On the accurate detection of the Pareto frontier for bi-objective mixed integer linear problems

Lavinia Amorosi* and Marianna De Santis†

Abstract

We propose a criterion space search algorithm for 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. Our algorithm works by alternating the resolution of single objective mixed integer linear problems with bi-objective linear ones. Along the iterations of the algorithm, the non-dominated points and line segments found are stored in an ordered manner, using the tree data-structure proposed in [1]. The performance of the algorithm is improved using suitably defined cuts and related strategies. Under specific assumptions, we can prove 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 [8] 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.

Key Words: Bi-objective Programming, Mixed-Integer Linear Programming, Criterion Space Algorithm.

Mathematics subject classifications (MSC 2010): 90C11, 90C29, 90C57.

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 the energy sector the design and management of renewable energy

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

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

storage systems is performed by taking into account both installation costs and energy self-sufficiency [3]. In the mobility sector, the planning of walking routes is done by minimizing travel times but also maximizing the air quality of the route [32]. In supply chain, maximization of profit and minimization of CO2 emissions are simultaneously performed [9]. 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, the study in this research area is producing increasingly efficient algorithms capable of dealing with instances of increasing size [20].

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 [14]. The class of multi-objective integer or binary linear optimization problems has also been the focus of many works from the literature [15]. 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. [26, 29, 3, 27]). Recently, works dealing with bi- and multi-objective mixed integer nonlinear optimization problems have also been proposed (see e.g. [11, 12, 13, 16]).

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 $I \subseteq \{1, \dots, n\}$. We denote by $C = \{1, \dots, n\} \setminus I$ 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*

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

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*

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

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

The set of all non-dominated points of a BOMILP is called non-dominated set or also 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 [17]. Since the feasible set in the image space of a BOMILP is an intersection 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* [4]. 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 our algorithm to cut dominated region in the image space. Within Section 2.2, we provide an example to describe in details how our algorithm works. 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 [8] is shown. Section 5 concludes.

1.1 Literature Review

Since the first contribution by Mavrotas and Diakoulaki [22], three main categories of algorithms have been proposed in the literature for solving multiob-

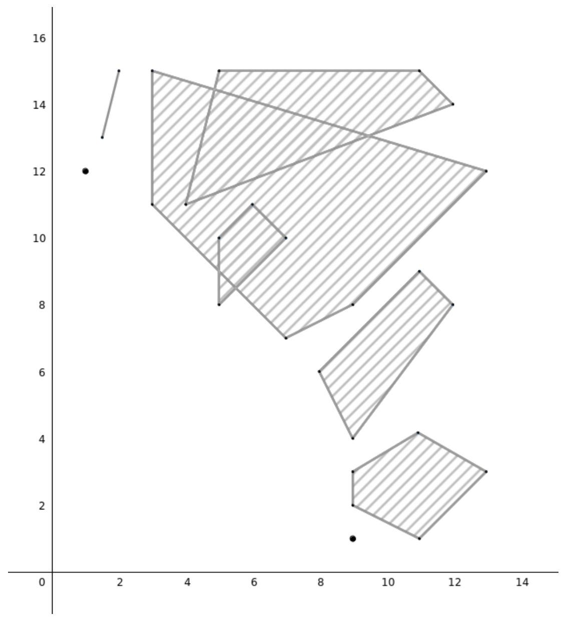


Figure 1: The feasible set in the image space of a BOMILP [17]. Its Pareto frontier contains isolated points, half-open and closed line segments.

jective mixed integer linear problems (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. [22, 30, 31, 6, 23, 18]) and in particular with BOMILPs (see [4, 5, 28, 2]). 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 [8] 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 repeated. 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 [28] the ϵ-*Tabu Constraint Algorithm* is introduced. As in [8], 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 [7, 17, 25, 24] as additional methods appeared in the literature to deal with BOMILPs. We also refer to [20] for a comprehensive overview on solution approaches for multiobjective mixed integer linear problems.

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. The scheme of our method is reported in Algorithm 1. The idea is to detect the efficient integer assignments of (BOMILP) and the related extreme non-dominated points, scrolling the feasible set in the image space from top-left to bottom-right. As a first step, 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. [4]). In particular, since x_I^0 is obtained by minimizing $z_1(x)$, the first identified polyhedron is located in the upper-left part of the feasible set in the image space. Its extreme non-dominated points 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. Then, we enter in a loop, in order to explore the feasible set in the image space until the minimum with respect to $z_2(x)$ 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 $x^{id,2} = (x_I^{id,2}, x_C^{id,2})$. At each iteration, we identify the new slice problem to address and a related potential efficient integer assignment as follows. We first solve a single-objective mixed integer linear sub-problem, where $z_1(x)$ is minimized over the original feasible set X intersected with two constraints bounding the values of $z_2(x)$ and $z_1(x)$, respectively and a bunch of so called *no good/Tabu* constraints [28]:

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

Following the ideas in [28], 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. 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 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 $z_1(x)$ 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 $\mathcal{S}^k$. Once problem (MILP_k) is solved, we address a sequence of bi-objective linear problems (BOLPs), in order to detect the partial potential Pareto frontier of the slice problem related to $x_I = x_I^k$. These BOLPs are defined using specific cuts in the image space, in order to avoid the exploration of dominated regions. Details are given in section 2.1.

For how it is defined, our method relies on the availability of two solvers: one able to address mixed integer linear problems (such as, e.g., GUROBI [19]), one able to address bi-objective linear problems (such as, e.g., BENSOLVE [21]). In order to efficiently store and filter the points and line segments detected at each iteration, we use the data structure developed in [1], 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 [1] (see Algorithm 2).

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

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 consider specific cuts. 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 $i \in \{1, \dots, p^{k-1}\}$.

Then, within a loop on the index $i \in \{1, \dots p^{k-1}\}$ 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 3.2), 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. Examples in this respect are given in Section 2.2.

2.2 Illustrative example

In order to better explain how Algorithm 1 works, we describe its iterations when applied to the instance proposed in [17], whose feasible set in the image space is reported in Figure 1. 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). The point z_2^* is also detected (see Figure 2 (a)). The corresponding slice problem, once the integer variables are fixed to x_I^0, is the singleton $Y^0 = \{y_1^0\}$ that is inserted in the BoT data structure π. Note that, in this case, $p^0 = 1$. Then, we solve problem (MILP$_k$), $k = 1$, imposing $z_2(x) \leq$

Algorithm 1: PADMe = Pareto Accurate Detection Method

Input: (BOMILP), $k = 0$, $\pi = \emptyset$

Output: the Pareto frontier $\mathcal{Y}_N$ of $(BOMILP)$;

Compute $x^{id,1} = (x_I^0, x_C^0) \in arg\min_{x \in X} z_1(x)$

Compute the set $Y^0 = \{y_1^0, \ldots, y_{p^0}^0\}$ of extreme non-dominated points of $\min_{x \in X, x_I = x_I^0}(z_1(x), z_2(x))^T$

for $j = 1, \ldots, p^0 - 1$ **do**
- $\pi^* = (((y_j^0)_1, (y_j^0)_2), (y_{j+1}^0)_1, (y_{j+1}^0)_2)), \emptyset, \emptyset)$
- Insert(π^*, π)

end

Compute $x^{id,2} = (x_I^{id,2}, x_C^{id,2}) \in arg\min_{x \in X} z_2(x)$

while $(z_2((x_I^k, x_C^k)) > (z_{id})_2$ *&* $\mathcal{S}^k \neq \emptyset)$ **do**
- **Set** $k = k + 1$
- **Compute** (x_I^k, x_C^k) by solving (MILP$_k$)
- **Set** $Y^k = \emptyset$
- **Set** F^k as in (1)
- **for** $i = 1, \ldots, p^{k-1}$ **do**
  - **if** $(p^{k-1} == 1)$ **then**
    - **Compute** the set $Y^{i,k} = \{y_1^{i,k}, \ldots, y_{p^{i,k}}^{i,k}\}$ of extreme non-dominated points of $\min_{x \in F^k}(z_1(x), z_2(x))^T$
  - **end**
  - **else**
    - **Compute** w_i as in (2)
    - **Set** W_i^k as in (3)
    - **if** $W_i^k \neq \emptyset$ **then**
      - **Compute** the set $Y^{i,k} = \{y_1^{i,k}, \ldots, y_{p^{i,k}}^{i,k}\}$ of extreme non-dominated points of $\min_{x \in W_i^k}(z_1(x), z_2(x))^T$
    - **end**
  - **end**
  - $Y^k = Y^k \cup Y^{i,k}$
  - **Update** π by Algorithm 2
- **end**
- **if** $Y^k = \emptyset$ **then**
  - **Set** $Y^k = Y^{k-1}$
- **end**
- **Set** $p^k = |Y^k|$

end

Return $\mathcal{Y}_N = \pi$

Algorithm 2: Update of the Bi-Objective Tree data structure [1]

Input: $Y^{i,k}$, $p^{i,k}$, π

if $p^{i,k} == 1$ **then**

  $\pi^* = (((y_1^{i,k})_1, (y_1^{i,k})_2), (y_1^{i,k})_1, (y_1^{i,k})_2)), \emptyset, \emptyset)$

  $\text{Insert}(\pi^*, \pi)$

end

else

  for $j = 1, \ldots, p^{i,k} - 1$ **do**

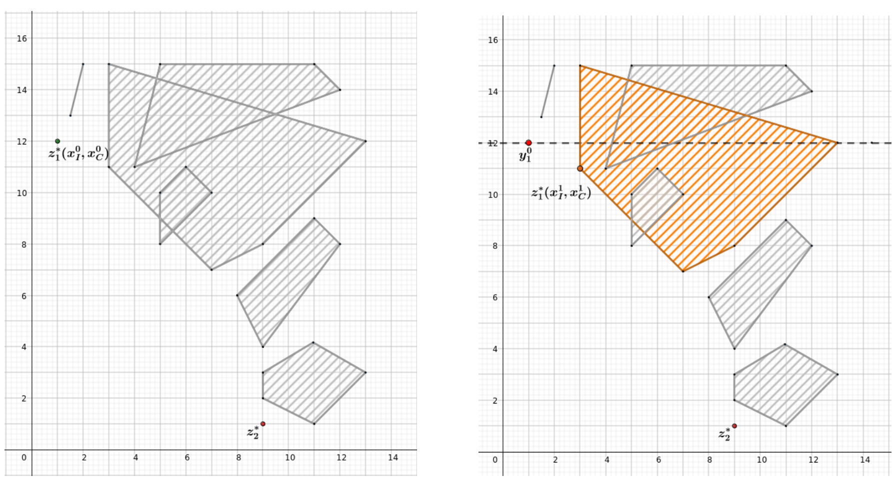
    $\pi^* = (((y_j^{i,k})_1, (y_j^{i,k})_2), (y_{j+1}^{i,k})_1, (y_{j+1}^{i,k})_2)), \emptyset, \emptyset)$

    $\text{Insert}(\pi^*, \pi)$

  end

end

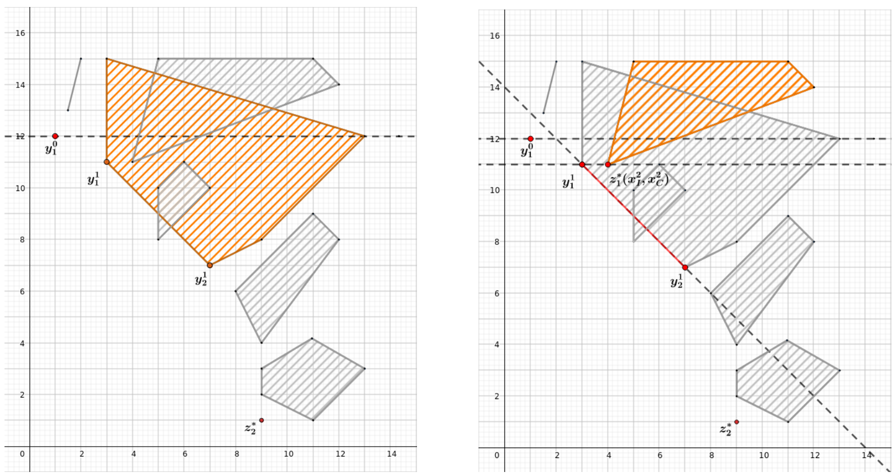
$(y_1^0)_2$, $z_1(x) \leq z_1(x^{id,2})$ and the Tabu constraint $x_I \neq x_I^0$. The solution (x_I^1, x_C^1) is obtained (see Figure 2 (b)) and the extreme non-dominated points $\{y_1^1, y_2^1\}$ are detected, by addressing the BOLP $\min_{x \in F^1}(z_1(x), z_2(x))^T$ (see Figure 3 (a)). Therefore, $Y^1 = \{y_1^1, y_2^1\}$, $p^1 = 2$ and π is updated accordingly. Note 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.



(a) (b)

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

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) (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 $(w_1)^T z(x) \leq (w_1)^T y_1^1$. Such intersection is empty so that we discard the related slice problem, as it cannot



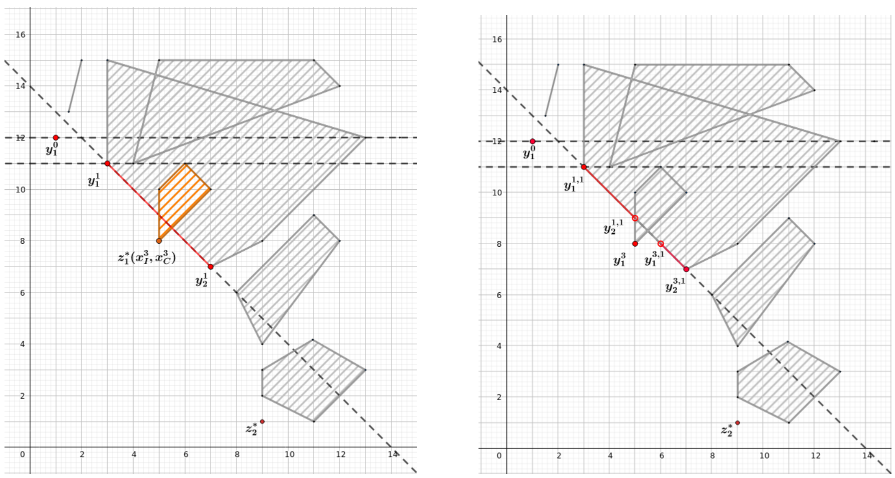
(a) (b)

Figure 3: Illustration of PADMe 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.

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) (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 as single 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 $[y_1^{1,1}, y_2^{1,1}]$ and $[y_1^{3,1}, y_2^{3,1}]$, together with the point y_1^3 are kept in the BoT data structure π (see Figure 4 (b)).

In Figure 5, the fourth iteration is shown. The new polyhedron to explore is higlighted in orange (see 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 $[y_1^4, y_2^4]$ is detected and memorized within π (see Figure 5 (b)). 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 $(w_1)^T z(x) \leq (w_1)^T y_1^4$ as feasible set. The components of $w_1 \in \mathbb{R}^2$ are computed according to (2), considering y_1^4 and y_2^4. The extreme non-dominated points detected will be $y_1^{1,5} = y_1^5$ 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



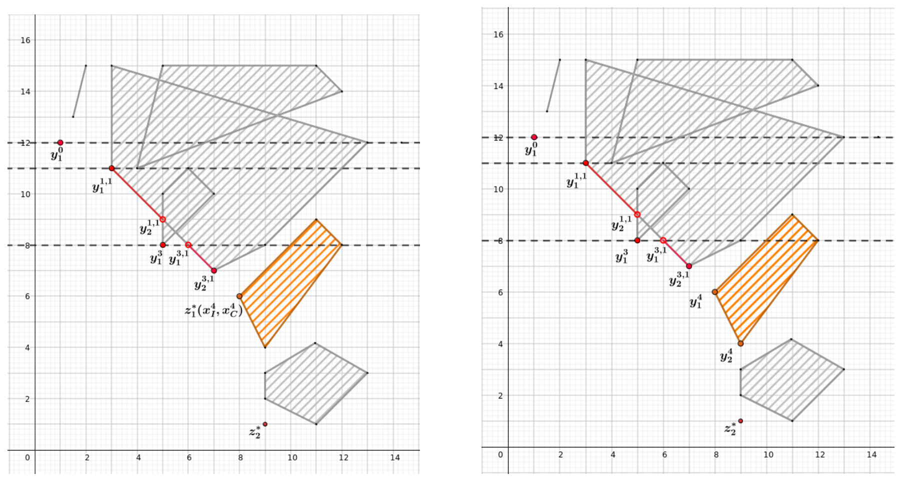
(a) (b)

Figure 4: Illustration of `PADMe` 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.

$y_1^{2,5} = y_1^5$ and $y_2^{2,5}$ will be detected and will 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 $y_1^6 = z(x_I^6, x_C^6)$ 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, so that the previously detected line segment $[y_1^5, y_2^{2,5}]$ 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.

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. For the correctness of Algorithm 1, we need to make a first standard assumption:



(a) (b)

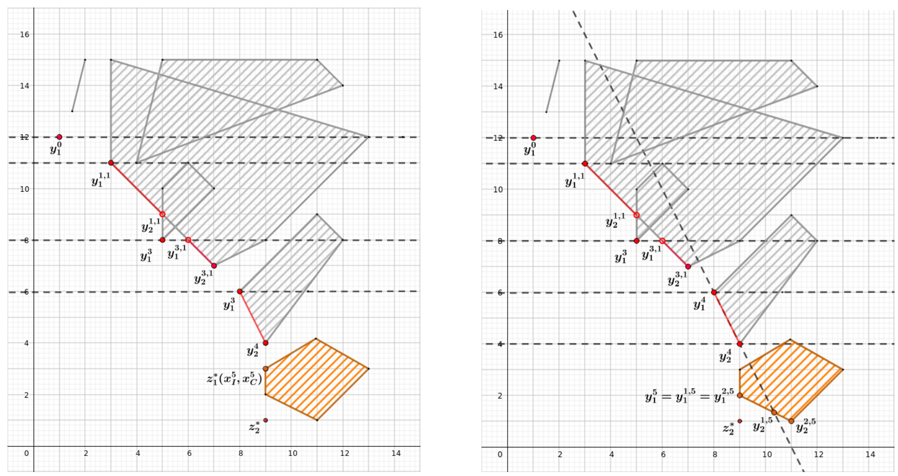
Figure 5: Illustration of `PADMe` on a BOMILP instance: the fourth iteration. By solving problem (MILP$_4$) the point (x_I^4, x_C^4) is detected (a). Once the related slice problem is addressed the set $Y^4 = \{y_1^4, y_2^4\}$ is detected (b).

Assumption 3.1. *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.*

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 efficient-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.

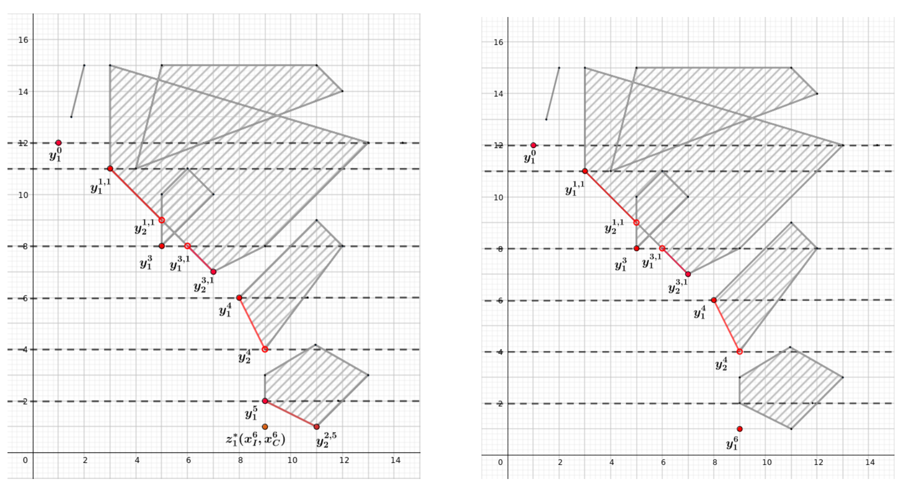
Proposition 3.2. *Given* (BOMILP)*, let Assumption 3.1 holds. Let* $y \in \mathcal{Y}_N$*. Then,* $k \in \mathbb{N}$ *exists such that* y *belongs to the partial potential Pareto frontier obtained from* Y^k *in Algorithm 1.*

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



(a) (b)

Figure 6: Illustration of PADMe 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 on a BOMILP instance: the sixth iteration. The complete non-dominated set detected by PADMe is reported in subfigure (b) and highlighted in red.

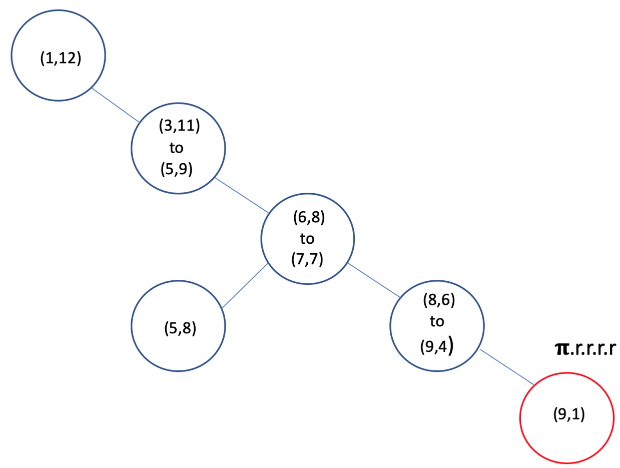


Figure 8: The BoT (bi-objective tree) [1] associated with the Pareto frontier of the BOMILP instance solved by `PADMe`.

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, we can prove that Algorithm 1 exactly detects its non-dominated frontier with no additional assumption but Assumption 3.1:

Theorem 3.3. *Given* (BOMILP)*, assume that the integer variables are constrained to be binary, namely $x_i \in \{0, 1\}$, for $i \in I$, with $|I| \leq n$. Let Assumption 3.1 holds. Then, Algorithm 1 detects the Pareto frontier of* (BOMILP) *in a finite number of iterations.*

Proof. From Proposition 3.2, 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. Furthermore, under the assumption that the integer variables are constrained to be binary, a finite number of feasible integer assignments exists, so that a finite number of slice problem exists. □

In order to prove that our algorithm stops in a finite number of iterations in the general integer case, we need to make a specific assumption. Indeed, we need to exclude cases where non-dominated points or segments belong to the partial potential Pareto frontier of an infinite number of slice problems. Such situation can indeed occur, as Example 3.4 shows.

Example 3.4. *Consider the problem*

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

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.*

Assumption 3.5. *Given* (BOMILP) *and a non-dominated point or line segment, we assume that at most* $q \in \mathbb{N}$ *slice problems share such point or line segment as partial potential Pareto frontier .*

Under Assumption 3.5, we are able to prove the following result.

Theorem 3.6. *Given* (BOMILP)*, let Assumption 3.1 and Assumption 3.5 hold. Then, Algorithm 1 detects the Pareto frontier of* (BOMILP) *in a finite number of iterations.*

Proof. From Assumption 3.1, we have that the non-dominated set $\mathcal{Y}_N$ is a bounded set. Indeed, $\mathcal{Y}_N$ is contained in the box $[z_1^{id}, z_1(x^{id,2})] \times [z_2^{id}, z_2(x^{id,1})]$. In particular, thanks to Assumption 3.5, we have that a finite number of slice problems belongs to the box $[z_1^{id}, z_1(x^{id,2})] \times [z_2^{id}, z_2(x^{id,1})]$. Since in the detection of the new slice problem, i.e. in the solution of problem (MILP$_k$), the constraint $z_1(x) \leq z_1((x^{id,2}))$ is included, we have that Algorithm 1 analyzes a finite number of slice problems. From Proposition 3.2, we have that this finite number of slice problems suffices to detect the exact non-dominated set of (BOMILP), as 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. □

Remark 3.7. *Assumption 3.5 is naturally satisfied by binary problems, where the number of slice problems is finite. On the other hand, when dealing with general integer instances, such assumption is needed by any algorithm making use of Tabu constraints to exclude already analyzed slice problems (such as the one proposed in [28]), in order to ensure finiteness.*

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 [22]. This class has been used in the majority computational studies of algorithms for solving BOMILPs and are available at [10] 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 have 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 [19] as solver for the MILP subproblems and BENSOLVE [21] as solver for the BOLP subproblems. We use default parameters for both solvers. Furthermore, as already mentioned, we use the BoT data structure proposed in [1] 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`. We underline that `PADMe` can be interpreted as a way of turning a solver for bi-objective linear problems (BENSOLVE in our implementation), into a solver for BOMILPs. Therefore, a first test we propose, is related to the performance of BENSOLVE in combination with the extreme-inequalities introduced. In the original version of `PADMe`, as reported in Algorithm 1, the extreme-inequalities are used to help the BOLP solver by reducing the feasible region of the BOLP subproblems that have to be addressed. However, the number of BOLPs to be solved at iteration k of Algorithm 1 depends on the cardinality of Y^{k-1}, that in turn depends on the dimension of the instance. Such number soon becomes prohibitive as the dimension of the instances grows (see Table 1). We then compare Algorithm 1, with two other versions, differing on the way the extreme-inequalities are used. In a version called `PADMe` (only some), we consider only *meaningful* extreme-inequalities. More precisely, 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 a further version of Algorithm 1, called `PADMe` (ID check), 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, after having detected a new integer assignment, we check 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 4.1. An example of such situation is depicted in Figure 10. 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 9.

Proposition 4.1. *Given* (BOMILP)*, let* z_{id}^k *be the ideal point of the slice*

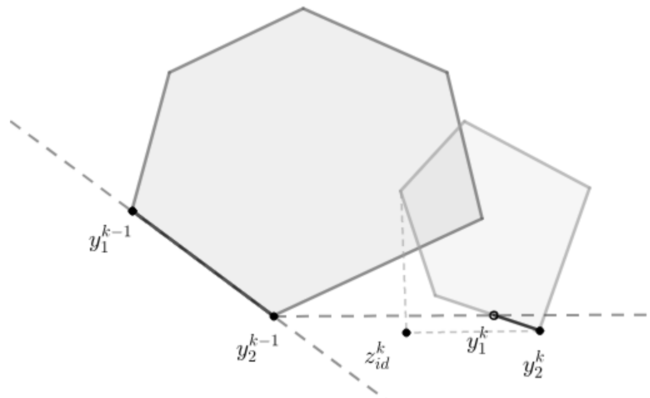


Figure 9: Use of the extreme-inequalities in combination with the ideal point: in this case, z_{id}^k is not cut by every extreme-inequality. The slice problem is then addressed and the potential extreme non-dominated points y_1^k and y_2^k are added to the BoT data structure.

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 $i \in \{1, \dots, p^{k-1}\}$, no extreme non-dominated point can be detected by 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 $i \in \{1, \dots, p^{k-1}\}$ 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 Table 1, we report a comparison among the original Algorithm 1 (referred as to `PADMe` (all cuts)), the version where only some cuts are considered, with $\eta = 0.1$ (referred as to `PADMe` (only some)) and the version where the strategy related to the ideal point is implemented (referred as to `PADMe` (ID check)).

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

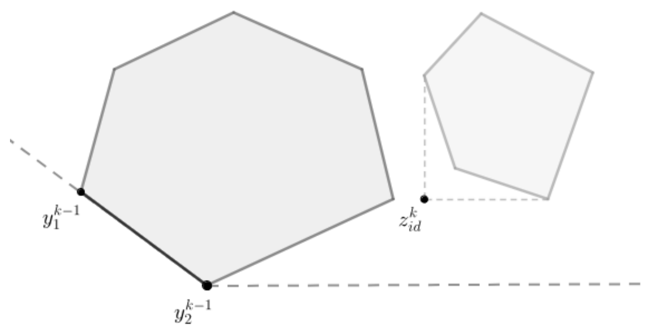


Figure 10: Illustration of Proposition 4.1. If the ideal point z_{id}^k is cut by every extreme-inequality, the current slice problem can be discarded. In the figure such case is depicted: $(w_i)^T z_{id}^k > (w_i)^T y_i^{k-1}$ for all $i \in \{1, \dots, p^{k-1}\}$.

the benefit of reducing the number of BOLPs to be solved and then the overall computational time, as the performance of `PADMe` (only some - $\eta = 0.1$) 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 check) 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 cuts) 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 check) version with the Triangle Splitting Method (`TSM`) [8], that is a criterion space method designed for bi-objective problems. We mention that a comparison with branch-and-bound approaches working in the decision space would not be fair, as criterion search algorithms like ours and `TSM` strongly exploit the advantage of working into a 2-dimensional space, as already mentioned in Section 1.1.

In Table 2, we report the results obtained by our method and `TSM` on the bi-objective mixed-binary instances proposed in [22] and available at [10]. 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

Inst	PADMe (all cuts)			PADMe (only some - $\eta = 0.1$)			PADMe (ID check)		
	# 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 [22]. Each version differs in the way the extreme-inequalities are used in combination with BENSOLVE.

subproblems (# 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 [28].

For what concerns the efficiency, we can notice that `PADMe` compares favorably in terms of number of MILPs addressed 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 is strongly related to the accuracy asked to `PADMe` in detecting the Pareto frontier.

In this respect, 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 frontier 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`.

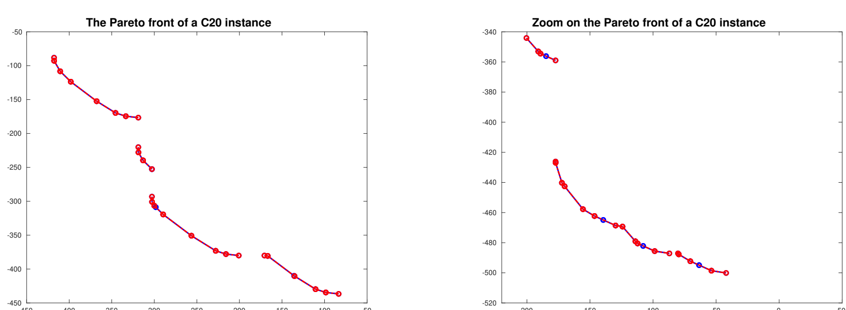


Figure 11: Examples of Pareto fronts of C20 instances.

Inst	PADMe (ID check)				TSM [8]					EPS-tabu [28]
	# 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 check) and TSM on the bi-objective instances from [22].

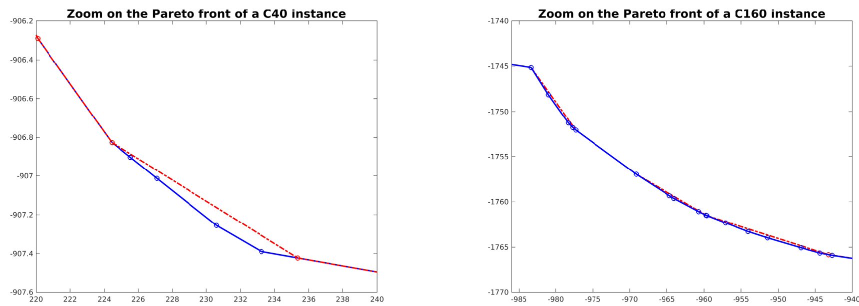


Figure 12: Comparison on the accuracy of the Pareto frontier detected. The Pareto frontier obtained by TSM is reported in dashed line, dominated by the Pareto frontier detected by PADMe (ID check).

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-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. Thanks to the good performance of BENSOLVE [21], 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 [8] 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, 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 and 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.

References

[1] Nathan Adelgren, Pietro Belotti, and Akshay Gupte. Efficient storage of pareto points in biobjective mixed integer programming. *INFORMS Journal on Computing*, 30(2):324–338, 2018.

[2] Nathan Adelgren and Akshay Gupte. Branch-and-bound for biobjective mixed-integer linear programming. *INFORMS Journal on Computing*, 34(2):909–933, 2022.

[3] Lavinia Amorosi, Luca Cedola, Paolo Dell'Olmo, and Francesca Lucchetta. 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(1):81–100, 2022.

[4] Pietro Belotti, Banu Soylu, and Margaret M Wiecek. A branch-and-bound algorithm for biobjective mixed-integer programs. *Optimization Online*, pages 1–29, 2013.

[5] Pietro Belotti, Banu Soylu, and Margaret M Wiecek. Fathoming rules for biobjective mixed integer linear programs: Review and extensions. *Discrete Optimization*, 22:341–363, 2016.

[6] Fritz Bökler, Sophie N Parragh, Markus Sinnl, and Fabien Tricoire. An outer approximation algorithm for generating the edgeworth–pareto hull of multi-objective mixed-integer linear programming problems. *Mathematical Methods of Operations Research*, pages 1–28, 2024.

[7] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS Journal on Computing*, 27(4):735–754, 2015.

[8] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618, 2015.

[9] Massimiliano Caramia and Emanuele Pizzari. A bi-objective cap-and-trade model for minimising environmental impact in closed-loop supply chains. *Supply Chain Analytics*, 3:100020, 2023.

[10] Hadi Charkhgard. Triangle splitting method implementation. `https://usf.app.box.com/s/6i7rcdd7njkqsvnpi7x97ku9og3j8782`, Accessed: 2023.

[11] Marianna De Santis, Gabriele Eichfelder, Julia Niebling, and Stefan Rocktäschel. Solving multiobjective mixed integer convex optimization problems. *SIAM Journal on Optimization*, 30(4):3122–3145, 2020.

[12] Marianna De Santis, Gabriele Eichfelder, and Daniele Patria. On the exactness of the ε-constraint method for biobjective nonlinear integer programming. *Operations Research Letters*, 50(3):356–361, 2022.

[13] Marianna De Santis, Giorgio Grani, and Laura Palagi. Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming. *European Journal of Operational Research*, 283(1):57–69, 2020.

[14] M. Ehrgott. *Multicriteria Optimization.* Lecture notes in economics and mathematical systems. Springer, 2000.

[15] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 322:425–460, 2000.

[16] Gabriele Eichfelder and Leo Warnow. Advancements in the computation of enclosures for multi-objective optimization problems. *European Journal of Operational Research*, 310(1):315–327, 2023.

[17] Ali Fattahi and Metin Turkay. A one direction search method to find the exact nondominated frontier of biobjective mixed-binary linear programming problems. *European Journal of Operational Research*, 266(2):415–425, 2018.

[18] Nicolas Forget and Sophie N Parragh. Enhancing branch-and-bound for multiobjective 0-1 programming. *INFORMS Journal on Computing*, 36(1):285–304, 2024.

[19] LLC Gurobi Optimization. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2023.

[20] Pascal Halffmann, Luca Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 29, 03 2022.

[21] Andreas Löhne and Benjamin Weißing. The vector linear program solver bensolve–notes on theoretical background. *European Journal of Operational Research*, 260(3):807–813, 2017.

[22] George Mavrotas and Danae Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[23] Sophie N Parragh and Fabien Tricoire. Branch-and-bound for bi-objective integer programming. *INFORMS Journal on Computing*, 31(4):805–822, 2019.

[24] Diego Pecin, Ian Herszterg, Tyler Perini, Natashia Boland, and Martin Savelsbergh. A fast and robust algorithm for solving biobjective mixed integer programs. *Mathematical Methods of Operations Research*, pages 1–42, 2024.

[25] Tyler Perini, Natashia Boland, Diego Pecin, and Martin Savelsbergh. A criterion space method for biobjective mixed integer programming: The boxed line method. *INFORMS Journal on Computing*, 32(1):16–39, 2020.

[26] Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299–1331, 2009.

[27] Andrea Raith and Matthias Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36(6):1945–1954, 2009.

[28] Banu Soylu and Gazi Bilal Yıldız. An exact algorithm for biobjective mixed integer linear programming problems. *Computers & Operations Research*, 72:204–213, 2016.

[29] Sarah Steiner and Tomasz Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research*, 35(1):198–211, 2008. Part Special Issue: Applications of OR in Finance.

[30] Thomas Vincent, Florian Seipp, Stefan Ruzika, Anthony Przybylski, and Xavier Gandibleux. Mavrotas and diakoulaki's algorithm for multiobjective mixed 0-1 linear programming revisited. *MOPGP10*, 2010.

[31] Thomas Vincent, Florian Seipp, Stefan Ruzika, Anthony Przybylski, and Xavier Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40(1):498–509, 2013.

[32] Judith Y. T. Wang, Zhengyu Wu, Yating Kang, Edward Brown, Mengfan Wen, Christopher Rushton, and Matthias Ehrgott. Walking school bus line routing for efficiency, health and walkability: A multi-objective optimisation approach. *Journal of Multi-Criteria Decision Analysis*, 30(3-4):109–131, 2023.