Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming

Branching with Hyperplanes in the Criterion Space: the Frontier Partitioner Algorithm for Biobjective Integer Programming

Marianna De Santis[*,1], **Giorgio Grani**[*,2], **Laura Palagi**[*,3]

Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome. Via Ariosto 25, Rome, 00185, Italy. E-mail: marianna.desantis\g.grani\laura.palagi at uniroma1.it; Tel: +390677274 078\086\081

Abstract

We present an algorithm for finding the complete Pareto frontier of biobjective integer programming problems. The method is based on the solution of a finite number of integer programs. The feasible sets of the integer programs are built from the original feasible set, by adding cuts that separate efficient solutions. Providing the existence of an oracle to solve suitably defined single objective integer subproblems, the algorithm can handle biobjective nonlinear integer problems, in particular biobjective convex quadratic integer optimization problems. Our numerical experience on a benchmark of biobjective integer linear programming instances shows the efficiency of the approach in comparison with existing state-of-the-art methods. Further experiments on biobjective integer quadratic programming instances are reported.

Keywords: Multiobjective Optimization; Integer Programming; Criterion Space Search

1. Introduction

Most real-world optimization problems in the areas of applied sciences, engineering and economics involve multiple, often conflicting, goals. In the mathematical modelling of these problems, under the necessity of reflecting discrete quantities, logical relationships or decisions, integer and 0-1 variables need to be considered. We are in the context

[1]Corresponding author. This author acknowledges support within the project "Nonlinear Approaches for the Solution of Hard Optimization Problems with Integer Variables" (2017) (No RP11715C7D8537BA) which has received funding from Sapienza, University of Rome.

[2]This author acknowledges support within the project "Exact Approaches for Solving Multiobjective Integer Optimization Problems" (2018) (No RP1181641D22304F) which has received funding from Sapienza, University of Rome.

[3]This author acknowledges support within the project "Distributed optimization algorithms for Big Data" (2017) (No RM11715C7E49E89C) which has received funding from Sapienza, University of Rome.

of multiobjective integer programming (MOIP) and the generic MOIP problem can be stated as follows:

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} y(x) = \min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (y_1(x), \ldots, y_p(x)) \qquad \text{(MOIP)}$$

where $y_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, p$, $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{X} \cap \mathbb{Z}^n$ represents the feasible set in the decision space. The image of $\mathcal{X} \cap \mathbb{Z}^n$ under the vector-valued function $y : \mathbb{R}^n \to \mathbb{R}^p$ is called the image space and is denoted by $\mathcal{Y} = \{z \in \mathbb{R}^p : z = y(x) \text{ for some } x \in \mathcal{X} \cap \mathbb{Z}^n\}$. The challenging nature of MOIPs and the need for methods with guaranteed performance, motivated the development of exact approaches for multiobjective integer programming problems. Algorithms for multiobjective optimization can be divided into decision space search algorithms, i.e., approaches that search in the space of feasible solutions, and criterion space search algorithms, i.e., methods that search in the space of objective function values.

Among the decision space search algorithms, the first branch-and-bound algorithm for solving multiobjective mixed 0-1 integer programs was proposed by Mavrotas and Diakoulaki [24], who improved and extended their work in [23, 25]. The method proposed is a depth-first branch and bound algorithm where nodes are pruned on the base of ideal vectors. Later in [32], the authors propose a branch-and-bound algorithm for multiobjective integer linear programming problems extending the bounding procedure introduced in [16]: the aim in [32] is to find separating hypersurfaces in the objective space between the upper and lower bound sets in order to prune the current node in the enumeration tree. More recently, Belotti and coauthors proposed advanced branch-and-bound algorithms for biobjective mixed-integer problems [2, 3]. They focus on the idea of finding the complete Pareto frontier for a relaxed subproblem, using this information to derive practical fathoming rules. We further mention [30, 33] as algorithms for biobjective mixed integer linear programming problems. For the special case of linear BOIPs, the criterion space search algorithm defined in [30] requires the solution of $2|\mathcal{Y}_N| - 1$ integer programs.

Criterion space search algorithms find non-dominated points by addressing a sequence of single-objective optimization problems. Once a non-dominated point is computed, the dominated parts of the criterion space are removed and the algorithms go on looking for new non-dominated points. One of the first criterion space search algorithms for solving biobjective integer programming problems was proposed by Chalmet et al. [11], and it is strongly related to the basic scheme of our algorithm as we explain later. Another important criterion space search algorithm is the one proposed in [34] and improved in [20, 22]. Several contributions in the context of criterion space search algorithms have been given by Boland and coauthors [4, 5, 6, 7] and indeed the balanced box method proposed in [4] is considered one of the state-of-the-art method for biobjective integer linear programming problems. In our numerical experience, we compare our algorithm with the balanced box method proposed in [4], one of the state-of-the-art method for biobjective integer linear programming problems. The balanced box method maintains a priority queue with rectangles that are explored in order to detect non-dominated points. In [13, 21] new theoretical insights on how to efficiently update the search region in branch-and-bound algorithms are given. In particular, in [13],

assuming that a set of solutions is already known, an efficient algorithm to identify a minimal set of search zones that decompose the search region is proposed.
In the application context, we mention works on biobjective minimum cost flow problems [26, 28, 31], on network routing problems [29], on the stable robotic flow shop scheduling problem [12] as well as the assignment problem with three objectives [27].
In this paper, we focus on biobjective integer programming, i.e. Problem (MOIP) with $p = 2$. We propose a branch-and-cut algorithm, called the Frontier Partitioner Algorithm (FPA), that belongs to the class of criterion space search algorithms.

Our contribution. We provide an Algorithm for biobjective integer problems able to detect the complete Pareto frontier by solving $|\mathcal{Y}_N|+2$ single objective integer programs, where $|\mathcal{Y}_N|$ is the number of non-dominated points. To the best of our knowledge, this is one of the best bounds for this class of algorithms. Our approach can be used for handling nonlinear convex integer biobjective problems since the type of cuts introduced to partition the criterion space is linear in the criterion space and convex in the decision space as long as the objective functions of the problem are convex. This property allows us to tackle nonlinear convex integer biobjective problems, under the assumption that an oracle to solve the integer convex subproblems arising in the branching tree is available. The definition of our cuts relies on a positive scalar value, easily computable for several classes of problems. In particular, we are able to address quadratic convex integer biobjective problems as well as second order cone integer biobjective problems. As far as we know, the first general purpose method to tackle convex multiobjective integer programs is the heuristic algorithm proposed by Cacchiani and D'Ambrosio in [10]. In this respect, the Frontier Partitioner Algorithm gives a contribution in the context of exact methods for multiobjective nonlinear integer programs, defining a criterion space algorithm for nonlinear biobjective integer programs.

The paper is organized as follows. In Section 2, we give some basic definitions and concepts of multiobjective optimization, specifically adapted to biobjective integer optimization. We further report some assumptions we will need to define the algorithm as well as some solution techniques. In Section 3, we define the so called *custom weighted-sum scalarization* that will be used to improve the basic version of our algorithm. In Section 4, we introduce and analyze our algorithm, providing convergence analysis, examples and a discussion on the assumptions. In Section 5 we introduce suitable linear approximations of the nonlinear inequalities and we discuss how to deal with some numerical issues. Our numerical experience is presented in Section 6, where we compare our algorithm with the balanced box method proposed in [4] on a set of biobjective linear integer problems. We further report results on quadratic instances. In Section 7 we give some conclusions.
Notation Given a vector $x \in \mathbb{R}^n$, x_i is the $i-$th component and we use the sets $\mathbb{R}^n_{>0} = \{x \in \mathbb{R}^n : x_i > 0, i = 1, ..., n\}$, $\mathbb{R}^n_{\geq 0} = \{x \in \mathbb{R}^n : x_i \geq 0, i = 1, ..., n\}$ and $\mathbb{Z}^n_{\geq 0} = \{x \in \mathbb{Z}^n : x_i \geq 0, i = 1, ..., n\}$.

2. Preliminaries

We consider the biobjective integer programming BOIP, i.e. Problem (MOIP) with $p = 2$

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (y_1(x), y_2(x)), \tag{BOIP}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and the functions $y_1, y_2 : \mathbb{R}^n \to \mathbb{R}$ are continuous.

Definition 2.1. *A feasible solution $x \in \mathcal{X} \cap \mathbb{Z}^n$ is weakly efficient for Problem* (BOIP)*, if there is no feasible point $z \in \mathcal{X} \cap \mathbb{Z}^n$, such that $y_i(z) < y_i(x)$ for $i = 1, 2$. If $x \in \mathcal{X} \cap \mathbb{Z}^n$ is weakly efficient then $y(x)$ is called a weakly non-dominated point.*

Definition 2.2. *A feasible solution $x \in \mathcal{X} \cap \mathbb{Z}^n$ is efficient (or Pareto optimal) for Problem* (BOIP)*, if there is no feasible point $z \in \mathcal{X} \cap \mathbb{Z}^n$, such that $y_i(z) \leq y_i(x)$ for $i = 1, 2$ and $y(x) \neq y(z)$. If $x \in \mathcal{X} \cap \mathbb{Z}^n$ is efficient then $y(x)$ is called a non-dominated point. The set of all non-dominated points $\mathcal{Y}_N \subseteq \mathcal{Y}$ is called efficient frontier (or Pareto frontier).*

Definition 2.3. *The* ideal *objective vector of Problem* (BOIP) *is the vector $y^{id} \in \mathbb{R}^2$ defined component-wise as*

$$y_i^{id} = \min_{\mathcal{X} \cap \mathbb{Z}^n} y_i(x), \quad i = 1, 2. \tag{1}$$

2.1. Assumptions

Our goal is to design an algorithm able to produce the entire Pareto frontier $\mathcal{Y}_N$ of Problem (BOIP). In order to achieve our aim, we introduce a basic assumption on problem BOIP.

Assumption 2.4 (Existence of the ideal vector)**.** *We assume that the ideal objective values y_i^{id}, $i = 1, 2$ exist.*

In the definition of our algorithm we need to assume that a positive value exists that underestimates the distance between the image of two integer feasible points of (BOIP). We then consider the following class of functions.

Definition 2.5 (Positive γ-function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a positive γ-function over $\mathcal{X} \cap \mathbb{Z}^n$ if there exists a positive $\gamma \in \mathbb{R}$ such that $|f(x) - f(z)| \geq \gamma$, for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ with $f(x) \neq f(z)$.*

We will use the following assumption.

Assumption 2.6 (Positive gap value)**.** *The functions $y_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2$ in Problem* (BOIP) *are positive γ-function as in definition 2.5.*

Note that, if γ_i were zero for some $i \in \{1,2\}$ we would have that $y_i(x)$ is constant for all $x \in \mathcal{X} \cap \mathbb{Z}^n$. In Section 4.3 we will show that Assumption 2.6 is not restrictive and the algorithm can be applied to several classes of biobjective problems.
Under Assumption 2.6 and Assumption 2.4, we can prove that the Pareto frontier is finite which is a commonly used assumption when defining exact algorithms for MOIP (see, e.g., [30, 4, 3]). We need this result to prove the convergence of our Frontier Partitioner Algorithm in Section 4.1.

Proposition 2.7. *Let Assumptions 2.4 and 2.6 hold. Then, the Pareto frontier $\mathcal{Y}_N$ is a finite set.*

Proof. Under Assumption 2.4, there exist $\hat{x}^i \in \arg\min_{\mathcal{X}\cap\mathbb{Z}^n} y_i(x),\ i = 1,2$.
Hence there exist the values $M_1 = y_1(\hat{x}^2)$ and $M_2 = y_2(\hat{x}^1)$ such that the Pareto frontier $\mathcal{Y}_N \subseteq \mathcal{Y}$ is contained in the box $[y_1^{id}, M_1] \times [y_2^{id}, M_2]$ and hence it is bounded. Each objective function y_i can attain at most $\frac{M_i - y_i^{id}+1}{\gamma_i}$ distinct values, so that the cardinality of the Pareto frontier, obtained as the combination of the two, is finite and at most

$$\frac{(M_1 - y_1^{id} + 1)(M_2 - y_2^{id} + 1)}{\gamma_1\gamma_2}.$$

□

2.2. Solution techniques
To properly introduce our branching method we need to be able to get a Pareto point of a given (BOIP). To this aim, we can use any standard technique proposed in the literature. We refer the interested reader to [15] for a complete overview of solution techniques. In this section, we report only the two methods that we will use to define the new scalarization technique adopted in our algorithm, which are the weighted-sum method and the lexicographic optimization. Both these techniques lead to the solution of a single objective integer problem that we named as Inner Integer Program (IIP).
To prove convergence of our FPA algorithm we will need an assumption on the IIPs generated during the partitioning of the region (Assumption 4.1) which essentially implies that the IIPs cannot be unbounded below, so that either an optimal solution exists or infeasibility occurs. Hence in the following we show that this property holds for both the techniques.

Definition 2.8 (Weighted-sum IIP)**.** *Given (BOIP), the weighted-sum scalarization problem (IIP$_W$) is defined as*

$$\min_{x\in\mathcal{X}\cap\mathbb{Z}^n} \lambda_1 y_1(x) + \lambda_2 y_2(x), \tag{IIP$_W$}$$

with finite $\lambda_i \geq 0$, for $i = 1,2$.

For sake of simplicity it is not explicitly required that $\lambda_1 + \lambda_2 = 1$, since this property can be gained by simple normalization.
It is well known that under a proper choice of the weights the optimal solution of the scalarized problem (IIP$_W$) leads to an efficient solution as reported in the following proposition.

Proposition 2.9 (Proposition 3.9 [15]). *Let $\lambda_1, \lambda_2 > 0$, then each optimal solution of Problem* (IIP_W) *is an efficient solution for Problem* (BOIP).

We observe that the converse is true only under proper convexity assumptions. Since integer multiobjective optimization problems are non-convex we cannot expect to find all efficient solutions by weighted sum scalarization as pointed out in Chapter 8 of [15]. Nevertheless as we mentioned above, we only need to ensure that (IIP_W) is bounded and this easily follows from Assumption 2.4 as reported in Remark 2.10 below.

Remark 2.10. *Under Assumption 2.4 problem (IIP_W) is bounded. Indeed, assume by contradiction that (IIP_W) is unbounded. Then, for all $M > 0$ there exists $x \in \mathcal{X} \cap \mathbb{Z}^n$ such that $\lambda_1 y_1(x) + \lambda_2 y_2(x) < -M$. We get a contradiction from Assumption 2.4, as $y_1(x) \geq y_1^{id}$, $y_2(x) \geq y_2^{id}$ and $\lambda_1, \lambda_2 \geq 0$ are finite.*

In the following we define two further problems that will be used in order to define an improved version of our algorithm. The first definition is based on the concept of lexicographic optimality introduced in e.g. [15].

Definition 2.11 (Lexicographic IIP). *Given (BOIP) and a permutation (i_1, i_2) of the set $\{1, 2\}$, the **lexicographic problem** is defined as*

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \left\{ y_{i_1}(x) : y_{i_2}(x) = y_{i_2}^{id} \right\},$$

where $y_{i_2}^{id} = \min_{x \in \mathcal{X} \cap \mathbb{Z}^n} y_{i_2}(x)$.

In the following we denote the lexicographic problem as:

$$\text{lex} \min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (y_{i_1}(x), y_{i_2}(x)) \tag{IIP_L}$$

We note that tackling Problem (IIP_L) requires the solution of two single-objective mixed integer programming problems.
A well known result is the following.

Proposition 2.12 (Lemma 5.2 [15]). *Each optimal solution of Problem* (IIP_L) *is an efficient solution for Problem* (BOIP).

Also for the (IIP_L) we can prove boundedness under Assumption 2.4 as explained below in Remark 2.13.

Remark 2.13. *Under Assumption 2.4 problem (IIP_L) is bounded. In fact, by Assumption 2.4 we know that there exists finite $y_i^{id} = \min_{x \in \mathcal{X} \cap \mathbb{Z}^n} y_i(x)$, for each $i \in \{1, 2\}$. This implies that $y_{i_1}^{id} \leq \min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \left\{ y_{i_1}(x) : y_{i_2}(x) = y_{i_2}^{id} \right\}$.*

3. Custom weighted-sum scalarization

In this section, we define a new scalarized problem which smartly combines the weighted-sum and the lexicographic techniques.

Definition 3.1 (Custom weighted-sum IIP)**.** *Let $\hat{x} \in \mathcal{X} \cap \mathbb{Z}^n$ be any optimal solution of Problem* (IIP$_L$) *with respect to the permutation (i_1, i_2) of the set $\{1, 2\}$.*
The ***custom weighted-sum scalarization problem*** *is defined as*

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda_1^* y_1(x) + \lambda_2^* y_2(x), \tag{IIP*}$$

where $\Lambda^* = (\lambda_1^*, \lambda_2^*)$ belongs to the set $\{(\lambda_1, \lambda_2) \in \mathbb{R}^2_{>0} : \hat{x} \in \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda_1 y_1(x) + \lambda_2 y_2(x)\}$.

For sake of simplicity it is not explicitly required that $\lambda_1 + \lambda_2 = 1$, since this property can be gained by simple normalization.
The underlying idea of the custom weighted-sum scalarization (IIP*) is to reduce the computational complexity of the lexicographic optimization when dealing with a sequence of nested IIPs. Indeed if λ^* are known, an optimal solution of (IIP*) can be obtained by solving only one ILP instead of two as needed by (IIP$_L$). As it will be clarified later, the key tool in algorithm FPA* is to find suitable weights λ^* only once, since it will be shown that they remain valid also for the subproblems.
Refer now to (IIP) as any problem chosen among (IIP$_W$), (IIP$_L$) and (IIP*). Under Assumptions 2.4 and 2.6 by Proposition 2.7 the Pareto frontier is finite. Using Remarks 2.10 and 2.13, we have that any of the (IIP) problems either has an optimal solution which is an efficient solution for problem (BOIP), or it is infeasible and hence (BOIP) is too. This property turns out to be crucial when proving well-posedness and convergence of `FPA` (see the proof of Theorem 4.7).
In the definition of our algorithm, we can use also other solution techniques which guarantee that the inner integer problem either has an optimal solution or is infeasible. In particular compromise programming with a norm ℓ_p with $1 \leq p < \infty$ (see e.g. Chapter 4 of [15]) can also fit in this setting. However, the techniques proposed above present the advantage that the corresponding inner integer problem belongs to the same class of the original (BOIP). In other words, if BOIP has linear objective functions the inner integer problem is an ILP, if BOIP has quadratic objectives the inner integer problem is an IQP and more generally the objective function of the inner integer problem maintains the structure of the original ones in (BOIP).

In the following, we prove that when both Assumption 2.4 and Assumption 2.6 are satisfied, the custom weighted-sum problem (IIP*) can always be defined, and the values of the weights are easily computable. From an algorithmic point of view, we will see that the existence of such weights allows us to take advantage of the properties of both weighted-sum and lexicographic method, improving significantly the complexity of our algorithmic scheme.

Let us now describe a way to derive suitable weights for (IIP*). Given a permutation (i_1, i_2) of the set $\{1, 2\}$ let

$$\mathcal{P} = \arg\operatorname{lex}\min_{x\in\mathcal{X}\cap\mathbb{Z}^n}(y_{i_1}(x), y_{i_2}(x))$$

Let $\hat{x} \in \mathcal{P}$ and $y_i(\hat{x}) = \hat{y}_i$. For each $x \in \mathcal{P}$ we have that $y_i(x) = \hat{y}_i$. Further consider the reverse lexicographic optimization and let

$$\bar{x} = \arg\operatorname{lex}\min_{x\in\mathcal{X}\cap\mathbb{Z}^n}(y_{i_2}(x), y_{i_1}(x))$$

and $\bar{y} = y(\bar{x})$. By Proposition 2.9 both $\bar{y}$ and $\hat{y}$ are non-dominated points of (BOIP) and we can assume that $\hat{y} \neq \bar{y}$ as otherwise the ideal vector would represent the only optimal solution to the problem.
Let $\gamma = \min_{i=1,2}\{\gamma_i\} > 0$. For any $\zeta \in (0, \gamma)$ we define $\lambda(\zeta) \in \mathbb{R}^2_{>0}$ as

$$\lambda(\zeta)_i = \begin{cases} \dfrac{\gamma - \zeta}{\hat{y}_{i_1} - \bar{y}_{i_1}}, & \text{if } i = i_1 \\ 1, & \text{if } i = i_2 \end{cases} \tag{2}$$

We now show that $\lambda(\zeta) \in \Lambda^*$ for any $\zeta \in (0, \gamma)$, hence that this choice of weights for (IIP*) allows to prove correspondence with the lexicographic optimization.

Theorem 3.2. *Given (BOIP), let Assumption 2.4 and Assumption 2.6 hold. Given a permutation (i_1, i_2) of the set $\{1, 2\}$ there exist weights $\lambda^* \in \mathbb{R}^2_{>0}$ such that $\lambda^*_{i_1} \leq \lambda^*_{i_2}$ and*

$$\arg\operatorname{lex}\min_{x\in\mathcal{X}\cap\mathbb{Z}^n}(y_{i_1}(x), y_{i_2}(x)) \;=\; \arg\min_{x\in\mathcal{X}\cap\mathbb{Z}^n}\lambda^{*\intercal}y(x)$$

Proof. Without loss of generality we fix $i_1 = 1$ and $i_2 = 2$. Define $\hat{y}$, $\bar{y}$ and for a specific value of $\zeta \in (0, \gamma)$ define $\lambda^* = \lambda(\zeta) \in \mathbb{R}^2_{>0}$ as in (2) so that $\lambda^*_{i_1} \leq \lambda^*_{i_2}$. Let $x^* \in \arg\min_{x\in\mathcal{X}\cap\mathbb{Z}^n}\lambda^{*\intercal}y(x)$. Since $\lambda^* \in \mathbb{R}^2_{>0}$ by Proposition 2.9 $y(x^*) = y^*$ is a non-dominated point. Further we have that $\lambda^{*\intercal}y(x^*) \leq \lambda^{*\intercal}y(x) \;\; \forall\, x \in \mathcal{X} \cap \mathbb{Z}^n$, and, in particular, $\lambda^{*\intercal}(\hat{y} - y^*) \geq 0$ and from the definition of λ^* we get

$$\frac{\gamma - \zeta}{\hat{y}_1 - \bar{y}_1}(\hat{y}_1 - y^*_1) + \hat{y}_2 - y^*_2 \geq 0. \tag{3}$$

Taken into account that both $\hat{y}$ and y^* are non-dominated points for (BOIP) and that $\hat{y}$ is in $\mathcal{P}$ only one of the following situations may occur

(i) $y^*_1 < \hat{y}_1$ and $y^*_2 > \hat{y}_2$

(ii) $y^*_1 = \hat{y}_1$ and $y^*_2 = \hat{y}_2$

We show that only case (ii) holds and this implies the theorem. Assume by contradiction that (i) holds. Then, we would have $\bar{y}_1 \leq y^*_1 < \hat{y}_1$ implying

$$\hat{y}_1 - \bar{y}_1 \geq \hat{y}_1 - y^*_1. \tag{4}$$

Using (4) within (3) we get $\gamma - \zeta + \hat{y}_2 - y_2^* \geq 0$. Since $y_2^* - \hat{y}_2 \geq \gamma$ we have $\gamma - \zeta \geq \gamma$ and hence $\zeta \leq 0$ which leads to a contradiction. ∎

In the proof of the theorem above we consider the special choice of weights given by (2). We observe that given $\lambda \in \Lambda^*$ the vector $\alpha\lambda$ trivially belongs to Λ^* for any $\alpha > 0$. In the next theorem, we show that we can scale only one of the two components of the vector λ remaining within the set Λ^*.

Proposition 3.3. *Given (BOIP), let Assumption 2.4 and Assumption 2.6 hold. Given a permutation (i_1, i_2) of the set $\{1, 2\}$ and given weights $\lambda^* \in \Lambda^*$ and*

$$\arg\operatorname{lex}\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} (y_{i_1}(x), y_{i_2}(x)) \;=\; \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda^{*\intercal} y(x) \tag{5}$$

Then the vector weights $\tilde{\lambda} = (\alpha\lambda_1^*, \lambda_2^*)$ satisfy (5) for every $\alpha \in (0, 1]$.

Proof. Without loss of generality we set $i_1 = 1$ and $i_2 = 2$. Since we know from Theorem 3.2 that $\lambda_1^* \leq \lambda_2^*$ we define $\lambda^S = \frac{1}{\lambda_2^*}\lambda^* = \left(\lambda_1^S, 1\right)^\intercal \in \Lambda^*$ and $\lambda_1^S \leq 1$. Let

$$x^* \in \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda^{*\intercal} y(x)$$

and $\tilde{\lambda} = \left(\tilde{\lambda}_1, 1\right)^\intercal$. We show that any vector $\tilde{\lambda}$ with $\tilde{\lambda}_1 \in (0, \lambda_1^S]$ belongs to Λ^* so that

$$\arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda^{S\intercal} y(x) \;=\; \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \tilde{\lambda}^\intercal y(x)$$

By contradiction suppose that there exists a point $\tilde{x} \in \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \tilde{\lambda}^\intercal y(x)$ and $\tilde{x} \notin \arg\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \lambda^{S\intercal} y(x)$. We have that $\tilde{\lambda}^\intercal y(\tilde{x}) \leq \tilde{\lambda}^\intercal y(x^*)$ which implies

$$y_2(x^*) - y_2(\tilde{x}) \geq \tilde{\lambda}_1 \left(y_1(\tilde{x}) - y_1(x^*)\right) \tag{6}$$

By the definition of x^* we know that $\lambda^{S\intercal} y(x^*) < \lambda^{S\intercal} y(\tilde{x})$ and therefore

$$y_2(x^*) - y_2(\tilde{x}) < \lambda_1^S \left(y_1(\tilde{x}) - y_1(x^*)\right) \tag{7}$$

Combining (6) and (7) we get $y_1(\tilde{x}) > y_1(x^*)$. Since $\tilde{x}$ is an efficient point by Proposition 2.9 and x^* is a lexicographic efficient point by Proposition 2.12, we have that it should hold that $y_2(\tilde{x}) < y_2(x^*) = y_2^{id}$. We then get a contradiction. ∎

We now show that given a subset $\mathcal{H}$ of $\mathcal{X} \cap \mathbb{Z}^n$ the vector of weights λ^* can be used to define the custom weighted-sum associated with the set $\mathcal{H}$. This is needed in our branching scheme to properly define the (IIP) at each node.

Proposition 3.4 (λ-Inheritance)**.** *Let Assumption 2.4 and Assumption 2.6 hold. Given a permutation (i_1, i_2) of the set $\{1, 2\}$ and a value $\zeta \in (0, \gamma)$, let $\lambda^* = \lambda(\zeta)$ as in (2). Then for any subset $\mathcal{H}$ of $\mathcal{X} \cap \mathbb{Z}^n$ such that the Pareto frontier of $\min_{x \in \mathcal{H}} (y_{i_1}(x), y_{i_2}(x)) \subseteq \mathcal{Y}_N$, we have that the weight vector λ^* satisfies*

$$\arg\operatorname{lex}\min_{x \in \mathcal{H}} (y_{i_1}(x), y_{i_2}(x)) \;=\; \arg\min_{x \in \mathcal{H}} \lambda^{*\intercal} y(x)$$

Proof. Let $\lambda^{\mathcal{H}}$ be the weights obtained from (2) where $\hat{y}^{\mathcal{H}}$ and $\bar{y}^{\mathcal{H}}$ are the vectors $\hat{y}$ and $\bar{y}$ obtained subject to the constraint $x \in \mathcal{H}$. Since Assumption 2.4 and Assumption 2.6 hold these weights are such that

$$\arg\operatorname{lex}\min_{x\in\mathcal{H}}(y_{i_1}(x), y_{i_2}(x)) \;=\; \arg\min_{x\in\mathcal{H}} \lambda^{\mathcal{H}\intercal} y(x)$$

By assumption, we have that $\min_{x\in H}(y_{i_1}(x), y_{i_2}(x)) \subseteq \mathcal{Y}_N$. This implies that $\hat{y}^{\mathcal{H}}_{i_1} \leq \hat{y}_{i_1}$ and $\bar{y}^{\mathcal{H}}_{i_1} \geq \bar{y}_{i_1}$, so that $\lambda^{\mathcal{H}}_{i_1} \geq \lambda^*_{i_1}$. Since $\lambda^{\mathcal{H}}_{i_2} = \lambda^*_{i_2} = 1$, the assumptions of Proposition 3.3 are satisfied. Therefore

$$\arg\operatorname{lex}\min_{x\in\mathcal{H}}(y_{i_1}(x), y_{i_2}(x)) \;=\; \arg\min_{x\in\mathcal{H}} \lambda^{*\intercal} y(x)$$

□

4. The Frontier Partitioner Algorithm

In this section, we introduce the Frontier Partitioner Algorithm `FPA`. Convergence and finiteness of the algorithm are analyzed in Section 4.1.
The `FPA` uses a *divide and conquer* approach to explore the Pareto frontier of (BOIP). Starting from a non-dominated solution the method builds two subproblems in such a way that the chosen non-dominated solution and all the points dominated by it are infeasible for both the subproblems. Hence the key ingredients of `FPA` are

- the construction of subproblems using properly defined inequalities,
- the computation of non-dominated solutions at each node of the branching tree.

At a generic node k in the branching tree the subproblem $(\text{BOIP})^k$

$$\min_{x\in\mathcal{X}^k\cap\mathbb{Z}^n} y(x) \tag{BOIP^k}$$

is constructed, where $\mathcal{X}^k \subseteq \mathcal{X}$ is the set obtained intersecting $\mathcal{X}$ with properly defined inequalities. For $k = 0$, i.e. at the root node, we define (BOIP^0)=(BOIP) and $\mathcal{X}^0 = \mathcal{X}$. For $k > 0$ the definition $\mathcal{X}^k$ is clarified below.
In order to compute a non-dominated solution of $(\text{BOIP})^k$ we can use any techniques proposed in Sections 2.2 and 3 to get $(\text{IIP})^k$. We need the following assumption on the inner integer problem $(\text{IIP})^k$.

Assumption 4.1 (Solvability of the inner integer problem)**.** *There exists an oracle that either returns an optimal solution of (IIP) or certifies the infeasibility of problem $(IIP)^k$.*

From the point of view of implementation using an oracle means calling a solver suitable for problem $(\text{IIP})^k$. Depending on the choice of the solution techniques used, either (IIP_W), (IIP^*) or (IIP_L), the call of an oracle may require the solution of either one or two integer problems.

In case (IIP)^k has an optimal solution, two children nodes in the branching tree are produced. Let $\hat{x}^k \in \mathcal{X}^k \cap \mathbb{Z}^n$ be an optimal solution of $(IIP)^k$ and $\hat{y}^k = y(\hat{x}^k)$.

Let $\epsilon_i \in (0, \gamma_i]$, $i = 1, 2$ where γ_i satisfies 2.6. We consider the inequalities

$$y_i(x) \leq \hat{y}_i^k - \epsilon_i, \ i = 1, 2. \tag{8}$$

Remark 4.2. *The inequalities $y_i(x) \leq \hat{y}_i^k - \epsilon_i, \ i = 1, 2$ are violated by the non-dominated solution $\hat{y}^k$. Furthermore, they are linear in the criterion space and they are convex (linear) in the decision space as long as the functions $y_i(x), \ i = 1, 2$ are convex (linear).*

From $(BOIP)^k$, using inequalities (8), we define the two children nodes of node k as follows:

$$\min_{x \in \mathcal{X}_1^k \cap \mathbb{Z}^n} y(x) \qquad \mathcal{X}_1^k = \mathcal{X}^k \cap \{x \in \mathbb{R}^n \ : \ y_1(x) \leq \hat{y}_1^k - \epsilon_1\},$$

$$\min_{x \in \mathcal{X}_2^k \cap \mathbb{Z}^n} y(x) \qquad \mathcal{X}_2^k = \mathcal{X}^k \cap \{x \in \mathbb{R}^n \ : \ y_2(x) \leq \hat{y}_2^k - \epsilon_2\}.$$

The Frontier Partitioner Algorithm produces iteratively a finite list of BOIPs.

Remark 4.3. *At a generic node $k > 0$ of the branching tree, the feasible region $\mathcal{X}^k$ is obtained from the original $\mathcal{X}$ by adding at most two inequalities. Each inequality takes the form $y_i(x) \leq const_i$ with $i = 1, 2$, where $const_i = \min_{0 \leq j \leq k} \hat{y}_i^j - \epsilon_i$. Hence, letting m be the number of constraints defining $\mathcal{X}$, the number of constraints of $\mathcal{X}^k$ is at most $m + 2$ for all k, as among the k new constraints introduced there will be at most two non-redundant ones.*

The scheme of the Basic Frontier Partitioner Algorithm `FPA` is reported in Algorithm 1. The algorithm generates a list $\mathcal{L}$ of BOIPs by starting from the original $(BOIP)^0$. For each selected $(BOIP)^k$ from the list, the corresponding $(IIP)^k$ is constructed and solved by an oracle call. Using the optimal solution, if any, two new BOIPs are produced and added to the list. The algorithm stops when the list is empty.

Algorithm 1: FPA scheme

Input: $\mathcal{L} = \{(BOIP)^0\}$, $\mathcal{X}^0 = \mathcal{X}$, $\mathcal{Y}_N = \emptyset$, $\gamma_i > 0$, $\epsilon_i \in (0, \gamma_i], i = 1, 2$
Output: the Pareto frontier $\mathcal{Y}_N$ of $(BOIP)$
while $\mathcal{L} \neq \emptyset$ **do**
  Select a node $(BOIP)^k \in \mathcal{L}$ and delete it from $\mathcal{L}$
  Derive $(IIP)^k$ from $(BOIP)^k$
  Call an oracle on $(IIP)^k$.
  if $(IIP)^k$ *has an optimal solution* $\hat{x}^k$ **then**
    Set $\mathcal{Y}_N = \mathcal{Y}_N \cup \{\hat{y}^k\}$, where $\hat{y}^k = y(\hat{x}^k)$
    Build $(BOIP)^k_i$, $i = 1, 2$ from $(BOIP)^k$
      $(BOIP)^k_i := \min\left\{y(x) : x \in \mathcal{X}^k_i \cap \mathbb{Z}^n\right\}$
      Where $\mathcal{X}^k_i := \mathcal{X}^k \cap \{x \in \mathbb{R}^n : y_i(x) \leq \hat{y}^k_i - \epsilon_i\}$, $i = 1, 2$,
    Add the new nodes $(BOIP)^k_1$ and $(BOIP)^k_2$ to $\mathcal{L}$;
  end
end
Return $\mathcal{Y}_N$

We prove in Section 4.1 that under suitable assumptions `FPA` is well posed and terminates finitely, returning the entire Pareto frontier $\mathcal{Y}_N$.
The BOIPs generated are related to those that would be built applying the approach proposed in [11, 21, 13].
In particular in [11] the BOIPs are generated by using the weighted-sum scalarization method and the existence of an oracle for solving BOIP is assumed. In [21] the authors present a method which identifies a region where it is possible to find further non-dominated points. The region is updated iteratively each time a new non-dominated point is found. To this aim they construct a list $\mathcal{U}$ of *local upper bounds* and keep it updated according to the new non-dominated points found. It is mentioned how to use this list of local upper bounds in order to define an algorithm for multiobjective combinatorial optimization. At each iteration k, a local upper bound $u^k \in \mathcal{U}$ is selected and the subproblem $P(u^k)$ is built by adding to the feasible region $X \cap \mathbb{Z}^n$ the constraints $y(x) < u^k$. Problem $P(u^k)$ is solved, the list $\mathcal{U}$ is updated and a new efficient point is eventually found. In case $p = 2$, the inequalities $y(x) < u^k$ are strongly related to the inequalities introduced in (8), as $u^k \in \mathcal{U}$ is built by using components of non-dominated points found so far, so that they read as $y_i(x) < \hat{y}^j_i$ for some $j = 1, \ldots, k$ and for all $i = 1, 2$. The number of integer problems needed to be solved by this approach is $2|\mathcal{Y}_N| + 1$.
We underline that `FPA` does not need to construct the list $\mathcal{U}$ and hence it does not need any algorithmic procedure to keep $\mathcal{U}$ updated and filter dominated local upper bounds as needed in [21, 13]. Furthermore, in [21] minor details on how to solve the integer subproblem $P(u^k)$ are given. Indeed, handling strict inequalities is not allowed when using standard software such as e.g. CPLEX [19], Gurobi [18], SCIP [17], Couenne [1] or Bonmin [8].

In this respect, the definition of γ_i and of $\epsilon_i \in (0, \gamma_i]$, $i = 1, 2$ in (8) is crucial to obtain inner integer problems which satisfy Assumption 4.1, namely for which an oracle exists using standard available softwares. The main contribution of our paper stays in embedding the new custom weighted-sum scalarization procedure within the divide-and-conquer procedure. The use of the new scalarization allows to prove that exactly $|\mathcal{Y}_N| + 1$ nodes are generated as reported in the next Section.

4.1. Convergence Analysis

As a first step in the convergence analysis of the Frontier Partitioner Algorithm, we prove that the cuts used in Algorithm 1 induce a partition of the decision space.

Proposition 4.4. *Let Assumption 2.6 holds. Then $\mathcal{X}_1^k \cap \mathcal{X}_2^k \cap \mathbb{Z}^n = \emptyset$ for all k in Algorithm 1.*

Proof. Let $\hat{x}^k \in \mathcal{X}^k \cap \mathbb{Z}^n$ be an efficient point for $(BOIP)^k$ corresponding to the non-dominated value $\hat{y}^k$. Assume by contradiction that $\mathcal{X}_1^k \cap \mathcal{X}_2^k \cap \mathbb{Z}^n \neq \emptyset$. Then $x \in \mathcal{X}_1^k \cap \mathcal{X}_2^k \cap \mathbb{Z}^n$ exists and satisfies $y_i(x) < y_i(\hat{x}_k)$, for $i = 1, 2$ as $\epsilon_i > 0$ by assumption. This contradicts the fact that $\hat{x}^k$ is an efficient solution for $(BOIP)^k$. ∎

Remark 4.5. *Under Assumption 2.6, we have that $y(x) \neq \hat{y}^k$ for any $x \in \mathcal{X}_1^k \cap \mathbb{Z}^n$ and any $x \in \mathcal{X}_2^k \cap \mathbb{Z}^n$. Therefore, the inequalities used to define $\mathcal{X}_i^k$, $i = 1, 2$ exclude all the efficient solutions $\hat{x}$ such that $\hat{y}^k = y(\hat{x})$.*
Since $|y_i(x) - \hat{y}_i^k| \geq \gamma_i$ for $i = 1, 2$, we have that

$$\mathcal{Y}_N^k \;=\; \{\hat{y}^k\} \;\cup\; \mathcal{Y}_N^{k,1} \;\cup\; \mathcal{Y}_N^{k,2}$$

where $\mathcal{Y}_N^{k,i}$, $i = 1, 2$ denote the Pareto frontier of the children nodes of $(BOIP)^k$.
Therefore, the Pareto frontier is recursively obtained as $\mathcal{Y}_N = \{\hat{y}^0\} \cup \mathcal{Y}_N^{0,1} \cup \mathcal{Y}_N^{0,2}$.

In the following proposition, we prove that each node tackled in the Algorithm 1 can be pruned or produces a not yet known Pareto point .

Proposition 4.6. *Let $\hat{y} \in \mathcal{Y}_N^k$ be a non-dominated point of $(BOIP)^k$. Then, the child problem $(BOIP)_i^k$*

$$\begin{array}{ll} \min & y(x) \\ \text{s.t.} & x \in \mathcal{X}^k \cap \{y_i(x) \leq \hat{y}_i - \epsilon_i\} \\ & x \in \mathbb{Z}^n \end{array} \tag{9}$$

with $i = 1, 2$ is either infeasible or any of its optimal solutions is efficient for $(BOIP)^k$, leading to a new non-dominated point $\bar{y} \neq \hat{y}$.

Proof. If problem (9) is infeasible there is nothing to prove. W.l.o.g. let $i = 1$ and let $\bar{x}$ be a solution of (9) (case $i = 2$ can be proven identically). By contradiction, assume that $\bar{x}$ is not efficient for (BOIP)^k. Then, $\tilde{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ exists such that $y_i(\tilde{x}) \leq y_i(\bar{x})$, for $i = 1, 2$ and $y(\tilde{x}) \neq y(\bar{x})$. In particular, we have that

$$y_1(\tilde{x}) \leq y_1(\bar{x}) \leq \hat{y}_1 - \epsilon_1$$

so that $\tilde{x}$ is feasible for (9). Since $y(\tilde{x}) \neq y(\bar{x})$, we necessarily have that either $y_1(\tilde{x}) < y_1(\bar{x})$ or $y_2(\tilde{x}) < y_2(\bar{x})$, contradicting the fact that $\bar{x}$ is efficient for (9). Furthermore, we have that $y(x) \neq \hat{y}$ for any point x feasible for (9) as stated in Remark 4.5, so that $\bar{x}$ leads to a non-dominated point $y(\bar{x}) = \bar{y} \neq \hat{y}$. □

Now we are ready to prove the finite convergence of Algorithm 1.

Theorem 4.7. *Let Assumptions 2.4, 2.6 and 4.1 hold. Algorithm 1 returns the complete Pareto frontier $\mathcal{Y}_N$ of* (BOIP) *after generating exactly $2|\mathcal{Y}_N| + 1$ $(BOIP)^k$.*

Proof. At each iteration k of the `FPA` a node $(BOIP)^k$ is chosen and the corresponding $(IIP)^k$ is built.
Assumption 4.1 allows us to solve Problem $(IIP)^k$. Using Remark 2.10, we have that either $(IIP)^k$ has an optimal solution or is infeasible. If $(IIP)^k$ is infeasible, we conclude that $\mathcal{X}^k \cap \mathbb{Z}^n$ does not contain any efficient point and the node $(BOIP)^k$ is pruned. Otherwise, we have that the returned optimal solution of $(IIP)^k$ is efficient for $(BOIP)^k$, giving us a non-dominated point $\hat{y}^k$. Using Proposition 4.6, we have that $\hat{y}^k$ belongs to the Pareto frontier of (BOIP). By Proposition 4.4 and Remark 4.5, the non-dominated point $\hat{y}^k \in \mathcal{Y}_N$ cannot be detected again by addressing any subsequent node in the branching tree and the inequalities induced by $\hat{y}^k$ do not cut any yet unknown Pareto point. Summarizing, whenever a node is addressed, either we get a new non-dominated point or we detect infeasibility and the node is pruned. Therefore since `FPA` produces exactly two children for each non-dominated point of (BOIP), we have that the branching tree has exactly $2|\mathcal{Y}_N| + 1$ nodes (including the root node) so that exactly $2|\mathcal{Y}_N| + 1$ $(\text{BOIP})^k$ are generated. □

We now show that the use of different solution techniques may lead to a different number of solutions of inner integer programs (IIP), namely to a different number of oracle calls, which represent the main computational burden. In particular, we will show that the use of the custom weighted-sum allows us to define an improved version of `FPA`, called `FPA`*, able to detect the complete Pareto frontier after having solved only $|\mathcal{Y}_N| + 2$ integer programs.
Before describing the special case of the FPA with the custom weighted-sum scalarization, we report the results that follows directly from Theorem 4.7. In particular, for the FPA with (IIP) obtained by the weighted-sum scalarization technique, we have the following results that have already been proved in [11].

Corollary 4.8. *If $(IIP_W)^k$ as defined in* (2.9) *with strictly positive weights $\lambda_i > 0$ is used to define the scalarization program then Algorithm 1 returns the complete Pareto after having solved $2|\mathcal{Y}_N| + 1$ IIPs of which $|\mathcal{Y}_N| + 1$ must have an empty feasible set and after having called the oracle $2|\mathcal{Y}_N| + 1$ times.*

Proof. The proof is a straightforward consequence of the construction of the (IIP_W) in the branching tree. □

When using (IIP_L) or (IIP^*) as inner integer problems we obtain a drastic reduction of the number of $(\text{BOIP})^k$ tackled by the algorithm.

Corollary 4.9. *If either (IIP$_L$) or (IIP*) is used to define the solution technique, then at least one of the two subproblems (BOIP$_i^k$), $i = 1, 2$ has empty feasible set, so that the number of generated BOIPs is $|\mathcal{Y}_N| + 1$, where only one problem is empty.*

Proof. Every time a subproblem (IIP)^k is solved and a new Pareto point is found, the algorithm creates two subproblems $(BOIP)_i^k$, $i = 1, 2$, from $(BOIP)^k$ according the formulas $(BOIP)_i^k := \min\left\{y(x) : x \in \mathcal{X}_i^k \cap \mathbb{Z}^n\right\}$ and $\mathcal{X}_i^k := \mathcal{X}^k \cap \{x \in \mathbb{R}^n : y_i(x) \leq \hat{y}_i^k - \epsilon_i\}$ $i = 1, 2$. By the definition of lexicographic problem we have that $\mathcal{X}_{i_2}^k = \emptyset$, where (i_1, i_2) is the permutation of set $\{1, 2\}$ adopted by (IIP)^k. As a consequence we can fathom one subproblem at every branching, so that the algorithm produces $|\mathcal{Y}_N| + 1$ problems (BOIP)^k. Note that the last subproblem has empty feasible set but it cannot a priori fathomed. ∎

Remark 4.10. *If the lexicographic problem (IIP$_L$) is used in* `FPA`*, then Algorithm 1 returns the complete Pareto frontier after $|\mathcal{Y}_N|+1$ oracle calls but it requires the solution of $2|\mathcal{Y}_N| + 1$ single objective problems, with only one of them being infeasible.*
On the other hand when (IIP*) is used in `FPA`*, exactly $|\mathcal{Y}_N|+1$ single objective problems must be solved.*

In the following, we present the `FPA`*, which is a particular version of `FPA` where (IIP*) is used as scalarization technique, which allows reducing the number of integer programs that need to be solved.
The scheme of the `FPA`* is reported in Algorithm 2. As a first step the computation of the custom weights is considered. Thanks to Proposition 3.3 we can use any value $\alpha\lambda_1(\zeta)$ where $\lambda(\zeta)$ is defined as in (2) and $\alpha \in (0, 1)$. We propose the choice of λ^* in the `FPA`* satisfying $\lambda_1^* < \lambda_1(\zeta)$ for fixed values of ζ. Further Proposition 3.4 allows to use the same weights λ^* in all the nodes (BOIP)^k and we exploit this property. In the main loop, there is no need in constructing a list of open problems because the BOIPs are generated one at the time by using the solution of the preceding program (IIP*)^k.

Algorithm 2: FPA*

Input: $(BOIP^0) = (BOIP)$, $\mathcal{X}^0 = \mathcal{X}$, $\mathcal{Y}_N = \emptyset$,
$\gamma_i > 0$, $\epsilon_i \in (0, \gamma_i]$, $\zeta_i \in (0, \gamma_i), i = 1, 2$
$\{i_1, i_2\}$ permutation of $\{1, 2\}$, $k = 0$
Output: the Pareto frontier $\mathcal{Y}_N$ of $(BOIP)$
Compute $\hat{x}^j = \arg\min_{\mathcal{X} \cap \mathbb{Z}^n} y_j(x)$, $j = i_1, i_2$,
Evaluate the custom weights as

$$\lambda_i^* = \begin{cases} \dfrac{\gamma_{i_1} - \zeta_{i_1}}{y_{i_1}(\hat{x}^{i_2}) - y_{i_1}(\hat{x}^{i_1})}, & \text{if } i = i_1 \\ 1, & \text{if } i = i_2 \end{cases}$$

while $\left(y_{i_1}(x^k) > y_{i_1}^{id}\right)$ **do**
  Derive $(IIP^*)^k$ from $(BOIP)^k$
  Solve $(IIP^*)^k$ and let $\hat{x}^k$ be an optimal solution
  Set $\mathcal{Y}_N = \mathcal{Y}_N \cup \{\hat{y}^k\}$, where $\hat{y}^k = y(\hat{x}^k)$
  Build $(BOIP)^{k+1}$ from $(BOIP)^k$
    $(BOIP)^{k+1} := \min\left\{y(x) : x \in \mathcal{X}^{k+1} \cap \mathbb{Z}^n\right\}$
    Where $\mathcal{X}^{k+1} := \mathcal{X}^k \cap \{x \in \mathbb{R}^n : y_{i_1}(x) \leq \hat{y}_{i_1}^k - \epsilon_i\}$,
  Set $k = k + 1$;
end
Return $\mathcal{Y}_N$

The choice of unique value for λ^* outside the main loop allows us to improve the bound on the number single-objective integer problems solved as proved in the next theorem.

Theorem 4.11. *Let Assumptions 2.4, 2.6 and 4.1 hold. Algorithm 2 returns the complete Pareto frontier $\mathcal{Y}_N$ of* (BOIP) *after having solved $|\mathcal{Y}_N| + 2$ single-objective integer programs.*

Proof. Algorithm 2 is exactly FPA customized over (IIP*). By Corollary 4.9 if we use FPA we will solve $|\mathcal{Y}_N| + 1$ (IIP*)s, where exactly one has empty feasible set. By using FPA* we solve the problems sequentially by changing the level of the i_1-th function. Since in the first step of the algorithm we derive the ideal vector, we know that the i_1-th function will end up at that value, in other words the last point found by the algorithm will be the extreme point with the lowest value of the i_1-th function. Since in the *while-loop* a stopping criterion on the i_1-th function is introduced, the algorithm will stop before tackling an empty subproblem. Under these considerations FPA* will stop after having solved $|\mathcal{Y}_N|$ (IIP*)s.

As λ^* is computed outside the while loop, every time we solve the program $(\text{IIP}^*)^k$ in FPA* we have to deal with only one integer problem. According to Propositions 3.3 and 3.4, if we use λ^* as defined in FPA*, we need only to solve two integer problems to

find the custom weights valid for all the subproblems. In the end, the algorithm will need to solve exactly $|\mathcal{Y}_N| + 2$ single-objective integer programs. □

4.2. Toy example

In this section we show the behavior of algorithm FPA* on the following simple example proposed in [15] (example 8.6)

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^2} (x_1, x_2), \tag{10}$$

where $\mathcal{X}$ is the polyhedral set defined as

$$\mathcal{X} = \{x \in \mathbb{R}^2_{\geq 0} : 2x_1 + 3x_2 \geq 11,\ x_1 \leq 4,\ x_2 \leq 4\}.$$

The criterion space $\mathcal{Y}$ of Problem (10) (which is the same as the decision space) is represented in Figure 1a. The Pareto frontier is $\mathcal{Y}_N = \{(0,4);\ (1,3);\ (3,2);\ (4,1)\}$. As pointed out in [15] there is no setting of the weights $\lambda \in \mathbb{R}^2$ such that the point $(3,2)$ can be obtained as an optimal solution of a weighted-sum problem. We show that our algorithm FPA* is able to detect the full Pareto frontier $\mathcal{Y}_N$.

For this instance we have $\gamma_i = 1$ for $i = 1, 2$ and we set $\epsilon_i = 1$. At every iteration, in order to produce new non-dominated points of Problem (10), we use as scalarization the custom weighted sum method with weights $\lambda^* = (\frac{1-\zeta}{4}, 1)^\intercal$, so that at every node we solve the single-objective integer problem $(\text{IIP}^*)^k$:

$$\min_{x \in \mathcal{X}^k \cap \mathbb{Z}^2} \frac{1-\zeta}{4} x_1 + x_2$$

In Figure 1, we report the iterations of FPA* on Problem (10) fixing $\zeta = 0.3$. The non-dominated points found by the algorithm are circled.

In particular Figure 1b reports the optimal solution of $(\text{IIP}^*)^0$ where we plot the level lines of the objective function of $(\text{IIP}^*)^0$. The non-dominated point found is $(4,1)$.

When using a basic FPA we would have two children nodes having as feasible regions respectively $\mathcal{X}^0_1 = \mathcal{X} \cap \{x \in \mathbb{R}^2_{\geq 0} : x_1 \leq 3\}$ and $\mathcal{X}^0_2 = \mathcal{X} \cap \{x \in \mathbb{R}^2_{\geq 0} : x_2 \leq 0\}$.

As expressed in Theorem 4.11 $\mathcal{X}^0_2$ is by construction empty and then not at all considered in FPA*. In Figure 1c the feasible region $\mathcal{X}^0_1$ is plotted and the optimal solution of the corresponding integer problem $(\text{IIP}^*)^1$ is reported in Figure 1d. The new non-dominated point found is $(3,2)$.

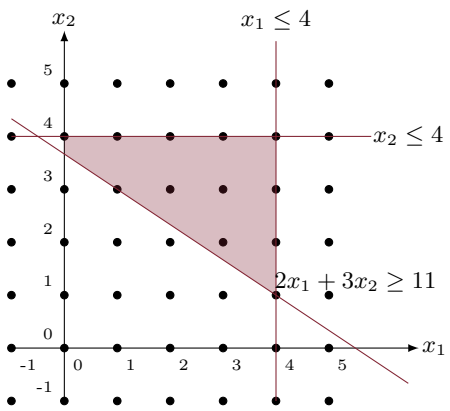
Reapplying the procedure we produce again only one integer problem which gives the Pareto point $(1,3)$ as shown in Figure 1e. Finally in Figure 1f we find the last Pareto point and the algorithm terminates returning the full Pareto front.

4.3. Discussion on the assumptions

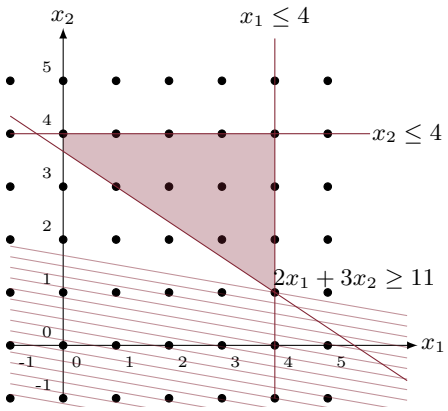
In this section we present some classes of functions that easily satisfy Assumption 2.6 and Assumption 4.1.

As a first step we look for sufficient conditions to have $\gamma > 0$ and $\epsilon \in (0, \gamma]$ computable whenever $f(x) \neq f(z)$ for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ (Assumption 2.6).

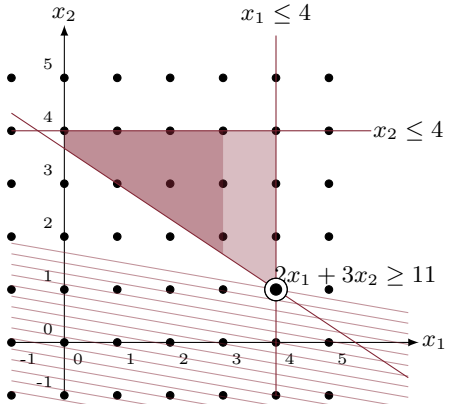
Proposition 4.12. *Assume that $f : \mathbb{Z}^n \to \mathbb{Z}$. Then $\gamma \geq 1$.*



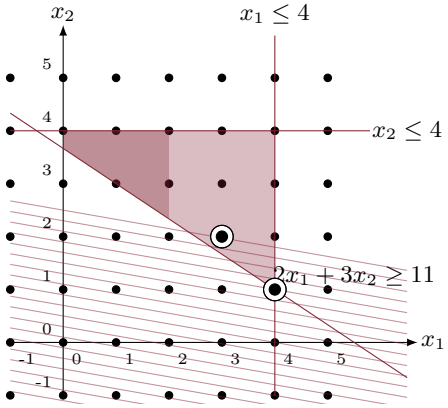
(a) the decision/criterion space



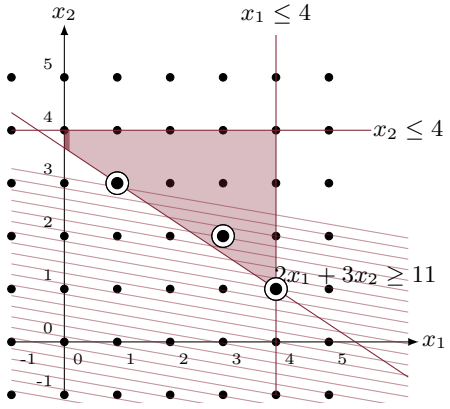
(b) Solution of $(\text{IIP})^0$



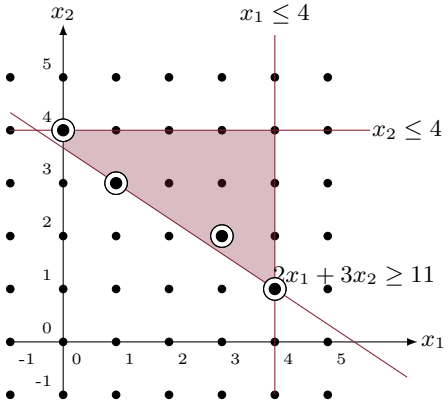
(c) Feasible region of $(\text{IIP}^*)^1$



(d) Solution of $(\text{IIP}^*)^1$ and related cut



(e) Solution of $(\text{IIP}^*)^2$ and related cut



(f) Solution of $(\text{IIP}^*)^3$

Figure 1: The Frontier Partitioner Algorithm applied to Problem (10)

Proof. Since the image of $\mathcal{X} \cap \mathbb{Z}^n$ under f is a subset of $\mathbb{Z}$, we have that $|f(x) - f(z)| \geq 1$, for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ such that $f(x) \neq f(z)$. ∎

Remark 4.13. *As a matter of example of functions satisfying the condition in Proposition 4.12 we have all the polynomials with integer coefficients and in particular $f(x) = c^\intercal x$ with $c \in \mathbb{Z}^n$ and $f(x) = x^\intercal Q x + c^\intercal x$ with $Q \in \mathbb{Z}^{n \times n}$ and $c \in \mathbb{Z}^n$.*

We now look for larger classes of functions for which $\gamma > 0$. We focus on functions defined on rational domains.

Proposition 4.14. *Let $f(x) : \mathbb{Z}^n \to \mathbb{R}$ be a polynomial with rational coefficients, $f(x) \neq 0$. Then $r \in \mathbb{N}$, $r \neq 0$ exists so that $\gamma \geq \frac{1}{r}$.*

Proof. Let $f(x) = \sum_{k=0}^{s} \alpha_k q_k(x)$, where $q_k(x) = \prod_{i=1}^{n} x_i^{m_{ik}}$ and $m_{ik} \in \mathbb{N}$, for $k = 0, \ldots, s$, $i = 1, \ldots, n$. We have $q_k(x) \in \mathbb{Z}$, for $k = 0, \ldots, s$ and for all $x \in \mathbb{Z}^n$. Since $r \in \mathbb{N}$, $r \neq 0$ exists such that $r\alpha_k \in \mathbb{Z}$, $k = 0, \ldots, s$, we have that $g(x) = rf(x)$ satisfies the assumption of Proposition 4.12 and

$$|f(x) - f(z)| \geq \frac{1}{r}, \quad \forall\, x, z \in \mathcal{X} \cap \mathbb{Z}^n : f(x) \neq f(z).$$

∎

Remark 4.15. *Proposition 4.14 holds when $f(x) = x^\intercal Q x + c^\intercal x$, where $Q \in \mathbb{Q}^{n \times n}$ and $c \in \mathbb{Q}^n$.*

Remark 4.16. *Of course Proposition 4.14 holds when $f(x)$ is a linear function with rational coefficients: $f(x) = c^\intercal x$, $c \in \mathbb{Q}^n$.*

Note that the value $r \in \mathbb{N}$ used in Proposition 4.14 is easily calculable as the least common multiple of the denominators of the rational coefficients.

Proposition 4.17. *Let $f(x) = \|Ax + b\|_2$ and assume that $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^n$ and that*

$$\bar{v} = \max_{x \in \mathcal{X} \cap \mathbb{Z}^n} \|Ax + b\|_2^2 \in \mathbb{R}_+ < \infty.$$

Then $\gamma \geq \sqrt{\bar{v} + 1} - \sqrt{\bar{v}}$.

Proof. Since $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^n$ we have that $(Ax + b) \in \mathbb{Z}^m$ for all $x \in \mathcal{X} \cap \mathbb{Z}^n$. Therefore for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ such that $f(x) \neq f(z)$ we get

$$|f(x) - f(z)| = |\|Ax + b\|_2 - \|Az + b\|_2| \geq |\|v\|_2 - \|w\|_2|$$

with $v, w \in \mathbb{Z}^m$ such that $v \neq w$ and they differ exactly for one component, which is the least difference possible. We can assume w.l.o.g. that $v_i = w_i$ for all $i \neq j$ and $w_j = v_j + 1$ and we finally get

$$|f(x) - f(z)| \geq \left| \sqrt{\sum_{i=1}^{m} v_i^2} - \sqrt{\sum_{i=1}^{m} v_i^2 + 2v_j + 1} \right| \geq \sqrt{\|v\|^2 + 1} - \sqrt{\|v\|^2}.$$

Let $g(x) = \|Ax + b\|_2^2$, the function $\sqrt{g+1} - \sqrt{g}$ is monotonically decreasing hence it attains its minimum value at its upper bound $\bar{v}$ and

$$|f(x) - f(z)| \geq \sqrt{\bar{v} + 1} - \sqrt{\bar{v}}, \quad \forall\ x, z \in \mathcal{X} \cap \mathbb{Z}^n : f(x) \neq f(z).$$

□

In Table 1, we report some classes of objective functions that can be considered when using integer programming solvers such as CPLEX [19], Gurobi [18], SCIP [17], Couenne [1] or Bonmin [8], in order to deal with problem $(IIP)^k$. In particular,

- if both $y_i(x)$ $i = 1, 2$ are linear, then $(IIP)^k$ is an Integer Linear Program (ILP)
- if one $y_i(x)$ is written as $\|Ax+b\|_2$, then $(IIP)^k$ is an Integer Second Order Cone Program (ISOCP)
- if one $y_i(x)$ is convex quadratic, then $(IIP)^k$ is a Quadratically Constrained Quadratic Integer Program (QCQIP)
- if one $y_i(x)$ is general convex, then $(IIP)^k$ is a Convex Integer Program (CIP).

$y_i(x) =$	$\gamma \geq$	oracle
$c^\intercal x$ with $c \in \mathbb{Z}^n$	1	*ILP*
$c^\intercal x$ with $c \in \mathbb{Q}^n$	$\frac{1}{r}$	*ILP*
$\|Ax + b\|_2$ with $A \in \mathbb{Z}^{n\times m}$, $b \in \mathbb{Z}^m$	$\sqrt{\bar{v}+1} - \sqrt{\bar{v}}$	*ISOCP*
$x^\intercal Q x + c^\intercal x$ with $Q \succeq 0$, $Q \in \mathbb{Z}^{n\times n}$, $c \in \mathbb{Z}^n$	1	*QCQIP*
$x^\intercal Q x + c^\intercal x$ with $Q \succeq 0$, $Q \in \mathbb{Q}^{n\times n}$, $c \in \mathbb{Q}^n$	$\frac{1}{r}$	*QCQIP*
$: \mathbb{Z}^n \to \mathbb{Z}$, convex	1	*CIP*

Table 1: Classes of functions that satisfy Assumptions 2.6 and 4.1. In the table we denote with r the least common multiple of the denominators of the rational coefficients used in Proposition 4.14. We denote with $\bar{v}$ the value defined in Proposition 4.17.

Remark 4.18. *For the classes of functions mentioned above, we can set $\epsilon_i \in (0, \gamma_i]$ to the values reported in Table 1, as they represent valid lower bounds on γ_i.*

5. Computational aspects

From a computational point of view, several issues may arise when tackling biobjective nonlinear integer problems. A first issue is related to the fact that the cuts introduced in our FPA are nonlinear inequalities in the decision space and this may increase the difficulty of the IIPs to be solved. In Subsection 5.1 we discuss linear approximations of the nonlinear inequalities that provide simpler cuts.
A second issue is related to the numerical instabilities that we experienced when dealing with weighted sum IIPs where one of the two weights is near to zero. In Subsection 5.2 we describe a possible procedure to avoid numerical instabilities, devising an algorithm that combines the two weighted sum scalarized problems (IIP_W) and $(\text{IIP})^*$.

5.1. Circumventing nonlinear inequalities

In the Frontier Partitioner Algorithm, new nodes are built adding inequalities to the feasible set in the decision space. More specifically, at a generic node k, the set $\mathcal{X}^k \cap \mathbb{Z}^n$ is intersected with the following set:

$$C = \{x \in \mathbb{R}^n \,:\, y_i(x) \leq \hat{y}_i^k - \epsilon_i\}, \tag{11}$$

where $i = 1$ or $i = 2$ and $\hat{y}^k$ the non-dominated point found at node k. When $y_i(x)$ is nonlinear we are introducing a nonlinear cut, as the set in (11) is defined according to a nonlinear inequality. However, we do not necessarily need to add nonlinear inequalities: for our purposes, it would suffice to define a valid formulation for the integer set $\{x \in \mathbb{Z}^n \,:\, y_i(x) \leq \hat{y}_i - \epsilon_i\}$, or, in other words, it would suffice to find a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$ such that

$$\{x \in \mathbb{Z}^n \,:\, Ax \leq b\} = \{x \in \mathbb{Z}^n \,:\, y_i(x) \leq \hat{y}_i - \epsilon_i\}.$$

However, from a practical point of view, it is not yet clear how to easily generate a valid formulation. In this section, we investigate how to define linear approximations that lead to a relaxation of the feasible region that can be fruitfully exploited in the FPA.

Under the assumption that $y_i : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable, we have that

$$\nabla y_i(\hat{x}^k)^\intercal (\hat{x}^k - x) \geq y_i(\hat{x}^k) - y_i(x) \geq \epsilon_i.$$

Therefore, we can think of defining $\mathcal{X}_i^k$ intersecting $\mathcal{X}^k$ with the halfspace

$$\{x \in \mathbb{R}^n \,:\, \nabla y_i(\hat{x}^k)^\intercal (\hat{x}^k - x) \geq \epsilon_i\}.$$

The resulting `FPA`, may eventually not terminate with the entire Pareto frontier, as we are not guaranteed to cut all the efficient solutions associated to the current non-dominated point. On one hand, we lose the exactness of the method, on the other we have the advantage of dealing only with linear constraints as long as $\mathcal{X}$ is a polyhedron. For the specific class of problems where the objectives are strictly convex quadratic forms, we can prove the following result:

Proposition 5.1. *Let $y_i(x) = x^\intercal Q x$, with $Q \succ 0$. Let $C \subseteq \mathbb{R}^n$ be defined as in* (11). *Then*

$$C \cap \mathbb{Z}^n \subseteq C^1 \cap C^\infty$$

where

$$C^1 = \left\{ x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_1 \leq \frac{1}{\sqrt{n}} \sqrt{\hat{y}_i - \epsilon_i} \right\}$$

and

$$C^\infty = \left\{ x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_\infty \leq \sqrt{\hat{y}_i - \epsilon_i} \right\}$$

Proof. We have that

$$\begin{aligned} \{x \in \mathbb{Z}^n \,:\, y_i(x) \leq \hat{y}_i - \epsilon_i\} &= \{x \in \mathbb{Z}^n \,:\, x^\intercal Q x \leq \hat{y}_i - \epsilon_i\} \\ &= \{x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_2^2 \leq \hat{y}_i - \epsilon_i\} \\ &= \{x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_2 \leq \sqrt{\hat{y}_i - \epsilon_i}\} = C \cap \mathbb{Z}^n \end{aligned}$$

Using the relations between norms $\sqrt{n}\|x\|_2 \leq \|x\|_1$ and $\|x\|_2 \geq \|x\|_\infty$ we have that

$$C \cap \mathbb{Z}^n \subseteq C^1 = \{x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_1 \leq \tfrac{1}{\sqrt{n}} \sqrt{\hat{y}_i - \epsilon_i}\}$$

$$C \cap \mathbb{Z}^n \subseteq C^\infty = \{x \in \mathbb{Z}^n \,:\, \|Q^{1/2}x\|_\infty \leq \sqrt{\hat{y}_i - \epsilon_i}\}$$

hence we get the result. ∎

Note that both C^1, C^∞ are defined by $2n$ linear inequalities. Hence, in the specific case of problems where the objective functions are strictly convex quadratic forms, $(BOIP)^k$ can be generated using these $4n$ linear inequalities. Again, the resulting `FPA` will be a heuristic approach, as we are not guaranteed of cutting all the efficient solutions associated with the non-dominated points found so far. However, we have the advantage of dealing, at every node, with a quadratic integer programming problem, that can be handled more efficiently than a quadratically constrained quadratic integer programming problem (see, e.g., [9]).

5.2. Managing numerical instabilities

`FPA`* offers a tight bound with respect to the number of integer problems to be solved in order to get the Pareto frontier. From our computational experience, `FPA`* works very well when dealing with biobjective linear integer problems, but it suffers when nonlinearities occur. This is mostly due to the fact that available software have difficulties in solving the nonlinear IIPs when one of the entries of the weight vector is near to zero. We can deal with numerical instabilities following two different strategies:

- iterative weights updating,
- combining the two weighted sum scalarized problems (IIP$_W$) and (IIP*) into an hybrid scheme `SFPA`.

The first strategy can be implemented at almost no additional computational cost. We can update the entries of the weight vector of the single $(\text{IIP}^*)^k$, using the value $\hat{y}^k_{i_1} = y_{i_1}(x^{k-1})$, where x^{k-1} is an optimal solution of $(\text{IIP}^*)^{k-1}$. This is an upper bound of the nadir vector of the subproblem $(\text{BOIP})^k$. From the proof of Proposition 3.4 it follows that this choice leads to a weight vector with larger entries.

The second strategy adopted attempts to combine (IIP_W) with (IIP^*). We call this procedure Stable-FPA (`SFPA`) and it is detailed in Algorithm 3. We associate at each $(\text{BOIP})^k$ the points $\bar{y}^k$ and $\hat{y}^k$, which are respectively lower and upper bounds of the ideal and the nadir point of $(\text{BOIP})^k$. When tackling a subproblem $(\text{BOIP})^k$ we calculate the weights w needed to obtain the custom weighted sum problem $(\text{IIP}^*)^k$ using these bounds. If $w_{min} = \min\{w_1, w_2\}$ is too small we apply the basic `FPA` with the weighted sum problem (IIP_W), using weights $w = (0.5, 0.5)^\intercal$. Otherwise, `SFPA` considers the custom weighted sum problem (IIP^*).
The number of single objective integer problems solved by `SFPA` lies between $|\mathcal{Y}_N| + 2$ and $2|\mathcal{Y}_N| + 1$.

Algorithm 3: FPA (`SFPA`)

Input: (BOIP), $\gamma_i > 0$, $\epsilon_i \in (0, \gamma_i], i = 1, 2$, $\delta \in (0, 1)$
Output: the Pareto frontier $\mathcal{Y}_N$ of $(BOIP)$
Initialization: $(\text{BOIP})^0 = (\text{BOIP})$, $\mathcal{L} = \{(\text{BOIP})^0\}$, $\mathcal{X}^0 = \mathcal{X}$, $\mathcal{Y}_N = \emptyset$
Compute $\hat{x}^j \in \arg\min_{\mathcal{X} \cap \mathbb{Z}^n} y_j(x)$, $j = 1, 2$
Fix the ideal vector $y^{id} = \big(y_1(\hat{x}^1), y_2(\hat{x}^2)\big)^\intercal$
Fix the upper approximation of the nadir point $y^u = \big(y_1(\hat{x}^2), y_2(\hat{x}^1)\big)^\intercal$
Put $\bar{y}^0 = y^{id}$ and $\hat{y}^0 = y^u$; $k = 0$
while $\mathcal{L} \neq \emptyset$ **do**
  Select a node $(\text{BOIP})^k \in \mathcal{L}$ and delete it from $\mathcal{L}$
  Choose $i_1 \in \{1, 2\}$
  Using $\bar{y}^k$ and $\hat{y}^k$, **calculate**

$$w = \begin{cases} \dfrac{\gamma_{i_1} - \epsilon_{i_1}}{\hat{y}^k_{i_1} - \bar{y}^k_{i_1}}, & \text{if } i = i_1 \\ 1, & \text{if } i = i_2 \end{cases}$$

  if $w_{i_1} \leq \delta$ **then**
    Derive $(\text{IIP}_W)^k$ from $(\text{BOIP})^k$ with $\lambda = (0.5, 0.5)^\intercal$
    Call an oracle on $(\text{IIP}_W)^k$
    if *$(IIP_W)^k$ has an optimal solution x^k* **then**
      Set $\mathcal{Y}_N = \mathcal{Y}_N \cup \{y(x^k)\}$
      Build $(\text{BOIP})^k_i$, $i = 1, 2$ from $(BOIP)^k$ as in FPA
      Compute $\bar{y}^{k,i}$ and $\hat{y}^{k,i}$ according to

$$\bar{y}^{k,i}_j = \begin{cases} \bar{y}^k_j, & \text{if } i = j \\ y_j(x^k), & \text{if } i \neq j \end{cases} \quad \text{and} \quad \hat{y}^{k,i}_j = \begin{cases} y_j(x^k), & \text{if } i = j \\ \hat{y}^{k,i}_j, & \text{if } i \neq j \end{cases} \tag{12}$$

      Associate $\bar{y}^{k,i}$ and $\hat{y}^{k,i}$ with $(\text{BOIP})^k_i$, $i = 1, 2$
      Add the new nodes $(\text{BOIP})^k_i$, $i = 1, 2$, to $\mathcal{L}$
    end
  else
    Derive $(IIP^*)^k$ from $(BOIP)^k$ with weights $\lambda^* = (w_1, 1)^\intercal$
    Solve $(IIP^*)^k$ and let x^k be an optimal solution
    Set $\mathcal{Y}_N = \mathcal{Y}_N \cup \{y(x^k)\}$
    Build $(\text{BOIP})^k_{i_1}$,from $(BOIP)^k$ as in FPA*
    Compute $\bar{y}^{k,i_1}$ and $\hat{y}^{k,i_1}$ according to (12)
    Associate $\bar{y}^{k,i_1}$ and $\hat{y}^{k,i_1}$ with $(\text{BOIP})^k_{i_1}$
    Add the new node $(\text{BOIP})^k_{i_1}$ to $\mathcal{L}$
  end
end
Return $\mathcal{Y}_N$

6. Numerical results

Algorithm `FPA`, `FPA`* and `SFPA` are implemented in Java. We addressed only the case of linear and quadratic biobjective integer problems so that the oracles used for solving the corresponding inner integer problem $(\text{IIP})^k$ at each node are respectively the MILP and MIQP solvers of `CPLEX 12.7.1` [19].
All our experiments were carried out on an Intel Core i7 processor running at 2.40 GHz. All running times were measured in CPU seconds.

6.1. Linear biobjective integer instances

The generic biobjective integer linear programming problem is modeled as

$$\begin{aligned} \max \quad & (c_1^{\intercal}x,\, c_2^{\intercal}x) \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_{\geq 0} \end{aligned}$$

We consider two classes of instances. The first class is obtained from instances available at `http://home.ku.edu.tr/~moolibrary/`, where problems have three, four and five objectives. In our experiments, we construct biobjective instances by taking only the first two functions. Parameters are $c_i \in \mathbb{Z}^n$, $i = 1, 2$, $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. In particular, c_{ij} is generated in the ranges $[-100, -1]$ with probability 0.2 and $[0, 100]$ with probability 0.8, $j = 1, \ldots, n$ and $i = 1, 2$. The coefficients a_{kl} are generated in the ranges $[-100, -1]$ with probability 0.1, $[1, 100]$ with probability 0.8 and $a_{kl} = 0$ with probability 0.1. The right-hand side b_k is generated randomly in the range $[100, \sum_{l=1}^{n} a_{kl}]$. These instances are characterized by a Pareto frontier with at most 65 points, which is relatively small to highlight the performance of our algorithms. Hence we randomly generated a second class of instances with a larger Pareto Frontier with a number of non-dominated points between 75 and a thousand which is publicly available at `https://github.com/GiorgioGrani/Biobjective_Instances`.
The second class of instances has been randomly generated with $c_i \in \mathbb{Z}^n$, $i = 1, 2$, $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. We produced 97 instances: 58 of them have a number of constraints which is 83% of the number of variables and their coefficients are chosen such that $c_{ij} \in [-100, -1]$ with probability 0.2 and $c_{ij} \in [0, 100]$ with probability 0.8, $j = 1, \ldots, n$; $i = 1, 2$. The coefficients a_{kl} are generated in the ranges $[-100, -1]$ with probability 0.05, $[1, 100]$ with probability 0.9 and $a_{kl} = 0$ with probability 0.05. The right-hand side b_k is generated randomly in the range $[0, \sum_{l=1}^{n} a_{kl}]$. The remaining 39 instances have exactly 10 constraints each and their coefficients are chosen such that c_{ij} is generated in the range $[-100, -1]$ with probability 0.02, in the range $[0, 100]$ with probability 0.08 and they are set to zero with probability 0.9, $j = 1, \ldots, n$; $i = 1, 2$. The coefficients a_{kl} are generated in the ranges $[-100, -1]$ with probability 0.2, $[1, 100]$ with

probability 0.6 and $a_{kl} = 0$ with probability 0.2. The right-hand side b_k is generated randomly in the range $[0, \sum_{l=1}^{n} a_{kl}]$.

Note that, for both classes of instances, the condition in Proposition 4.12 is satisfied, so that $\gamma_i \geq 1$ and we can set $\epsilon_i = 1$ for $i = 1, 2$.

In order to assess the performance of the algorithms considered, we make use of performance profiles, as proposed in [14]. Given our set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, we compare the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. To this end, we define the performance ratio $r_{p,s} = t_{p,s} / \min\{t_{p,s'} : s' \in \mathcal{S}\}$, where $t_{p,s}$ is the computational time, and we consider a cumulative distribution function $\rho_s(\tau) = |\{p \in \mathcal{P} : \ r_{p,s} \leq \tau\}|/|\mathcal{P}|$. The performance profile for $s \in S$ is the plot of the function ρ_s.

We compared four algorithms: `FPA`*, `FPA` with weighted sum (`FPA-W`), `SFPA` and the Balanced Box Method (`BBM`) proposed in [4]. `BBM` is one of the state-of-the-art algorithms for biobjective integer linear problems; it uses lexicographic method and it divides the criterion space into rectangles.

In order to study the effect of changing the weight vector in the definition of $(\mathrm{IIP}_W)^k$, we consider three different settings for $\lambda \in \mathbb{R}^2$ within `FPA-W`, namely

$$\lambda \in \left\{ \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix}, \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix} \right\}.$$

In Table 2, we report the results obtained on the two classes of instances by applying `FPA-W`, where $(\mathrm{IIP}_W)^k$ is defined using one of the weight vectors above for every k.

The results of `BBM`, `FPA`* and `SFPA` on the same linear instances are reported in Table 3. In the tables 2 and 3, the instances are grouped according to the cardinality of the Pareto frontier $|\mathcal{Y}_N|$. We report the number of instances (# inst.) belonging to a specific range of $|\mathcal{Y}_N|$, the average (avg), the minimum (min) and the maximum (max) CPU time in seconds needed to detect the entire Pareto frontier.

We note from the results in Table 2 that the performances of `FPA-W` using different weight vectors are all very similar and there is no clear winner. However, the choice of the weight vector $\lambda = (0.25, 0.75)^\intercal$ seems to be slightly better on average.

As expected from the theoretical bound obtained, `FPA`* outperforms `FPA-W`, `BBM` and `SFPA` on all instances.

We further report in Figures 2 and 3 the performance profiles for the first and the second group of instances, respectively. Performance profiles confirm the analysis made above.

Another important aspect to be taken into account in the comparisons is the number of single objective integer programs solved by each algorithm. We report these results in a compact form in Figures 4 and 5 using box plots. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. It is evident that `FPA`* saves solver calls. It is also evident that `SFPA` oscillates between the performances of `FPA`* and `FPA-W`.

$\lvert\mathcal{Y}_N\rvert$	# inst.	$\lambda = (0.1, 0.9)^\intercal$ avg	min	max	$\lambda = (0.25, 0.75)^\intercal$ avg	min	max	$\lambda = (0.5, 0.5)^\intercal$ avg	min	max
≤ 10	66	0.7	0.1	11.6	0.7	0.1	8.2	0.7	0.1	9.2
$> 10, \leq 20$	71	8.4	0.2	105.7	8.1	0.2	103.2	8.4	0.2	102.3
$> 20, \leq 30$	37	47.6	0.4	395.9	46.5	0.4	392.3	50.3	0.3	414.8
$> 30, \leq 40$	19	29.0	0.5	130.5	29.3	0.5	132.1	31.1	0.5	155.3
$> 40, \leq 50$	16	221.0	0.6	894.8	220.1	0.6	906.6	222.1	0.7	872.8
$> 50, \leq 65$	10	600.5	7.2	2313.3	582.6	7.3	2252.4	612.8	7.0	2508.3
≤ 70	13	13.9	0.9	80.0	13.4	0.9	76.4	27.7	0.8	253.0
$> 70, \leq 150$	16	54.9	5.1	172.1	51.7	5.1	180.8	57.3	5.2	176.4
$> 150, \leq 300$	19	213.2	10.3	2092.8	207.2	10.1	2058.1	211.7	10.0	2021.8
$> 300, \leq 500$	26	297.4	34.5	963.8	285.9	29.6	949.5	329.9	29.9	1574.3
$> 500, \leq 1000$	21	1122.8	58.8	3669.2	1073.2	59.0	3657.9	1185.8	55.2	3607.7

Table 2: Results (CPU time) of `FPA-W` with different weight vectors on biobjective integer linear programming instances.

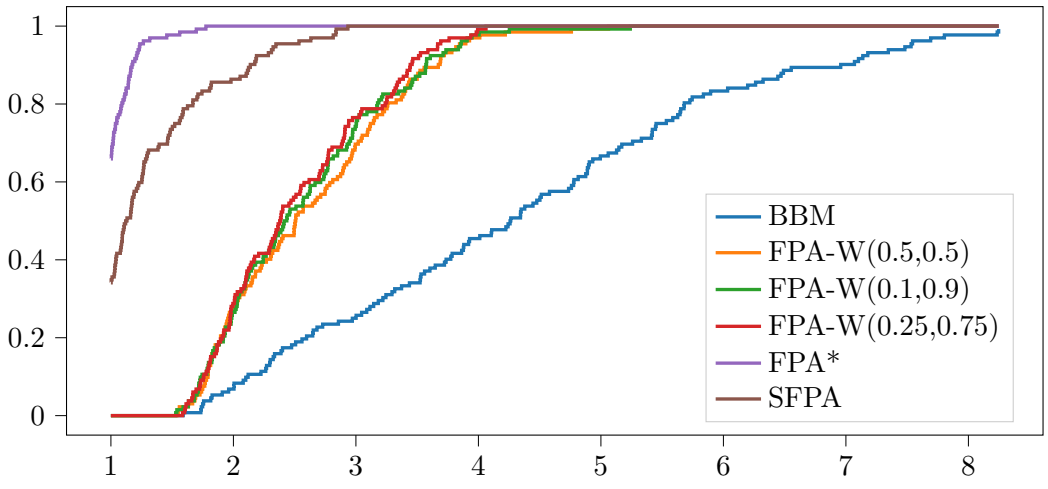


Figure 2: Performance profiles with respect to CPU time on the first group of instances.

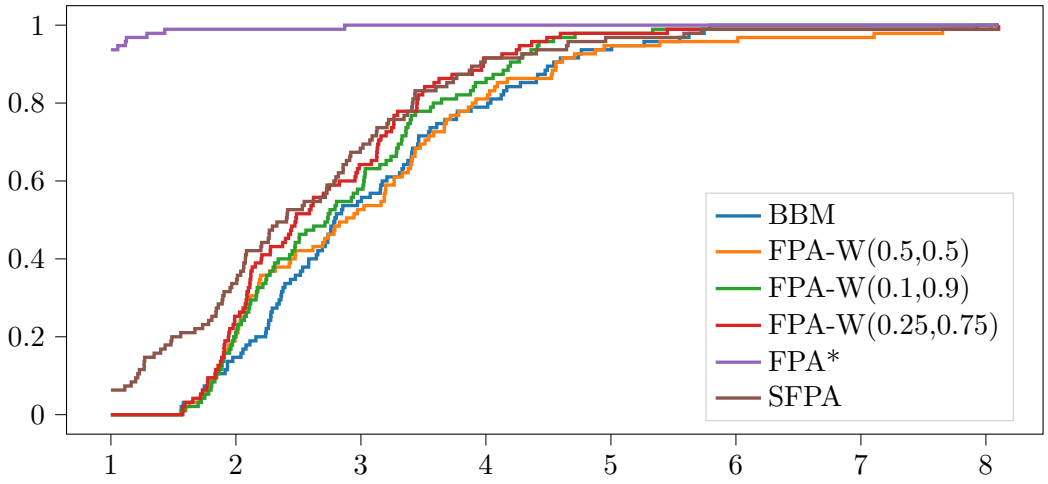


Figure 3: Performance profiles with respect to CPU time on the second group of instances.

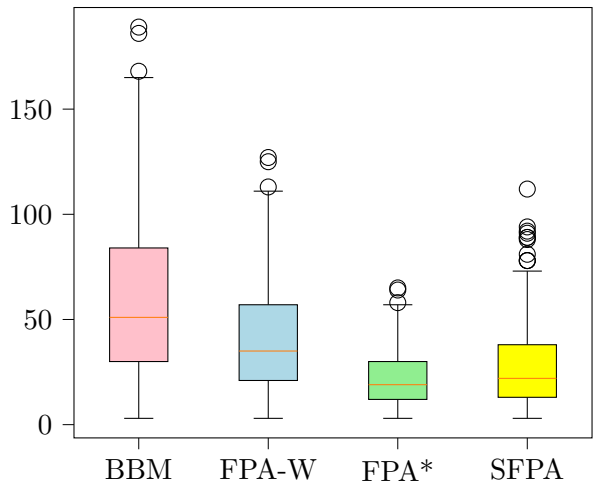


Figure 4: Box plot with respect to the number of oracle calls in the first group of instances.

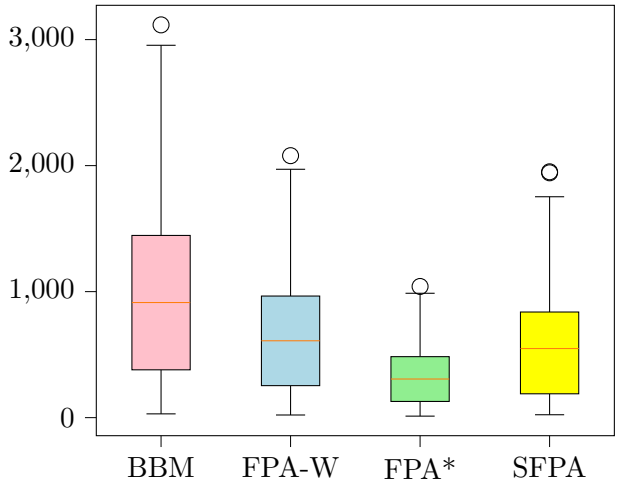


Figure 5: Box plot with respect to the number of oracle calls in the second group of instances.

$\lvert\mathcal{Y}_N\rvert$	# inst.	BBM avg	BBM min	BBM max	FPA* avg	FPA* min	FPA* max	SFPA avg	SFPA min	SFPA max
≤ 10	66	1.0	0.1	15.9	0.3	0.1	2.7	0.3	0.1	3.3
$> 10, \leq 20$	71	13.6	0.2	134.7	3.2	0.1	35.2	4.0	0.1	50.2
$> 20, \leq 30$	37	76.2	0.3	544.0	16.6	0.2	146.2	24.2	0.2	263.5
$> 30, \leq 40$	19	49.0	0.4	255.1	9.7	0.3	46.3	11.0	0.3	41.2
$> 40, \leq 50$	16	299.3	0.5	1167.2	76.1	0.4	325.4	149.3	0.4	710.7
$> 50, \leq 65$	10	830.4	9.6	3298.1	237.6	3.1	996.7	278.0	5.5	1142.1
≤ 70	13	13.1	0.6	55.3	6.2	0.4	31.9	11.3	0.8	33.7
$> 70, \leq 150$	16	57.5	5.7	175.1	19.4	3.3	64.3	34.4	5.0	116.9
$> 150, \leq 300$	19	198.2	8.9	1885.9	62.0	5.7	589.0	190.5	8.3	1,683.1
$> 300, \leq 500$	26	333.3	32.8	1153.0	95.4	15.2	393.4	254.8	31.2	1,073.3
$> 500, \leq 1000$	21	1076.6	65.3	3752.4	350.7	30.9	1216.1	978.1	44.4	2751.3

Table 3: Comparison of `BBM`, `FPA`*, `SFPA` on biobjective integer linear programming instances (CPU time).

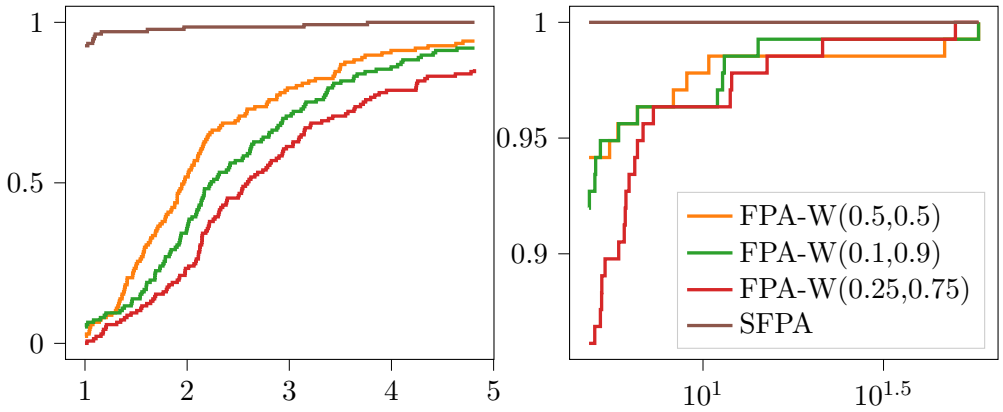


Figure 6: Performance profile on quadratic instances.

6.2. Numerical experiments on quadratic instances

The generic biobjective integer quadratic programming problem is modeled as

$$\begin{aligned} \min \quad & (x^\intercal Q_1 x + c_1^\intercal x,\ x^\intercal Q_2 x + c_2^\intercal x) \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_{\geq 0}. \end{aligned}$$

We considered a subgroup of the first class of instances used in Section 6.1 (`http://home.ku.edu.tr/~moolibrary/`), where we selected only the instances with no more than 60 variables. In the objective functions we added the quadratic term $x^\intercal Q_i x$, with $Q_i \succeq 0$, $Q_i \in \mathbb{Z}^{n\times n}$, $i = 1, 2$ are randomly generated. We generated the matrix Q_i as $L_i L_i^\intercal$ where $L_i \in \{0, 1\}^{n\times h}$, where h integer and randomly chosen in $[1, n]$. The generic element l of the matrix L_i is chosen to be 0 with probability 0.8 and to be 1 with probability 0.2.

Note that the condition in Proposition 4.12 is satisfied, so that $\gamma_i \geq 1$ and we can set $\epsilon_i = 1$ for $i = 1, 2$. The test problems are publicly available at `https://github.com/GiorgioGrani/Biobjective_Instances`.

For the quadratic instances, we experienced that the choice of the weights in `FPA`* can affect numerical stability as explained in Section 5. Indeed CPLEX fails in detecting positive semidefiniteness of the matrices Q_i on several instances so that we are not reporting the results of `FPA`*. We used for comparison only `SFPA` and `FPA-W` with the different weights.

In Table 4, we group the instances according to the cardinality of the Pareto frontier $|\mathcal{Y}_N|$. We consider seven different ranges and for each range of $|\mathcal{Y}_N|$, we report the number of instances (# inst.) belonging to that range and the average (avg), the minimum (min) and the maximum (max) CPU time in seconds needed to detect the entire Pareto frontier by `SFPA` or by the different `FPA-W`s.

In Figure 6 we report the performance profile with respect to the CPU time. `SFPA` clearly outperforms `FPA-W`.

$\lvert\mathcal{Y}_N\rvert$	# inst.	$\lambda = (0.1, 0.9)^\intercal$ avg	min	max	$\lambda = (0.25, 0.75)^\intercal$ avg	min	max
≤ 10	36	3.4	0.0	18.9	3.4	0.0	19.0
$> 10, \leq 20$	39	25.0	0.4	70.4	27.5	0.4	74.1
$> 20, \leq 30$	30	448.3	8.8	5772.8	587.8	4.5	8203.3
$> 30, \leq 40$	31	271.4	0.7	1729.3	303.3	0.8	1618.8
$> 40, \leq 50$	4	2122.2	454.5	3767.0	2219.6	407.4	6387.8
$> 50, \leq 65$	12	803.6	106.6	3680.5	953.6	105.1	3632.4
$> 65, \leq 100$	7	1076.8	26.0	2681.3	1794.0	32.1	5814
$\lvert\mathcal{Y}_N\rvert$	# inst.	$\lambda = (0.5, 0.5)^\intercal$ avg	min	max	SFPA avg	min	max
≤ 10	36	3.8	0.0	18.9	1.5	0.0	9.9
$> 10, \leq 20$	39	30.3	0.4	75.3	13.4	0.3	49.9
$> 20, \leq 30$	30	712.0	4.3	8851.7	96.6	2.0	992.3
$> 30, \leq 40$	31	401.2	0.9	2402.3	111.8	0.4	558.3
$> 40, \leq 50$	4	2806.5	501.6	7371.5	895.2	130.3	1705.6
$> 50, \leq 65$	12	1557.6	113.9	7714.2	439.0	66.2	2133.1
$> 65, \leq 100$	7	652.6	16.9	1427.1	454.2	16.2	1389.0

Table 4: Comparison on biobjective integer quadratic programming instances - `FPA` applied with different weight vectors and `SFPA` (CPU time).

7. Conclusions

We presented a criterion space search algorithm able to detect the entire Pareto frontier of biobjective integer programming problems. Using a suitable solution technique, a single-objective integer programming problem needs to be tackled at every node of the branching tree. In our algorithm, we use suitable weighted sum scalarized problems. We can prove that, under a particular choice of the weights which encompass the property of the lexicographic optimization, the number of integer problems to be solved in order to get the full Pareto frontier $\mathcal{Y}_N$ is $|\mathcal{Y}_N| + 2$. This represents a very good bound in the context of criterion space search algorithms for biobjective integer programming problems.
The approach is based on a partition of the criterion space based on linear inequalities that are violated by already detected Pareto points. These inequalities rely on a problem-dependent parameter easily calculable for some classes of problems, including integer convex quadratic instances. Therefore, our approach can handle biobjective nonlinear integer problems, as long as the objective functions satisfy specific properties and an oracle able to solve the inner integer problem is available. In order to overcome numerical issues when solving the scalarized problems, we devised a hybrid version of our algorithm. Numerical results on both linear and convex quadratic biobjective integer instances confirm the efficiency of our approach.

8. Acknowledgments

The authors acknowledge Prof. Hadi Charkhgard for having kindly provided the code of the balanced box method [4], Prof Gabriele Eichfelder for fruitful discussion and the anonymous referees for their comments that lead to a significantly improved version of the paper.

References

[1] Pietro Belotti. Couenne: a user's manual. Technical report, Lehigh University.

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

[3] 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.

[4] 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.

[5] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. The l-shape search method for triobjective integer programming. *Mathematical Programming Computation*, 8(2):217–251, 2016.

[6] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European Journal of Operational Research*, 260(3):904–919, 2017.

[7] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. The quadrant shrinking method: A simple and efficient algorithm for solving tri-objective integer programs. *European Journal of Operational Research*, 260(3):873–885, 2017.

[8] Pierre Bonami, Lorenz T Biegler, Andrew R Conn, Gérard Cornuéjols, Ignacio E Grossmann, Carl D Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, et al. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[9] C. Buchheim, M. De Santis, S. Lucidi, F. Rinaldi, and L. Trieu. A Feasible Active Set Method with Reoptimization for Convex Quadratic Mixed-Integer Programming. *SIAM Journal on Optimimization*, 26(3):1695–1714, 2016.

[10] Valentina Cacchiani and Claudia D'Ambrosio. A branch-and-bound based heuristic algorithm for convex multi-objective minlps. *European Journal of Operational Research*, 260:920–933, 2017.

[11] LG Chalmet, L Lemonidis, and DJ Elzinga. An algorithm for the bi-criterion integer programming problem. *European Journal of Operational Research*, 25(2):292–300, 1986.

[12] Ada Che, Vladimir Kats, and Eugene Levner. An efficient bicriteria algorithm for stable robotic flow shop scheduling. *European Journal of Operational Research*, 260(3):964–971, 2017.

[13] Kerstin Dächert, Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. Efficient computation of the search region in multi-objective optimization. *European Journal of Operational Research*, 260:841–855, 2017.

[14] E. Dolan and J.Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[15] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[16] Matthias Ehrgott and Xavier Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers and OR*, 34:2674–2694, 2007.

[17] Ambros Gleixner, Leon Eifler, Tristan Gally, Gerald Gamrath, Patrick Gemander, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Stefan Vigerske, Dieter Weninger, Jonas T. Witt, and Jakob Witzig. The scip optimization suite 5.0. Technical Report 17-61, ZIB, Takustr. 7, 14195 Berlin, 2017.

[18] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2016.

[19] IBM ILOG CPLEX Optimizer, 2018. https://www.ibm.com/products/ilog-cplex-optimization-studio.

[20] Gokhan Kirlik and Serpil Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488, 2014.

[21] Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. On the representation of the search region in multi-objective optimization. *European Journal of Operational Research*, 245:767–778, 2015.

[22] Banu Lokman and Murat Köksalan. Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 57(2):347–365, 2013.

[23] George Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2):455–465, 2009.

[24] 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.

[25] George Mavrotas and Danae Diakoulaki. Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied mathematics and computation*, 171(1):53–71, 2005.

[26] Siamak Moradi, Andrea Raith, and Matthias Ehrgott. A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem. *European Journal of Operational Research*, 244(2):369–378, 2015.

[27] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149–165, 2010.

[28] 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.

[29] Ted K Ralphs, Matthew J Saltzman, and Margaret M Wiecek. An improved algorithm for biobjective integer programming and its application to network routing problems. *Annals of Operations Research*, 73:253–280, 2004.

[30] Ted K Ralphs, Matthew J Saltzman, and Margaret M Wiecek. An improved algorithm for solving biobjective integer programs. *Annals of Operations Research*, 147(1):43–70, 2006.

[31] Antonio Sedeño-Noda and C González-Martın. An algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 28(2):139–156, 2001.

[32] Francis Sourd and Olivier Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, 20(3):472–484, 2008.

[33] Thomas Stidsen, Kim Allan Andersen, and Bernd Dammann. A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science*, 60(4):1009–1032, 2014.

[34] John Sylva and Alejandro Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55, 2004.