that the current integer assignment is not an efficient one and we can avoid the resolution of BOLPs. Note that the non-dominated points and line segments detected for the integer fixing x_I^k, can still be (even only partially) removed on a later iteration, in case extreme points related to a new slice problem are dominating them.

We report in Algorithm 3, a scheme of `SolveSP` when $ID = 0$.

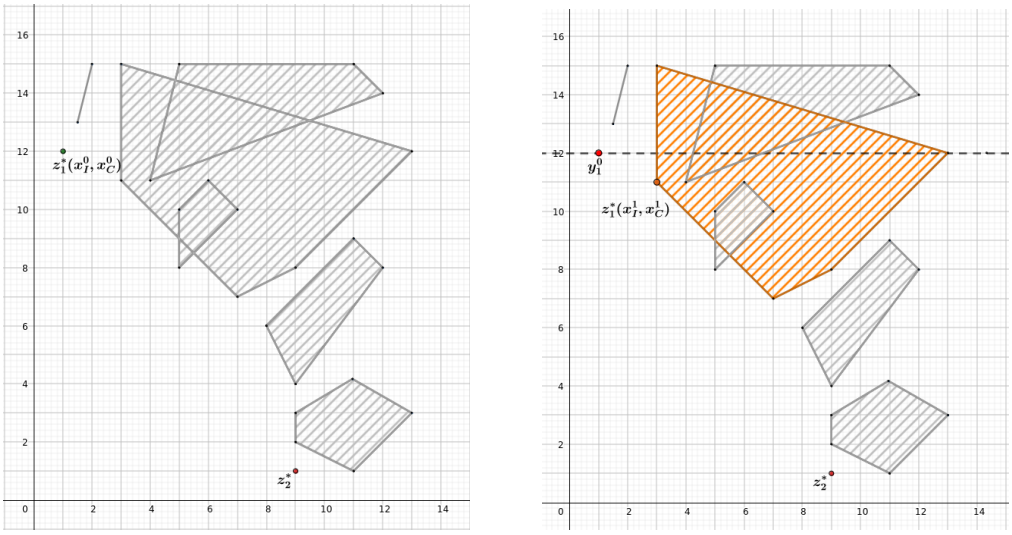
In order to better explain how Algorithm 1 works when $ID = 0$, we describe its iterations when applied to the instance proposed in (Fattahi and Turkay, 2018), whose feasible set in the image space is reported in Figure 1 (see the Appendix for the associated mathematical formulation). In Figure 2, we depict the initialization and the first iteration of `PADMe`. We start by minimizing $z_1(x)$ over the feasible set, detecting the solution (x_I^0, x_C^0) that is $(1, 12)$ in the image space. The point $z_2^* = z(x_I^{id,2}, x_C^{id,2}) = (9, 1)$ is also detected (see Figure 2 (a)). The corresponding slice problem, once the integer variables are fixed to x_I^0, is the singleton $Y^0 = \{y_1^0\} = (1, 12)$ that is inserted in the BoT data structure π (see Figure 13 (a)). Note that, in this case, $p^0 = 1$. Then, we solve problem (MILP$_k$), $k = 1$, imposing $z_2(x) \leq (y_1^0)_2$, $z_1(x) \leq z_1(x^{id,2})$ and the Tabu constraint $x_I \neq x_I^0$. The solution (x_I^1, x_C^1) is obtained that is $(3, 11)$ in the image space (see Figure 2 (b)) and the extreme non-dominated points $\{y_1^1, y_2^1\}$, namely $(3, 11)$ and $(7, 7)$ are detected, by addressing the BOLP $\min_{x \in F^1}(z_1(x), z_2(x))^T$ (see Figure 3 (a)). Therefore, $Y^1 = \{y_1^1, y_2^1\}$, $p^1 = 2$ and π is updated accordingly (see Figure 13 (b)). Note

Algorithm 3 `SolveSP`$(x_I^k, F^k, p^{k-1}, \pi,\ ID = 0)$

for $i = 1, \dots, p^{k-1}$ **do**
  if $p^{k-1} == 1$ **then**
    Compute the set $Y^{i,k} = \{y_1^{i,k}, \dots, y_{p^{i,k}}^{i,k}\}$ of extreme non-dominated points of $\min_{x \in F^k}(z_1(x),\ z_2(x))^T$
  else
    Compute w_i as in (2)
    Set W_i^k as in (3)
    if $W_i^k \neq \emptyset$ **then**
      Compute the set $Y^{i,k} = \{y_1^{i,k}, \dots, y_{p^{i,k}}^{i,k}\}$ of extreme non-dominated points of $\min_{x \in W_i^k}(z_1(x),\ z_2(x))^T$
    end if
  end if
  $Y^k = Y^k \cup Y^{i,k}$
  $\pi =$ `Update`$\pi(Y^{i,k},\ p^{i,k},\ \pi)$
end for

that, for ease of notation, we dropped the second index in the apices of the potential extreme non-dominated points y_1^1 and y_2^1.

In Figure 3 and Figure 4, we plot the conclusion of the first iteration and the second iteration of `PADMe`. Problem (MILP_2) will have $z_2(x) \leq (y_1^1)_2$, $z_1(x) \leq z_1(x^{id,2})$, $x_I \neq x_I^0$ and $x_I \neq x_I^1$ as constraints. Solving (MILP_2) leads to the solution (x_I^2, x_C^2) that is $(4, 11)$ in the image space(see Figure 3 (b)). Since $p^1 = 2$, the set W_1^2 is defined as the intersection of F^2 with the extreme-inequality $(w_1)^T z(x) \leq (w_1)^T y_1^1$, where w_1, according to (2), is equal to $(1/\sqrt{2}, 1/\sqrt{2})^T$. The extreme inequality is then $z_1(x)/\sqrt{2} + z_2(x)/\sqrt{2} \leq 14/\sqrt{2}$. The intersection of F^2 with such extreme-inequality is empty as well as the intersection between F^2 and the second extreme-inequality $z_2(x) \leq (y_2^1)_2$, that is $z_2(x) \leq 7$. Therefore, we discard the related slice problem, as it cannot contribute to the definition of the Pareto frontier of the original problem. In particular, we set $Y^2 = Y^1 = \{y_1^1, y_2^1\}$. We go on solving problem (MILP_3) with $z_2(x) \leq (y_1^1)_2$, $z_1(x) \leq z_1(x^{id,2})$, $x_I \neq x_I^0$, $x_I \neq x_I^1$ and $x_I \neq x_I^2$ as constraints, detecting the solution (x_I^3, x_C^3) that is $(5, 8)$ in the image space (see Figure 4 (a)). Now, W_1^3 is defined as the intersection of F^3 with the extreme-inequality $(w_1)^T z(x) \leq (w_1)^T y_1^1$. Such intersection is non-empty and the BOLP $\min_{x \in W_1^3}(z_1(x), z_2(x))^T$ has $y_1^3 = (5, 8)$ as single



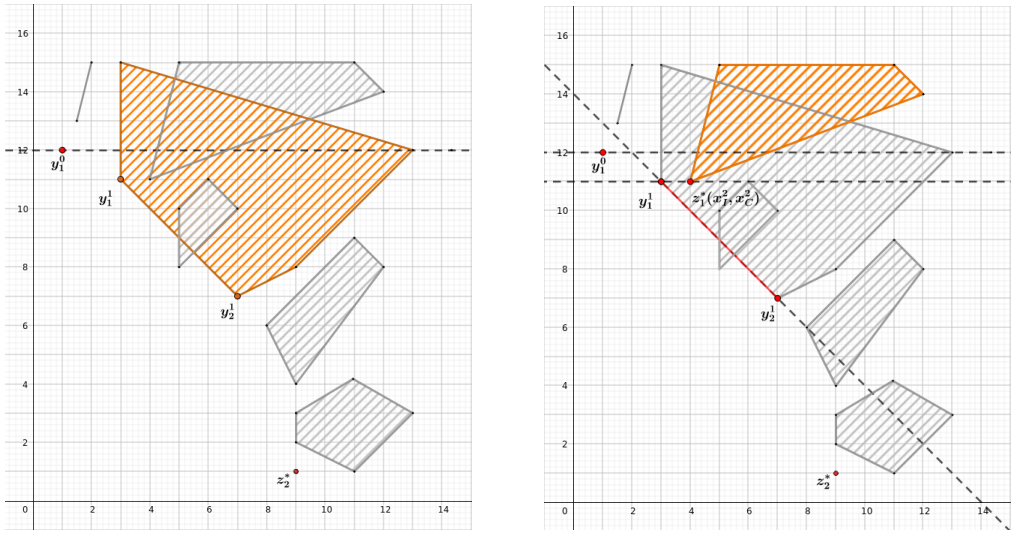
(a) (b)

Figure 2: Illustration of `PADMe` on a BOMILP instance: initialization and first iteration.

extreme non-dominated point. Adding this point in the data structure π has the effect of splitting the previously identified line segment $[y_1^1, y_2^1]$ into two smaller segments. The segments $[y_1^{1,1}, y_2^{1,1}] = [(3, 11), (5, 9)]$ and $[y_1^{3,1}, y_2^{3,1}] = [(6, 8), (7, 7)]$, together with the point y_1^3 are kept in the BoT data structure π (see Figure 4 (b) and Figure 13 (c)).

In Figure 5, the fourth iteration is shown. The new polyhedron to explore is highlighted in Figure 5 (a), and, since $p^3 = 1$, the BOLP considered at the fourth iteration is $\min_{x \in F^4}(z_1(x), z_2(x))^T$, where F^4 includes the constraints $z_2(x) \leq (y_1^3)_2$ and $x_I = x_I^4$. The extreme line segment $[y_1^4, y_2^4] = [(8, 6), (9, 4)]$ is detected and memorized within π (see Figure 5 (b) and Figure 13 (d)). At the fifth iteration the slice problem with $x_I = x_I^5$ is considered and the related steps of the algorithm are depicted in Figure 6.

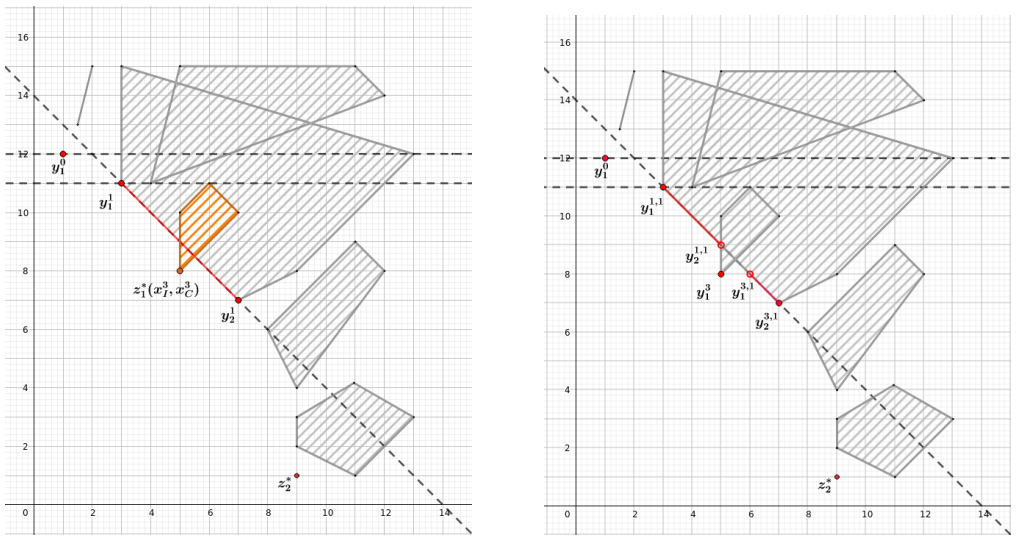
In the fifth iteration, two BOLPs will be addressed. The first BOLP addressed will have the intersection of F^5 with the extreme-inequality $(w_1)^T z(x) \leq (w_1)^T y_1^4$, that is $2z_1(x)/\sqrt{5} + z_2(x)/\sqrt{5} \leq 22/\sqrt{5}$ as feasible set. The components of $w_1 \in \mathbb{R}^2$ are computed according to (2), considering $y_1^4 = (8, 6)$ and $y_2^4 = (9, 4)$. The extreme non-dominated points detected will be $y_1^{1,5} = y_1^5 = (9, 2)$ and $y_2^{1,5}$, so that $Y^{1,5} = \{y_1^{1,5}, y_2^{1,5}\}$. Then, the second BOLP having F^5 intersected with $z_2(x) \leq y_2^4$ as feasible set will be considered. The points $y_1^{2,5} = y_1^5 = (9, 2)$ and $y_2^{2,5} = (11, 1)$ will be detected and will



(a) (b)

Figure 3: Illustration of `PADMe` with $ID = 0$ on a BOMILP instance: detection of $Y^1 = \{y_1^1, y_2^1\}$ (a) and second iteration (b). The slice problem obtained fixing the integer variables to x_I^2 is discarded, as the feasible set obtained intersecting F^2 with the extreme-inequality $(w_1)^T z(x) \leq (w_1)^T y_1^1$ is empty.

form the set $Y^{2,5}$. The partial potential Pareto frontier of the slice problem associated with the fixing x_I^5 is then made of the segment having as extreme non-dominated points y_1^5 and $y_2^{2,5}$, union of the sets $Y^{1,5}$ and $Y^{2,5}$ (see Figure 6 (b)). At the sixth iteration, given the constraint $z_2(x) \leq (y_1^5)_2$, problem (MILP$_6$) detects (x_I^6, x_C^6) and $y_1^6 = z(x_I^6, x_C^6) = (9, 1)$ is the extreme non-dominated point of the sixth slice problem considered (see Figure 7 (b)). The point y_1^6 is included in the data structure π (see Figure 13 (f)), so that the previously detected line segment $[y_1^5, y_2^{2,5}] = [(9, 2), (11, 1)]$ is removed as it is dominated by y_1^6. The entire Pareto frontier of the BOMILP is exactly detected and stored in the data structure π . In Figure 8, we report the BoT (bi-objective tree) π associated with the Pareto frontier of the instance. Each node, represents either an isolated non-dominated point or a pair of extreme non-dominated points, defining a non-dominated line segment. The label $\pi.r.r.r.r$ denotes the relation of the last generated node with respect to the root node. An ordered visit (from left to right and from bottom to top) of the BoT returns the complete Pareto frontier.



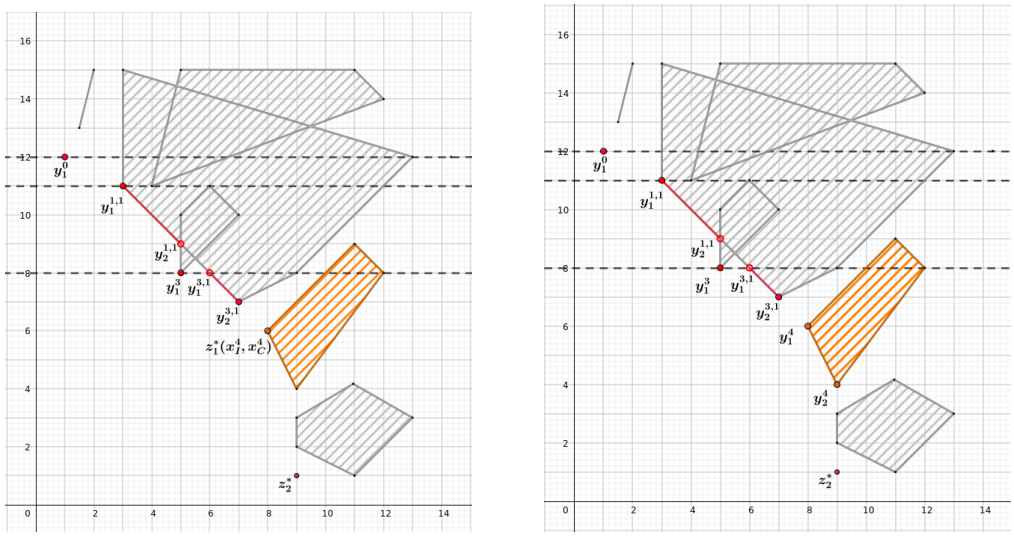
(a) (b)

Figure 4: Illustration of `PADMe` with $ID = 0$ on a BOMILP instance: the third iteration. The extreme-inequality $(w_1)^T z(x) \leq (w_1)^T y_1^1$ allows to cut dominated regions of the slice problem obtained by fixing the integer variables to x_I^3. Therefore, the BOLP solver addresses a problem with a reduced feasible set.

2.1.2. Setting $ID = 1$: check if z_{id}^k belongs to any of the halfspaces defined by the extreme-inequalities

In a second version of `SolveSP`, the extreme-inequalities are not used as further constraints to be added in the BOLP subproblems, but in order to check whether a slice problem can contribute to the non-dominated set or can be discarded. At every iteration k of Algorithm 1, if $ID = 1$, we check within `SolveSP` whether the ideal point of the new slice problem belongs to any of the halfspaces defined by the extreme-inequalities. If this is not the case, we can discard the integer assignment obtained as it cannot lead to extreme non-dominated points, as shown in Proposition 2.2 and visualized in Figure 9. Otherwise, in case the ideal point belongs to at least one halfspace defined by the extreme-inequalities, we consider the BOLP $\min_{x \in F^k}(z_1(x), z_2(x))^T$ without any additional constraint. An illustration of the strategy implemented is depicted in Figure 10 and detailed in the Algorithm 4.

Proposition 2.2. *Given* (BOMILP)*, let z_{id}^k be the ideal point of the slice problem obtained at iteration $k \in \mathbb{N}$. Then, if $(w_i)^T z_{id}^k > (w_i)^T y_i^{k-1}$ for all $i \in \{1, \ldots, p^{k-1}\}$, no extreme non-dominated point can be detected by*



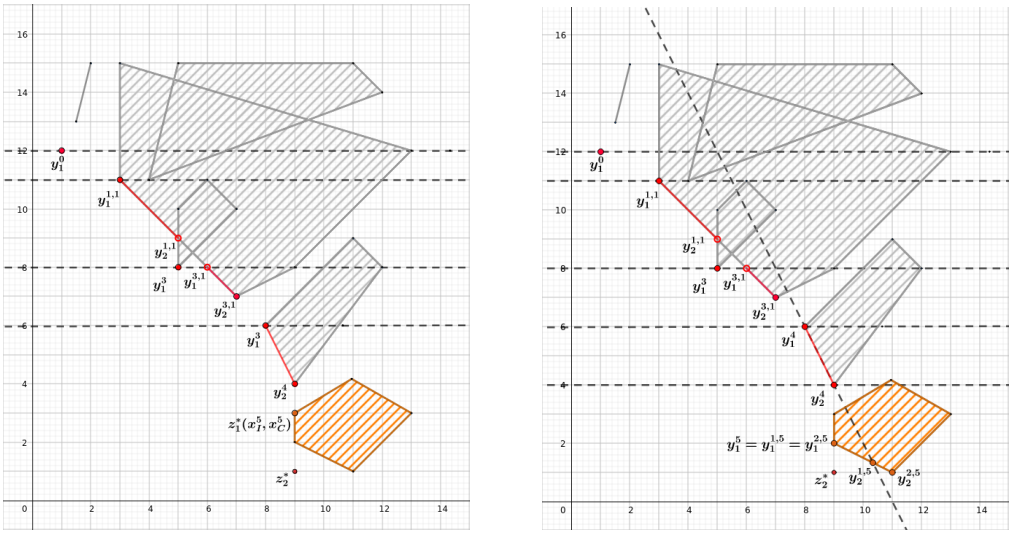
(a) (b)

Figure 5: Illustration of PADMe with $ID = 0$ on a BOMILP instance: the fourth iteration. By solving problem (MILP$_4$) the point (x_I^4, x_C^4) is detected (a). Once the related slice problem is addressed the set $Y^4 = \{y_1^4, y_2^4\}$ is detected (b).

addressing the k-th slice problem (so that the slice problem can be discarded).

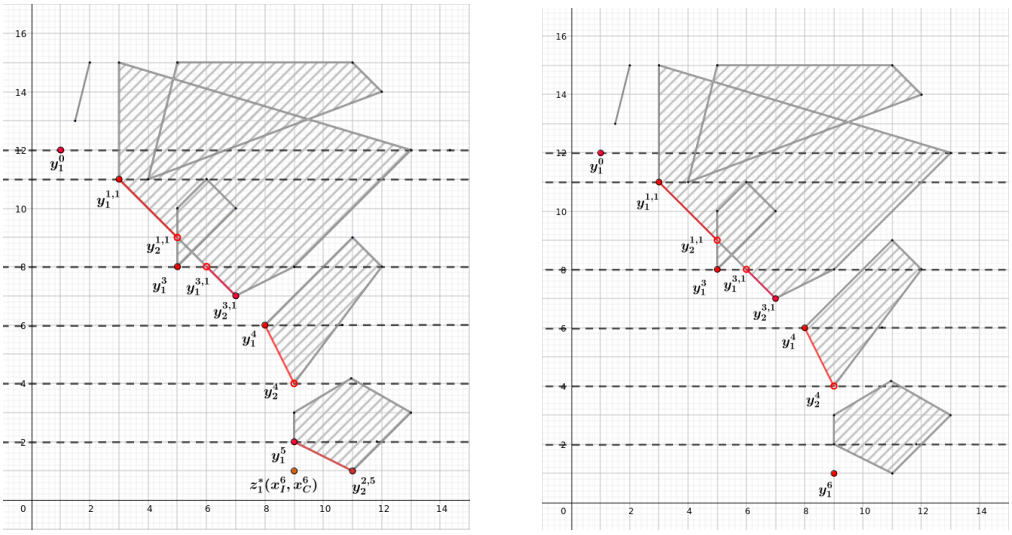
Proof. By definition of the ideal point z_{id}^k, we have that the image of each feasible solution $z(x)$ of the k-th slice problem, namely the image through $z_1(x)$ and $z_2(x)$ of every solution in the set $\{x \in X \mid x_I = x_I^k\}$, is dominated by the ideal point: $z(x) \geq z_{id}^k$, $z(x) \neq z_{id}^k$. Having $(w_i)^T z_{id}^k > (w_i)^T y_i^{k-1}$ for all $i \in \{1, \ldots, p^{k-1}\}$ implies that z_{id}^k is dominated by some non-dominated point of the $(k-1)$-th slice problem. Therefore, since $z(x) \geq z_{id}^k$, we have that any point of the k-th slice problem is dominated by some point from the $(k-1)$-th slice problem, so that no extreme non-dominated point can be detected by addressing the k-th slice problem. □

In order to better explain how Algorithm 1 works when $ID = 1$, we refer again to the illustrative example in Figure 1, based on the instance proposed in (Fattahi and Turkay, 2018). Note that the iterations of PADMe will identify exactly the same slice problems as when $ID = 0$. However, we can spot differences in two iterations. In the third iteration, the *extreme-inequality* defined from y_1^1 and y_2^1 is used to verify that the slice problem (obtained by fixing the integer variables to x_I^3) must be solved. Note that no reduction



(a) (b)

Figure 6: Illustration of PADMe with $ID = 0$ on a BOMILP instance: the fifth iteration. By solving problem (MILP$_5$) the point (x_I^5, x_C^5) is detected (a). Two BOLPs are addressed in order to populate $Y^5 = Y^{5,1} \cup Y^{5,2} = \{y_1^5, y_2^5\}$ (b).



(a) (b)

Figure 7: Illustration of PADMe with $ID = 0$ on a BOMILP instance: the sixth iteration. The complete non-dominated set detected by PADMe is reported in subfigure (b).

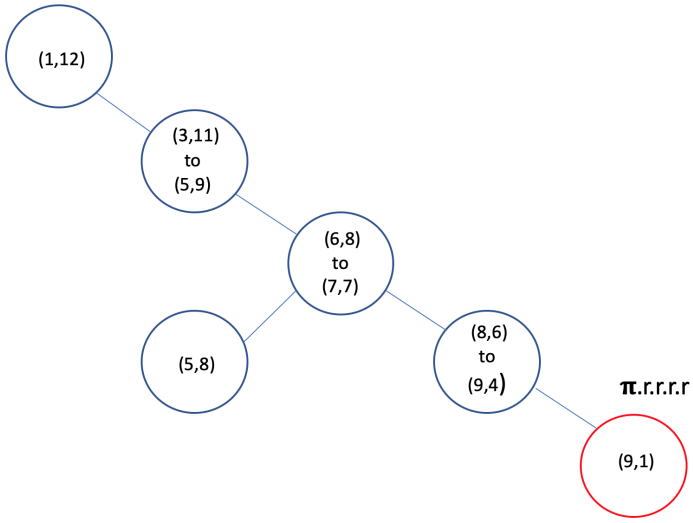


Figure 8: The BoT (bi-objective tree) (Adelgren et al., 2018) associated with the Pareto frontier of the BOMILP instance solved by `PADMe`.

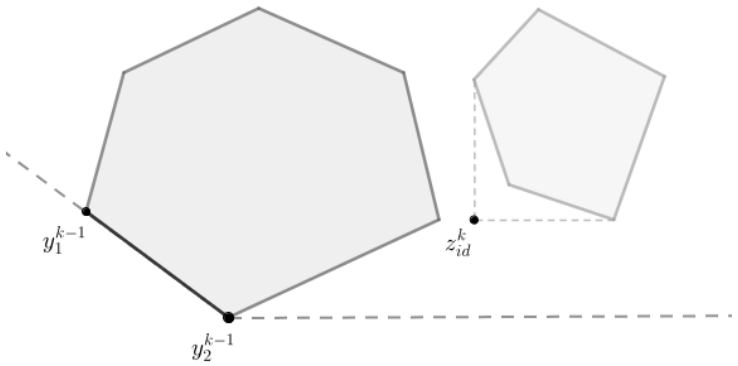


Figure 9: Illustration of Proposition 2.2. If the ideal point z_{id}^k is cut by every extreme-inequality, the current slice problem can be discarded. In the figure such case is depicted: $(w_i)^T z_{id}^k > (w_i)^T y_i^{k-1}$ for all $i \in \{1, \ldots, p^{k-1}\}$.

in its feasible region is induced. Similarly, in the fifth iteration, the two *extreme-inequalities* confirm that the new identified slice problem (obtained by fixing the integer variables to x_I^5) has to be addressed. Differently from $ID = 0$, only one BOLP is solved to detect the extreme non-dominated

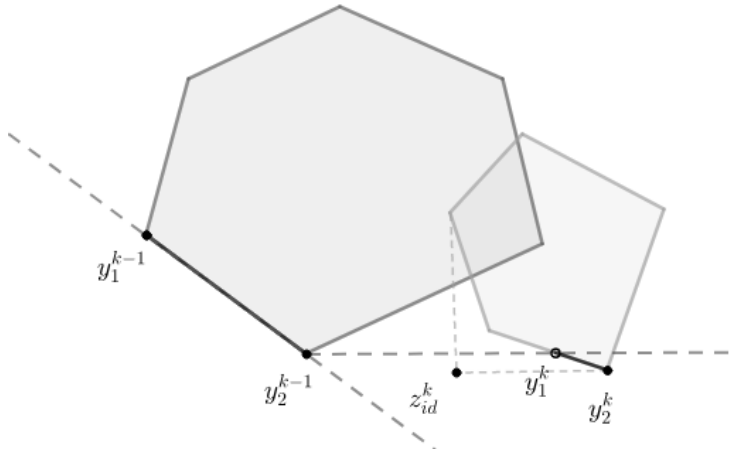


Figure 10: Use of the extreme-inequalities in combination with the ideal point: in this case, z_{id}^k is not cut by every extreme-inequality. The slice problem is then addressed and the potential extreme non-dominated points y_1^k and y_2^k are added to the BoT data structure.

Algorithm 4 `SolveSP`$(x_I^k, F^k, p^{k-1}, \pi,\ ID = 1)$

Compute z_{id}^k
for $i = 1, \ldots, p^{k-1}$ **do**
    Compute w_i as in (2)
    if $(w_i)^T z_{id}^k \leq (w_i)^T y_i^{k-1}$ **then**
        Compute the set $Y^k = \{y_1^k, \ldots, y_{p^k}^k\}$ of extreme non-dominated points of $\min_{x \in F^k}(z_1(x),\ z_2(x))^T$
        $\pi =$ `Update`$\pi(Y^k,\ p^k,\ \pi)$
        break
    else
        $Y^k = \emptyset$
    end if
end for

solutions y_1^5 and $y_2^{2,5}$ of the partial potential Pareto frontier. The sequence of BoT updates generated along the iterations is reported in Figure 13.

3. Analysis of the algorithm

In this Section, we show that Algorithm 1 detects the non-dominated set of (BOMILP) in a finite number of iterations, under specific assumptions. As a first result, we show that every non-dominated point of (BOMILP) belongs to the partial potential Pareto frontier of a slice problem. In particular, we show that the extreme-inequalities used along the iterations of Algorithm 1 are not cutting any non-dominated point and that any non-dominated point is detected at a certain iteration.

Proposition 3.1. *Given* (BOMILP)*, let Assumption 2.1 holds. Let* $y \in \mathcal{Y}_N$*. Then,* $k \in \mathbb{N}$ *exists such that* y *belongs to the partial potential Pareto frontier obtained from* Y^k *in Algorithm 1.*

Proof. Let $\{x_I^k\} \subseteq \mathbb{Z}^{|I|}$ be the sequence of integer feasible assignments for (BOMILP). Since $y \in \mathcal{Y}_N$, it necessarily belongs to the feasible set in the image space of (BOMILP) and, in particular, $k \in \mathbb{N}$ exists such that $y \in z(X^k)$, being $X^k = \{x \in X \mid x_I = x_I^k\}$. This in particular means that y is the image of a feasible solution for a specific slice problem. We show that y cannot be cut by any of the inequalities introduced in Algorithm 1. Let $x(y) \in X^k$ be the counterimage of y, i.e. $z(x(y)) = y$. Necessarily, it must be $x(y) \in F^k$, as otherwise $y_2 > (y_1^{k-1})_2$ holds and y would be dominated by y_1^{k-1}, that is a contradiction to $y \in \mathcal{Y}_N$. Assume that $x(y) \notin W_i^k$ for all $i \in \{1, \dots, p^{k-1}\}$ or, in other words, y is cut by every i-th extreme-inequality. Then, $w_i^T y > w_i^T y_i^{k-1}$ for all $i \in \{1, \dots, p^{k-1}\}$. By construction, $w_i \geq 0$ and $w_i \neq 0$ so that we get again a contradiction with $y \in \mathcal{Y}_N$, as y would be dominated by some point of the potential partial Pareto frontier of the previously analyzed slice problem. Therefore, let $j \in \{1, \dots, p^{k-1}\}$ be an index such that $x(y) \in W_j^k$. Then, since $y \in \mathcal{Y}_N$, we have that it is also a non-dominated point of the BOLP $\min_{x \in W_j^k}(z_1(x), z_2(x))^T$. The potential extreme non-dominated points of such BOLP are collected in $Y^{j,k}$ and then in Y^k. □

In case (BOMILP) is a bi-objective binary problem, or its integer variables are bounded, we can prove that Algorithm 1 exactly detects its non-dominated frontier.

Theorem 3.2. *Given* (BOMILP)*, assume that the integer variables are constrained to be binary, namely* $x_i \in \{0, 1\}$*, for* $i \in I$*, with* $|I| \leq n$ *or bounded,*

namely $x_i \in [l_i, u_i]$, *for* $i \in I$, *with* $|I| \leq n$ *and* $l_i, u_i \in \mathbb{Z}$. *Let Assumption 2.1 holds. Then, Algorithm 1 detects the Pareto frontier of* (BOMILP) *in a finite number of iterations.*

Proof. From Proposition 3.1, we have that every non-dominated point $y \in \mathcal{Y}_N$ is detected at a certain iteration $k \in \mathbb{N}$, by addressing a specific slice problem. Under the assumption that the integer variables are constrained to be binary or that the integer variables are bounded, a finite number of feasible integer assignments exists, so that a finite number of slice problems exists. □

Note that there exist cases in which Algorithm 1 cannot stop in a finite number of iterations. It can occur that an infinite number of integer feasible assignments exists although the number of slice problems belonging to $[z_1^{id}, z_1(x^{id,2})] \times [z_2^{id}, z_2(x^{id,1})]$ is finite, as Example 3.3 shows. This means that to ensure that Algorithm 1 stops in a finite number of iterations, we need to assume that a finite number of integer feasible assignments exists. Such assumption would be needed by any algorithm making use of Tabu constraints to exclude already analyzed slice problems (such as the one proposed in (Soylu and Yıldız, 2016)).

Example 3.3. *Consider the following* (BOMILP)*:*

$$\begin{array}{ll} \min & (x_1, x_2)^T \\ \text{s.t.} & x_1 + x_2 \geq 1, \\ & x_1 - x_3 \leq 0, \\ & x_3 \geq 1, \\ & x_3 \in \mathbb{Z}, \\ & x \geq 0. \end{array}$$

Its non-dominated set is made of a single closed line segment with extreme non-dominated points $(0,1)^T$ *and* $(1,0)^T$. *These extreme non-dominated points are detected by Algorithm 1 at the very beginning, before entering the while loop, when addressing the bi-objective continuous problem* $\min_{x \in X, x_I = x_I^0} (z_1(x), z_2(x))^T$. *Indeed,* $Y^0 = \{(0,1)^T, (1,0)^T\}$. *However, there exist an infinite number of integer feasible assignments associated to slice problems sharing the same partial potential Pareto frontier , that is exactly the segment with extreme points* $(0,1)^T$ *and* $(1,0)^T$. *In fact, all* $x_3 \in \mathbb{Z}$, $x_3 \geq 1$ *are efficient integer assignments defining the same slice problem.*

4. Numerical results

In the following, we analyze the performance of Algorithm 1 (`PADMe`) on the class of biobjective 0-1 mixed integer programs introduced by Mavrotas and Diakoulaki (Mavrotas and Diakoulaki, 1998). This class has been used in the majority computational studies of algorithms for solving BOMILPs and are available at (Charkhgard, Accessed: 2023) along with the Triangle Splitting Method (`TSM`) implementation. The problem size of the instances varies between 20 and 320 variables and the number of constraints (m) equals the number of decision variables (n). For all instances, the number of integer variables is half of the total number of variables, i.e. $|I| = 0.5\,n$. Our method has been implemented in C++ and Matlab and all the tests have been run on an iMac intel core i5, 6 core processor running at 3GHz with 32GB RAM. In our implementation of `PADMe`, we use GUROBI (Gurobi Optimization, 2023) as solver for the MILP subproblems and BENSOLVE (Löhne and Weißing, 2017) as solver for the BOLP subproblems. We use default parameters for both solvers. In case $ID = 0$, we let BENSOLVE detect whether W_i^k is empty, i.e. the BOLP is infeasible. Furthermore, as already mentioned, we use the BoT data structure proposed in (Adelgren et al., 2018) to efficiently store the extreme non-dominated points found along the iterations of `PADMe`. The implementation of such data structure is publicly available at `https://github.com/Nadelgren/IJOC-Efficient`. A first test we propose, is related to the comparison of the two versions of `PADMe`, as reported in Algorithm 1, with a further version in which only *meaningful* extreme-inequalities are considered. The number of BOLPs to be solved at iteration k of Algorithm 1 with $ID = 0$ depends on the cardinality of Y^{k-1}, that in turn depends on the dimension of the instance. This implies that such number soon becomes prohibitive as the dimension of the instances grows (see Table 1). Therefore, we implemented a version of Algorithm 1 with $ID = 0$, called `PADMe` (ID = 0 - light), where we allow the computation of W^{i+1} and the solution of the corresponding BOLP only if the new extreme-inequality is "sufficiently different" with respect to the previous one, namely we check whether $\|w_{i+1} - w_i\| > \eta$, being η a positive parameter. In this way, we allow our algorithm to solve a smaller number of (more meaningful) BOLPs, but we may incur in a less accurate Pareto frontier detected.

In Table 1, we report a comparison among the Algorithm 1 with $ID = 0$ (referred as to `PADMe` (ID = 0)), the version where only some cuts and then only a subset of BOLPs are considered, with $\eta = 0.1$ (referred as to `PADMe`

(ID = 0 - light)) and the version with $ID = 1$, where the strategy related to the ideal point is implemented (referred as to `PADMe` (ID = 1)).

For each instance and each algorithm, we report the number of MILPs addressed (# MILPs), the number of BOLPs addressed (# BOLPs) and the computational CPU time needed in seconds (time) obtained using the `clock` C++ function. From the results in Table 1, it is clear that considering all extreme-inequalities and then all possible BOLP subproblems at each iteration, becomes prohibitive as soon as the dimension of the problem raises, as the cardinality of Y^k depends on the number of vertices of the polyhedron of the slice problem analyzed. Allowing a smaller number of extreme-inequalities clearly has the benefit of reducing the number of BOLPs to be solved and then the overall computational time, as the performance of `PADMe` (ID = 0 - light) shows. Considering a higher value for η would even improve the performance in terms of numerical burden. However, reducing the number of extreme-inequalities may come at a cost in terms of accuracy of the Pareto frontier detected, as some extreme non-dominated points could be left undetected. Since our focus is in defining a method that could be as accurate as possible, depending on the precision allowed to the BOLP solver used, we do not investigate this strategy any further.

From the results in Table 1, the advantage in using the (ID check) strategy is evident. We can notice that `PADME` (ID = 1) is one order of magnitude faster and we can conclude that helping BENSOLVE in addressing BOLPs with reduced feasible set is not paying off as we expected. For all the instances with $n = 160$, the versions of Algorithm 1 (all BOLPs) and (only some) are not able to detect the non-dominated set within one hour of CPU time (used as time limit). For instances with $n = 360$, the time limit is reached by each version.

In the following, we compare `PADMe` in the (ID = 1) version with the Triangle Splitting Method (`TSM`) (Boland et al., 2015b), that is a criterion space method designed for bi-objective problems whose implementation is available at `https://usf.app.box.com/s/6i7rcdd7njkqsvnpi7x97ku9og3j8782`.

In Table 2, we report the results obtained by our method and `TSM` on the bi-objective mixed binary instances proposed in (Mavrotas and Diakoulaki, 1998) and available at (Charkhgard, Accessed: 2023). For `PADMe` we report the number of MILP subproblems (# MILPs), the number of bi-objective linear programming subproblems (# BOLPs) addressed, the total CPU time (time) in seconds and the number of extreme non-dominated points detected (# endp). For what concerns the `TSM` we report the number of MILP sub-

Inst	PADMe (ID = 0)			PADMe (ID = 0 - light)			PADMe (ID = 1)		
	# MILPs	# BOLPs	time (s)	# MILPs	# BOLPs	time (s)	# MILPs	# BOLPs	time (s)
C20	11	55	0.07	11	55	0.07	11	5	0.01
	18	112	0.14	18	109	0.13	18	16	0.03
	22	42	0.06	22	42	0.06	22	20	0.03
	25	212	0.26	25	212	0.26	25	22	0.04
	7	41	0.05	7	41	0.05	7	5	0.01
Avg.	16.6	92.4	0.12	16.6	91.8	0.11	16.6	13.6	0.03
Max.	25	212	0.26	25	212	0.26	25	22	0.04
C40	76	1352	4.20	76	1345	4.06	76	74	0.25
	30	298	0.79	30	298	0.79	30	28	0.88
	44	381	0.97	44	381	0.97	44	42	0.12
	84	1116	3.40	84	1077	3.2	84	82	0.27
	106	737	1.95	106	730	1.93	106	89	0.29
Avg.	68	776.8	2.26	68	766.2	2.19	68	63	0.36
Max.	106	1352	4.20	106	1345	4.06	106	89	0.88
C80	586	17745	452.03	586	16889	421.02	586	564	7.98
	–	–	–	–	–	–	2241	2130	65.71
	655	15367	426.13	655	14879	336.45	655	609	10.03
	594	13967	356.23	594	13716	330.04	594	585	9.71
	492	11864	265.62	492	11223	237.91	492	469	7.07
Avg.	581.75	14735.75	375	581.75	14176.75	331.35	913.6	871.4	20.1
Max.	655	17745	452.03	655	16889	421.02	2241	2130	65.71

Table 1: Comparison among different versions of PADMe on the bi-objective instances from (Mavrotas and Diakoulaki, 1998). Each version differs in the way the extreme-inequalities are used within SolveSP.

problems (# MILPs), the number of linear programming subproblems (# LPs) addressed, the total CPU time (time) in seconds and the number of extreme non-dominated points detected. For each instance, `TSM` reports both the number of points detected before a post-processing phase (# endp-b) and the number of points detected after a post-processing phase (# endp-a), used to convert the Pareto frontier produced into a minimal representation and filter dominated points.

In Table 2, we further report, for each instance, the number of MILPs solved by the ϵ-Tabu Constraint Algorithm available from Table 4 in (Soylu and Yıldız, 2016).

For what concerns the efficiency, we can notice that `PADMe` compares favorably in terms of number of MILPs solved and in terms of CPU time up to instances with 80 variables and 80 constraints. For instances with $n \geq 160$, the computational burden asked to BENSOLVE to address a growing number of BOLPs with higher dimension, does not allow `PADMe` to be competitive in terms of CPU time. This depends on the fact that once a not-to-be-discarded slice problem is detected, we ask to solve the related BOLP with the precision asked to BENSOLVE. In this sense, `PADMe` can be interpreted as a way of turning BENSOLVE - but actually any solver for bi-objective linear problems - into a solver for BOMILPs. We underline that `PADMe` is flexible in terms of how to address the slice problems. Other strategies could also be adopted, like for example a dichotomic scheme (Aneja and Nair, 1979). However, depending on the solver or algorithm adopted, this may produce a gain in efficiency but a loss in accuracy or vice versa. A further advantage in how `PADMe` is designed is in the fact that the feasible set in the criterion space is explored in an ordered manner, allowing to focus on specific ranges of the Pareto frontier.

The comparison with `TSM` concerning the number of extreme non-dominated points detected wants to be a measure of the accuracy of the Pareto frontier delivered. We can notice that the number of extreme non-dominated points detected by `PADMe` is always greater (equal in one case) than the number of extreme non-dominated points released by `TSM` after the post-processing phase. Such number increases as the dimension of the instances grows. For the C160 instances solved within one hour, we have that the number of extreme non-dominated points detected by `PADMe` is more than two times larger than the number of points detected by `TSM` before the post-processing phase. This higher accuracy can also be visualized by graphical inspection, checking the Pareto frontier obtained. In Figure 12, we depict parts of the Pareto fron-

tier of a C40 instance (left subfigure) and a C160 instance (right subfigure). It is clear that the higher number of extreme non-dominated points detected by `PADMe` results into a more accurate Pareto frontier that dominates the one detected by `TSM`.

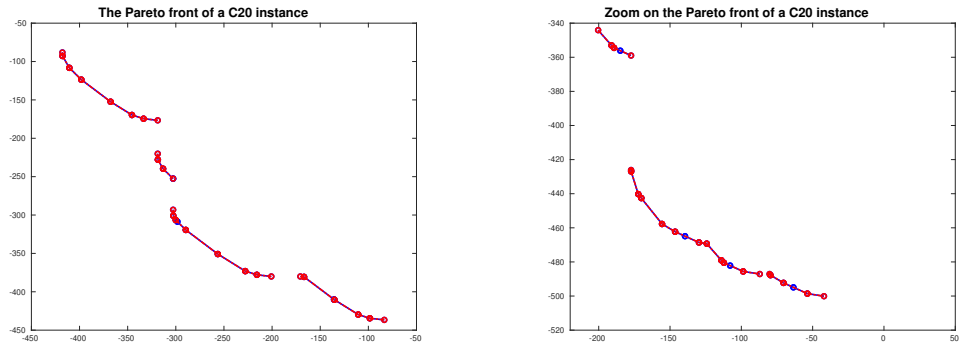


Figure 11: Examples of Pareto fronts of C20 instances.

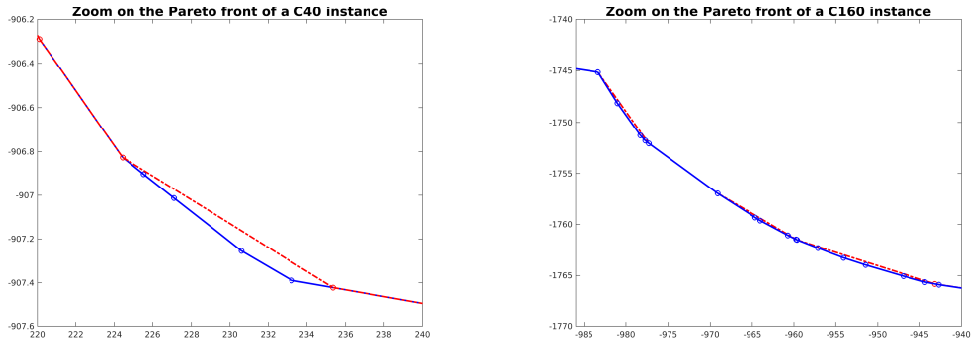


Figure 12: Comparison on the accuracy of the Pareto frontier detected. The Pareto frontier obtained by `TSM` is reported in dashed line, dominated by the Pareto frontier detected by `PADMe` (ID = 1).

5. Conclusions

We presented `PADMe`, a criterion space method able to deal with bi-objective mixed integer linear programming problems. The method alternates the resolution of mixed integer linear programming problems and bi-

Inst	PADMe (ID = 1)				TSM (Boland et al., 2015b)					EPS-tabu (Soylu and Yıldız, 2016)
	# MILPs	# BOLPs	time (s)	# endp	# MILPs	# LPs	time (s)	# endp-b	# endp-a	# MILPs
C20	**11**	5	**0.01**	27	70	8	0.12	31	26	33
	18	16	**0.03**	64	171	19	0.37	74	53	76
	22	20	**0.03**	43	151	37	0.34	45	43	66
	25	22	**0.04**	61	200	42	0.53	87	58	72
	7	5	**0.01**	31	74	4	0.12	35	29	37
Avg.	**16.6**	13.6	**0.03**	45.2	133.2	22	0.29	54.4	41.8	56.8
Max.	**25**	22	**0.04**	64	200	42	0.53	87	58	76
C40	**76**	74	**0.25**	355	763	59	4.56	373	252	380
	30	28	**0.88**	112	318	42	1.38	146	95	124
	44	41	**0.12**	137	351	51	1.97	167	112	153
	84	82	**0.27**	192	392	39	1.68	186	100	207
	106	89	**0.29**	124	319	50	1.61	148	107	142
Avg.	**68**	62.8	**0.36**	184.0	428.6	48.2	2.24	204	133.2	201.2
Max.	**106**	89	**0.88**	355	763	59	4.56	373	252	380
C80	**586**	564	**7.98**	1079	1700	197	39.42	844	296	1118
	2241	2130	65.71	719	1245	176	**24.95**	592	242	**752**
	655	609	**10.03**	962	1920	279	38.49	984	416	1024
	594	585	**9.71**	960	1841	267	42.12	904	329	1008
	492	469	**7.07**	768	1342	116	26.95	659	247	801
Avg.	**913.6**	871.4	**20.1**	897.60	1609.6	207	34.39	796.6	306	940.6
Max.	2241	2130	65.71	1079	1920	279	**42.12**	984	416	**1118**
C160	–	–	–	–	2470	546	222.68	1341	342	2888
	5450	5361	1914.26	5361	**2334**	509	**256.97**	1274	300	3070
	–	–	–	–	2260	452	221.37	1206	287	2844
	–	–	–	–	4593	987	577.38	2464	605	6353
	1356	1208	**73.78**	3144	2541	466	229.96	1310	345	3226
Avg.	–	–	–	–	2839.0	592	301.67	1519	375.8	3676.2
Max.	–	–	–	–	4593	987	577.38	2464	605	6353

Table 2: Comparison between PADMe (ID = 1) and TSM on the bi-objective instances from (Mavrotas and Diakoulaki, 1998).

objective linear ones. The method takes advantage of properly defined cutting planes in the criterion space, the so called extreme-inequalities, used to avoid the exploration of dominated regions. Under specific assumptions, `PADMe` is able to deliver the complete Pareto frontier in a finite number of iterations. From a computational point of view, the Pareto frontier is detected according to the accuracy of the solver used for the underlying bi-objective linear subproblems. `PADMe` can in fact be seen as a way to turn a solver for bi-objective linear problems into an algorithm for BOMILPs. As far as we are aware of, it is the first time that such possibility has been explored. The accuracy of `PADMe` depends on the BOLP solver employed, and the flexibility in selecting this solver allows users to prioritize either the precision of the retrieved Pareto frontier or the overall computational efficiency of the algorithm. Thanks to `BENSOLVE` (Löhne and Weißing, 2017), the solver used in our implementation of `PADMe`, our method turns out to be able to detect a higher number of extreme non-dominated points with respect to the Triangle Splitting Method (Boland et al., 2015b) and a much more accurate Pareto frontier. Such accuracy comes at the price of solving a large number of bi-objective linear subproblems, the larger the higher the dimension of the problem addressed. However, as long as the decision maker is interested in specific ranges of the Pareto frontier or in solving medium size BOMILPs, using `BENSOLVE` inside `PADMe` turns out to be both accurate and efficient. We finally want to underline that, as long as a solver for the bi-objective slice problems is available, our algorithm can be adapted to deal with bi-objective mixed integer nonlinear problems. Moving to the nonlinear case clearly introduces additional challenges. In particular, the Pareto frontier may consist of open, half-open and closed curves. This implies that we are no longer dealing with a finite set of extreme nondominated points, but rather with approximations of Pareto slices - likely constructed from finite sets of nondominated points that lie along these continuous segments. The concept of `extreme-inequalities`, as developed in the present work, would require careful generalization to remain applicable in this setting. For example, one could consider pairs of nondominated points lying on the same Pareto slice and construct linear inequalities that outer approximate these slices, thereby defining valid lower bound sets. Such inequalities could be seen as *extended* extreme-inequalities. Dealing with nonlinear objective functions is out of the scope of the present work, but we plan to explore this possibility as a future work.

Acknowledgements

The authors are very grateful to Nathan Adelgren and Pietro Belotti for their help in using the BoT data structure. They are also grateful to two anonymous referees for the careful reading of the manuscript and the valuable comments that helped to improve the paper. This research has been partially supported by Sapienza University of Rome n. RM1221816BB0FEE3 and by Unione europea - Next Generation EU grant number 20223MHHA8.

References

Adelgren, N., Belotti, P., Gupte, A., 2018. Efficient storage of pareto points in biobjective mixed integer programming. INFORMS Journal on Computing 30, 324–338.

Adelgren, N., Gupte, A., 2022. Branch-and-bound for biobjective mixed-integer linear programming. INFORMS Journal on Computing 34, 909–933.

Amorosi, L., Cedola, L., Dell'Olmo, P., Lucchetta, F., 2022. Multi-objective mathematical programming for optimally sizing and managing battery energy storage for solar photovoltaic system integration of a multi-apartment building. Engineering Optimization 54, 81–100. URL: https://doi.org/10.1080/0305215X.2020.1853715, doi:10.1080/0305215X.2020.1853715, arXiv:https://doi.org/10.1080/0305215X.2020.1853715.

Amorosi, L., Puerto, J., 2022. Two-phase strategies for the bi-objective minimum spanning tree problem. International Transactions in Operational Research 29, 3435–3463. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.13120, doi:https://doi.org/10.1111/itor.13120, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.13120.

Aneja, Y.P., Nair, K.P., 1979. Bicriteria transportation problem. Management Science 25, 73–78.

Belotti, P., Soylu, B., Wiecek, M.M., 2013. A branch-and-bound algorithm for biobjective mixed-integer programs. Optimization Online , 1–29.

Belotti, P., Soylu, B., Wiecek, M.M., 2016. Fathoming rules for biobjective mixed integer linear programs: Review and extensions. Discrete Optimization 22, 341–363.

Bökler, F., Parragh, S.N., Sinnl, M., Tricoire, F., 2024. An outer approximation algorithm for generating the edgeworth–pareto hull of multi-objective mixed-integer linear programming problems. Mathematical Methods of Operations Research , 1–28.

Boland, N., Charkhgard, H., Savelsbergh, M., 2015a. A criterion space search algorithm for biobjective integer programming: The balanced box method. INFORMS Journal on Computing 27, 735–754.

Boland, N., Charkhgard, H., Savelsbergh, M., 2015b. A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. INFORMS Journal on Computing 27, 597–618.

Caramia, M., Pizzari, E., 2023. A bi-objective cap-and-trade model for minimising environmental impact in closed-loop supply chains. Supply Chain Analytics 3, 100020. URL: https://www.sciencedirect.com/science/article/pii/S2949863523000195, doi:https://doi.org/10.1016/j.sca.2023.100020.

de Castro, P.J., Wiecek, M.M., 2025. Pareto leap: An algorithm for biobjective mixed-integer programs. Optimization Online .

Charkhgard, H., Accessed: 2023. Triangle splitting method implementation. https://usf.app.box.com/s/6i7rcdd7njkqsvnpi7x97ku9og3j8782.

De Santis, M., Eichfelder, G., Niebling, J., Rocktäschel, S., 2020a. Solving multiobjective mixed integer convex optimization problems. SIAM Journal on Optimization 30, 3122–3145.

De Santis, M., Eichfelder, G., Patria, D., 2022. On the exactness of the ε-constraint method for biobjective nonlinear integer programming. Operations Research Letters 50, 356–361.

De Santis, M., Eichfelder, G., Patria, D., Warnow, L., 2025. Using dual relaxations in multiobjective mixed-integer convex quadratic programming. Journal of Global Optimization 92, 159–186.

De Santis, M., Grani, G., Palagi, L., 2020b. Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming. European Journal of Operational Research 283, 57–69.

Ehrgott, M., 2000. Multicriteria Optimization. Lecture notes in economics and mathematical systems, Springer. URL: https://books.google.it/books?id=au6PMgEACAAJ.

Ehrgott, M., Gandibleux, X., 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum 322, 425–460. URL: https://doi.org/10.1007/s002910000046.

Eichfelder, G., Warnow, L., 2023. Advancements in the computation of enclosures for multi-objective optimization problems. European Journal of Operational Research 310, 315–327.

Fattahi, A., Turkay, M., 2018. A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems. European Journal of Operational Research 266, 415–425.

Forget, N., Parragh, S.N., 2024. Enhancing branch-and-bound for multiobjective 0-1 programming. INFORMS Journal on Computing 36, 285–304.

Gurobi Optimization, L., 2023. Gurobi optimizer reference manual. http://www.gurobi.com.

Halffmann, P., Schäfer, L., Dächert, K., Klamroth, K., Ruzika, S., 2022. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. Journal of Multi-Criteria Decision Analysis 29. doi:10.1002/mcda.1780.

Löhne, A., Weißing, B., 2017. The vector linear program solver bensolve–notes on theoretical background. European Journal of Operational Research 260, 807–813. URL: http://www.bensolve.org/.

Mavrotas, G., Diakoulaki, D., 1998. A branch and bound algorithm for mixed zero-one multiple objective linear programming. European Journal of Operational Research 107, 530–541.

Parragh, S.N., Tricoire, F., 2019. Branch-and-bound for bi-objective integer programming. INFORMS Journal on Computing 31, 805–822.

Pecin, D., Herszterg, I., Perini, T., Boland, N., Savelsbergh, M., 2024. A fast and robust algorithm for solving biobjective mixed integer programs. Mathematical Methods of Operations Research , 1–42.

Perini, T., Boland, N., Pecin, D., Savelsbergh, M., 2020. A criterion space method for biobjective mixed integer programming: The boxed line method. INFORMS Journal on Computing 32, 16–39.

Raith, A., Ehrgott, M., 2009a. A comparison of solution strategies for biobjective shortest path problems. Computers & Operations Research 36, 1299–1331. URL: https://www.sciencedirect.com/science/article/pii/S0305054808000233, doi:https://doi.org/10.1016/j.cor.2008.02.002.

Raith, A., Ehrgott, M., 2009b. A two-phase algorithm for the biobjective integer minimum cost flow problem. Computers & Operations Research 36, 1945–1954. URL: https://www.sciencedirect.com/science/article/pii/S0305054808001172, doi:https://doi.org/10.1016/j.cor.2008.06.008.

Soylu, B., 2018. The search-and-remove algorithm for biobjective mixed-integer linear programming problems. European Journal of Operational Research 268, 281–299.

Soylu, B., Yıldız, G.B., 2016. An exact algorithm for biobjective mixed integer linear programming problems. Computers & Operations Research 72, 204–213.

Steiner, S., Radzik, T., 2008. Computing all efficient solutions of the biobjective minimum spanning tree problem. Computers & Operations Research 35, 198–211. URL: https://www.sciencedirect.com/science/article/pii/S030505480600061X, doi:https://doi.org/10.1016/j.cor.2006.02.023. part Special Issue: Applications of OR in Finance.

Stidsen, T., Andersen, K.A., 2018. A hybrid approach for biobjective optimization. Discrete Optimization 28, 89–114.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., Gandibleux, X., 2010. Mavrotas and diakoulaki's algorithm for multiobjective mixed 0-1 linear programming revisited. MOPGP10 .

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., Gandibleux, X., 2013. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. Computers & Operations Research 40, 498–509.

Wang, J.Y.T., Wu, Z., Kang, Y., Brown, E., Wen, M., Rushton, C., Ehrgott, M., 2023. Walking school bus line routing for efficiency, health and walkability: A multi-objective optimisation approach. Journal of Multi-Criteria Decision Analysis 30, 109–131. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mcda.1803, doi:https://doi.org/10.1002/mcda.1803, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/mcda.1803.

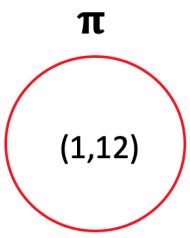
Yıldız, G.B., Soylu, B., 2019. A multiobjective post-sales guarantee and repair services network design problem. International Journal of Production Economics 216, 305–320. URL: https://www.sciencedirect.com/science/article/pii/S092552731930218X, doi:https://doi.org/10.1016/j.ijpe.2019.06.006.

Appendix

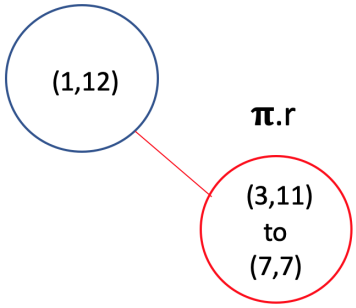
We report in this section the mathematical formulation of the illustrative example in Figure 1 (Fattahi and Turkay, 2018) used in Section 2 to describe step by step how PADMe works. We also report pictures showing the sequence of BoT updates generated along the iterations of PADMe on such example.

$$
\begin{aligned}
\min \quad & (z_1, z_2) = (x_1, x_2) \\
s.t. \quad & 9 - M(1-y_1) \le x_1 \le 9 + M(1-y_1) \\
& 1 - M(1-y_1) \le x_2 \le 1 + M(1-y_1) \\
& x_1 + 2x_2 \ge 13 - M(1-y_2) \\
& x_1 - x_2 \le 10 + M(1-y_2) \\
& x_1 + 2x_2 \le 19 + M(1-y_2) \\
& x_1 - 2x_2 \ge 3 - M(1-y_2) \\
& x_1 \ge 9 - M(1-y_2) \\
& 2x_1 + x_2 \ge 22 - M(1-y_3) \\
& 4x_1 - 3x_2 \le 24 + M(1-y_3) \\
& x_1 + x_2 \le 20 + M(1-y_3) \\
& x_1 - x_2 \ge 2 - M(1-y_3) \\
& x_1 + x_2 \ge 14 - M(1-y_4) \\
& x_1 - 2x_2 \le -7 + M(1-y_4) \\
& x_1 - x_2 \le 1 + M(1-y_4) \\
& 3x_1 + 10x_2 \le 159 + M(1-y_4) \\
& x_1 \ge 3 - M(1-y_4) \\
& x_1 \ge 5 - M(1-y_5) \\
& x_1 - x_2 \le -3 + M(1-y_5) \\
& x1 + x2 \le 17 + M(1-y_5) \\
& x_1 - x_2 \ge -5 - M(1-y_5) \\
& 4x_1 - x_2 \ge 5 - M(1-y_6) \\
& 3x_1 - 8x_2 \le -76 + M(1-y_6) \\
& x_1 + x_2 \le 26 + M(1-y_6) \\
& x_2 \le 15 + M(1-y_6) \\
& 1 - M(1-y_7) \le x_1 \le 1 + M(1-y_7) \\
& 12 - M(1-y_7) \le x_2 \le 12 + M(1-y_7) \\
& x_2 \ge 13 - M(1-y_8) \\
& x_2 \le 15 + M(1-y_8) \\
& -7 - M(1-y_8) \le 4x_1 - x_2 \le -7 + M(1-y_8) \\
& y_1 + y_2 + \cdots + y_8 = 1 \\
& x_k \ge 0 \quad k = 1, 2 \\
& y_i \in \{0, 1\} \quad i = 1, \ldots, 8.
\end{aligned}
$$

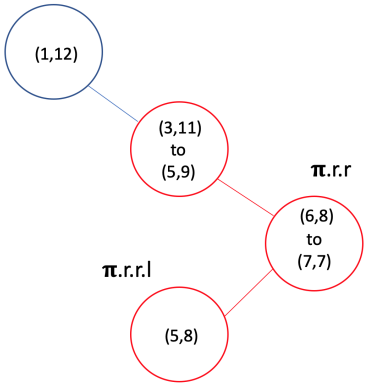
The model proposed in (Fattahi and Turkay, 2018), has $(z_1, z_2) = (x_1, x_2)$, being x_1 and x_2 the only two continuous variables. The other eight variables in the model are binary. A sufficiently large parameter M is used to model the constraints. Due to the constraint imposing that the sum of the binary variables must be equal to 1, we can easily see that for each feasible fixing we have a set of linear inequalities defining one polyhedron in the feasible set. We can observe that this is only one possible formulation associated with the example whose feasible region in the objective space is represented in Figure 1.



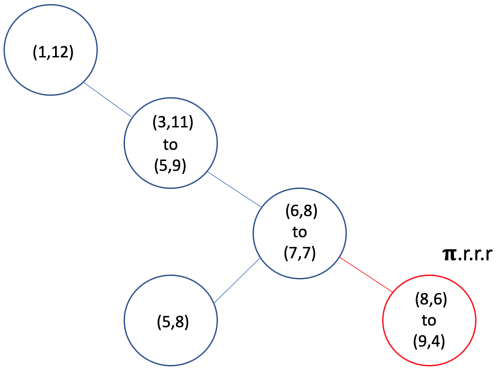
(a) Initial BoT (associated with Figure 2(a))



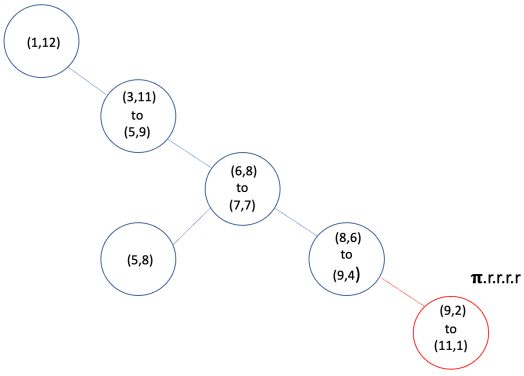
(b) Updated BoT (associated with Figure 3(a))



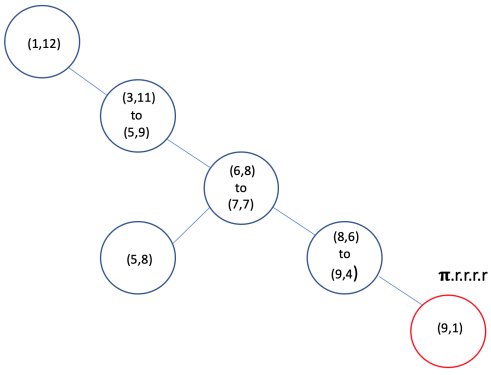
(c) Updated BoT (associated with Figure 4(b))



(d) Updated BoT (associated with Figure 5(b))



(e) Updated BoT (associated with Figure 7(a))



(f) Final BoT (associated with Figure 7(b))

Figure 13: Sequence of BoT updates generated by running `PADMe` on the example in Figure 1 (Fattahi and Turkay, 2018)