A Branch and Cut Algorithm for Biobjective Integer Programming

Marianna De Santis

Corresponding author
Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome. Via Ariosto 25, Rome, 00185, Italy.
E-mail: marianna.desantis@uniroma1.it Tel +39 0677274 0XX

Giorgio Grani

Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome. Via Ariosto 25, Rome, 00185, Italy.
E-mail: g.grani@uniroma1.it

Laura Palagi

Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome. Via Ariosto 25, Rome, 00185, Italy.
E-mail: laura.palagi@uniroma1.it

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, each of them returning a Pareto optimal point. The feasible sets of the integer programs are built from the original feasible set, by adding cuts that separate efficient solutions. Beside biobjective linear integer optimization problems, our approach can be easily extended to handle biobjective nonlinear integer 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 presented.

Keywords: Biobjective optimization; mettere 5 al max

1. Introduction

Most real-world optimization problems in the areas of applied sciences, engineering and economics involve multiple, often conflicting, goals. In the mathematical

model 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 then in the context 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 $\mathcal{X} \subseteq \mathbb{R}^n$ 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 $y : \mathbb{R}^n \to \mathbb{R}^p$ represents the feasible set in the criterion space, denoted by $\mathcal{Y} = \{z \in \mathbb{R}^p : z = y(x) \text{ for some } x \in \mathcal{X}\}$. The challenging nature of MOIPs and the need of methods that give performance guarantees in terms of the quality of the solution, motivated the development of exact approaches for multiobjective integer programming problems. The first branch-and-bound algorithm for solving multiobjective mixed 0-1 integer programs was proposed by Mavrotas and Diakoulaki [21], who improved and extended their work in [20, 22]. In [1, 2] Belotti and coauthors propose a branch-and-bound algorithm for biobjective mixed-integer problems. They focus on the idea of finding the complete Pareto frontier for a relaxed subproblem, using this information to derive practical fathoming rules. Among the branch-and-bound algorithms for biobjective mixed integer linear programming problems, we further mention [28, 30]. In the application context, biobjective minimum cost flow problems have been addressed in [24, 26, 29]. We also mention works on network routing problems [27], on the the stable robotic flow shop scheduling problem [9] and on the assignement problem with three objectives [25].

The algorithm we propose is a branch and cut algorithm that belongs to the class of criterion space search algorithms, i.e., it is an algorithm that works in the space of objective functions' values. Criterion space search algorithms find nondominated points by addressing a sequence of single-objective optimization problems. Once a nondominated point is computed, the dominated parts of the criterion space are removed and the algorithms go on looking for new nondominated points. One of the first criterion space search algorithm is the algorithm proposed in [31], improved in [18, 19]. Several contributions in this context have been given by Boland and coauthors [3, 4, 5, 6]. The balanced box method proposed in [3] is the criterion space algorithm for biobjective integer linear programming we compare with in our numerical experience.

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 discuss popular techniques in multiobjective optimization such as scalarization techniques and Goal programming. In Section 3, we introduce and analyze our algorithm. Our numerical experience is presented in Section 4. Section 5 concludes.

2. Notations and Preliminaries

We focus on biobjective integer programming, 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 continuously differentiable.

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 $z \in \mathcal{X} \cap \mathbb{Z}^n$, $z \neq x$, 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 nondominated 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 $z \in \mathcal{X} \cap \mathbb{Z}^n$, $z \neq x$, 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 nondominated point. The set of all nondominated 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 $y^{id} \in \mathbb{R}^2$ such that*

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

Assumption 2.4 (Problem BOIP)**.** *We assume that ideal objective vector y_i^{id}, $i = 1, 2$ exists and is finite.*

Our goal is to design an algorithm able to produce the entire Pareto frontier of Problem (BOIP) and Assumption 2.4 ensures the finiteness of $\mathcal{Y}_N$ of problem (BOIP) (as we will prove in Proposition 3.8), that is a standard assumption when defining exact algorithms for MOIP (see, e.g., [28]).

2.1. Scalarization techniques

The general idea of scalarization is that of converting a multiobjective optimization problem into a single objective optimization problem, known as *scalarized problem*. The scalarized problem depends on suitably defined parameters.

Definition 2.5. *A scalarization technique for Problem* (MOIP) *is correct if the optimal solution of the scalarized problem is efficient.*

Definition 2.6. *A scalarization technique for Problem* (MOIP) *is complete if all efficient solutions can be found by solving a scalarized problem with appropriately chosen parameters.*

In the following, we recall the weighted sum method and the ϵ-constraint method, as examples of well known and used scalarization techniques (we refer the interested reader to [13, 23] for further details). Both these techniques are correct, while completeness holds only for the ϵ-constraint method.
When applying the weighted sum method to Problem (MOIP), the scalarized problem is defined as:

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \sum_{i=1}^{p} \lambda_i y_i(x), \tag{1}$$

where $\sum_{i=1}^{p} \lambda_i = 1$, with $\lambda_i \geq 0$, for $i = 1, \ldots, p$.

Proposition 2.7. *Let $\lambda \in \mathbb{R}^p$ such that Problem* (1) *has a unique solution $\hat{x} \in \mathcal{X} \cap \mathbb{Z}^n$. Then, $\hat{x}$ is an efficient solution for Problem* (MOIP).

The ϵ-constraint method selects one objective $y_\ell(x)$ among $(y_1(x), \ldots, y_p(x))$ and introduces $p-1$ constraints of the form $y_i(x) \leq \epsilon_i$, for $i \neq \ell$. Then, the scalarized problem considered is the following

$$\begin{array}{ll} \min & y_\ell(x) \\ \text{s.t.} & y_i(x) \leq \epsilon_i \; \forall i = 1, \ldots, p;\, i \neq \ell \\ & x \in \mathcal{X} \cap \mathbb{Z}^n \end{array} \tag{2}$$

Proposition 2.8. *The solution $\hat{x} \in \mathcal{X} \cap \mathbb{Z}^n$ is an efficient solution for Problem* (MOIP) *if and only if it is solution of Problem* (2) *for every choice of $\ell \in \{1, \ldots, p\}$, with $\epsilon_i = y_i(\hat{x})$, $i \neq \ell$.*

Proposition 2.9. *If $\hat{x} \in \mathcal{X} \cap \mathbb{Z}^n$ is the unique solution of Problem* (2) *for some $\ell \in \{1, \ldots, p\}$, with $\epsilon_i = y_i(\hat{x})$, $i \neq \ell$, then $\hat{x}$ is an efficient solution for Problem* (MOIP).

As a further scalarization technique we recall Goal programming. The idea behind Goal programming techniques for multiobjective optimization is that of reaching a set of multiple goals as closely as possible (we refer the interested reader to [17, 23]). Let $y^{id} \in \mathbb{R}^p$ be the ideal vector of Problem (MOIP) (see definition 2.3). We look for the solution of the following problem

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \|y(x) - y^{id}\|_q \tag{3}$$

where $\|\cdot\|_q$ denotes the q-norm of a vector in $\mathbb{R}^p$, with $1 \leq q \leq \infty$. In particular, if $q = \infty$, Problem (3) is known as *Tchebycheff Problem.*

Proposition 2.10. *Every solution of Problem* (3) *with $1 \leq q < \infty$ is an efficient solution of Problem* (MOIP).

When $q = \infty$ we have the following weaker result.

Proposition 2.11. *The Tchebycheff Problem, i.e. Problem* (3) *with* $q = \infty$ *admits at least one solution that is an efficient solution of Problem* (MOIP).

Remark 2.12. *Note that when* $q = 1$*, Problem* (3) *is equivalent to*

$$\min_{x \in \mathcal{X} \cap \mathbb{Z}^n} \sum_{i=1}^{n} y_i(x),$$

as $y_i(x) \geq y_i^{id}$*, for all* $x \in \mathcal{X} \cap \mathbb{Z}^n$ *and* $i = 1, \ldots, n$*. Hence using goal programming with* $q = 1$ *preserves the structure of the original problem and it is not necessary to compute* y^{id}*.*

3. The Frontier Partitioner Algorithm

In order to properly state our algorithm, we need to give the following definition:

Definition 3.1. *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a continuous function. We define* $\gamma \in \mathbb{R}$ *as the maximum value such that* $|f(x) - f(z)| \geq \gamma$*, for all* $x, z \in \mathcal{X} \cap \mathbb{Z}^n$*.*

Of course it results $\gamma \geq 0$.
The Frontier Partitioner Algorithm (FPA) is a branch and cut algorithm. At each node of the branching tree the subproblem $(BOIP)^k$

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

is considered where $\mathcal{X}^k \subset \mathcal{X}$. For $k = 0$ we define $(BOIP)^0$=(BOIP) and $\mathcal{X}^0 = \mathcal{X}$ and $\mathcal{Y}^0 = \mathcal{Y}$.
Problem $(BOIP)^k$ is addressed by means of a correct scalarization technique as those reported in Section 2.1. First $(BOIP)^k$ is transformed into the following integer program, denoted as $(INLP)^k$:

$$\min_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} f(x), \tag{4}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is appropriately defined.
rivedere come scrivere in generale la scalarizzazione
We need the following assumption on the scalarized problem $(INLP)^k$.

Assumption 3.2 (Solvability of the scalarized problem)**.** *An oracle that either certifies the infeasibility of problem* $(INLP)^k$ *or it returns an optimal solution of* $(INLP)^k$*.*

By assumption 3.2, when tackling a node and solving $(INLP)^k$, either we find out that it is infeasible (and this implies that there are no efficient points) and the node is pruned or we get an efficient solution $\hat{x}^k \in \mathcal{X}^k \cap \mathbb{Z}^n$ for $(BOIP)^k$, as proven in Proposition 3.9. In the latter case two children nodes in the branching tree are built as follows:

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

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

being $\epsilon_i \in (0, \gamma_i]$, where γ_i is defined as in Definition 3.1 for functions $y_i(x)$, $i = 1, 2$.

Remark 3.3. *If $\gamma_i > 0$, we have that $y(x) \neq \hat{y}^k$ for any $x \in \mathcal{X}_1^k \cap \mathbb{Z}^n$ and any $x \in \mathcal{X}_2^k \cap \mathbb{Z}^n$. Hence the inequalities used to define $\mathcal{X}_i^k$ $i = 1, 2$ cut the nondominated point $\hat{y}^k$ and by addressing the children nodes we eventually find new nondominated points on the Pareto frontier.*

Remark 3.4. *The cuts $y_i(x) \leq \hat{y}_i^k - \epsilon_i$ are linear in the criterion space and preserve the same structure of the function $y(x)$ in the decision space.*

Remark 3.5. *We observe that at each node of the branching tree the feasible region $\mathcal{X}^k$ is obtained from the original $\mathcal{X}$ adding at most two cuts. 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.*

The algorithm iterates producing a list of BOIPs and of nondominated points. The scheme of the Frontier Partitioner Algorithm (`FPA`) is reported in Algorithm 1.
We prove in Section 3.2 that under suitable assumptions, `FPA` terminates finitely and returns the entire Pareto frontier $\mathcal{Y}_N$.

3.1. Toy example

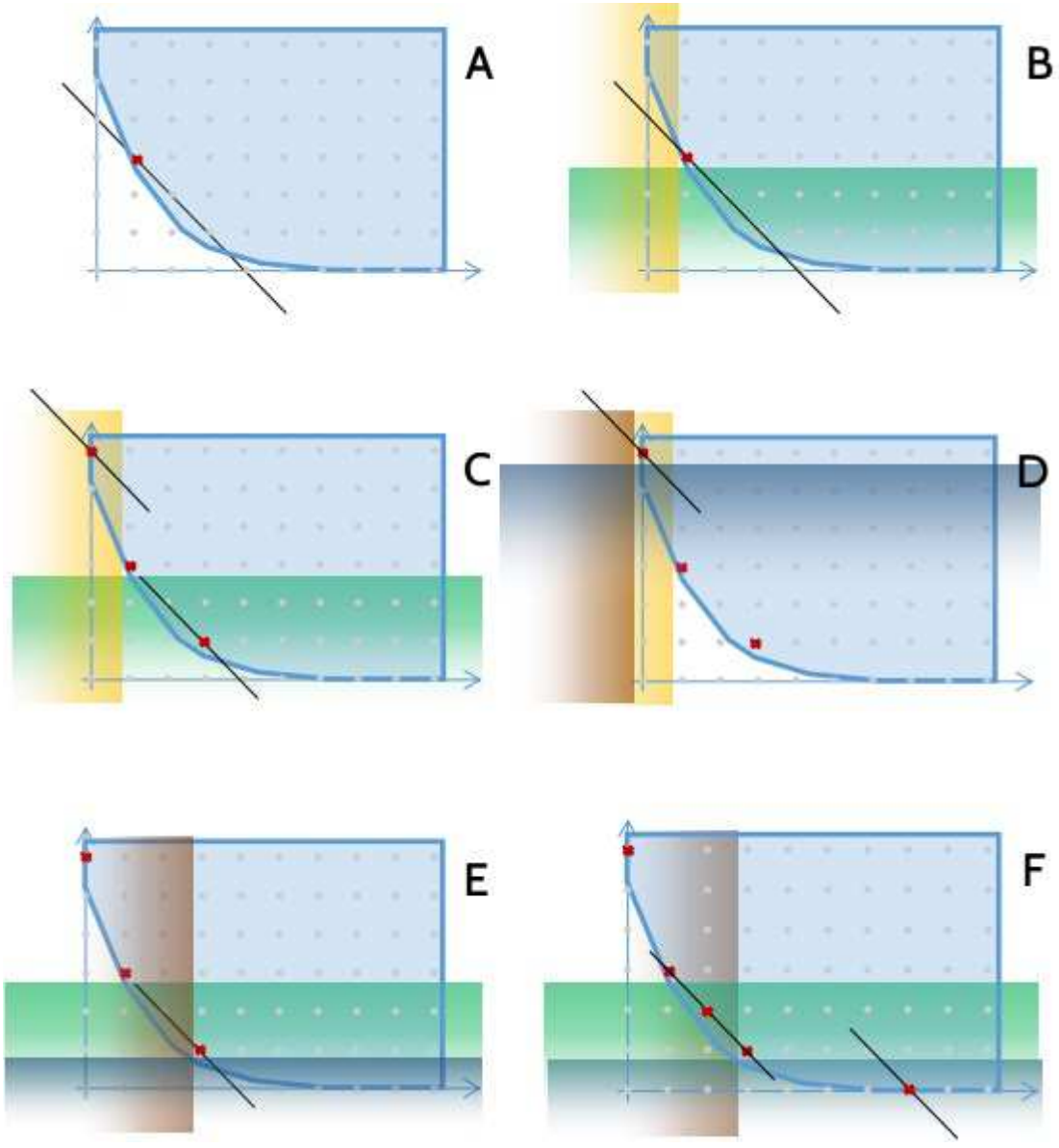
Rifare la figura togliendo la retta che non si capisce cosa è; inserire in modo più evidente i tagli indotti dalla solzuione di Pareto scelta; psfrag oppure fatte meglio le scritte sulla fgiura; riportare l'istanza disegnata

In Figure 1 we report the iterations of `FPA` on a simple instance.

3.2. Convergence Analysis

The Frontier Partitioner Algorithm requires that $\gamma > 0$ for all objective functions $y_i(x)$, $i = 1, 2$ in (BOIP).

Figure 1: The Frontier Partitioner Algorithm (`FPA`). **A:** First nondominated point found by Goal programming with $q = 1$ ($\hat{y}^0$). **B:** $\mathcal{X}_1^0$ and $\mathcal{X}_2^0$. **C:** The nondominated points $\hat{y}^1$ (top-left) and $\hat{y}^2$ (bottom-right). **D:** Both $(BOIP)_1^1$ and $(BOIP)_2^1$ are infeasible. **E:** Addressing $(BOIP)_1^2$ and $(BOIP)_2^2$. **F:** The complete Pareto frontier.



Algorithm 1 The Frontier Partitioner Algorithm (FPA)

Input: $\mathcal{L} = \{(BOIP)^0\}$, $\mathcal{X}^0 = \mathcal{X}$, $\mathcal{Y}^0 = \mathcal{Y}$, $\mathcal{Y}_N = \emptyset$, $\gamma_i > 0$, $\epsilon_i \in (0, \gamma_i], i = 1, 2$

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

1: **while** $\mathcal{L} \neq \emptyset$ **do**
2: **Select** a node $(BOIP)^k \in \mathcal{L}$ and delete it from $\mathcal{L}$
3: **Solve** $(INLP)^k$.
4: **if** $(INLP)^k$ is infeasible **then**
5: **go** to Step 2
6: **else**
7: **Set** $\mathcal{Y}_N = \mathcal{Y}_N \cup \hat{y}^k$, where $\hat{y}^k = y(\hat{x}^k)$ and $\hat{x}^k$ is a solution of $(INLP)^k$
8: **end if**
9: **Build** $(BOIP)_i^k$ from $(BOIP)^k$, as

$$\min_{x \in \mathcal{X}_i^k \cap \mathbb{Z}^n} y(x) \qquad \text{with} \qquad \mathcal{X}_i^k = \mathcal{X}^k \cap \{x \in \mathbb{R}^n \,:\, y_i(x) \leq \hat{y}_i^k - \epsilon_i\}, \quad i = 1, 2,$$

Add the new nodes $(BOIP)_1^k$ and $(BOIP)_2^k$ to $\mathcal{L}$
10: **end while**

Assumption 3.6 (Positive gap value)**.** *The functions $y_i : \mathbb{R}^n \to \mathbb{R}$ in Problem* (BOIP) *satisfy $\gamma_i > 0$, for $i = 1, 2$.*

As a first step we prove that the cuts used in Algorithm 1 induce a partition of the decision space.

Proposition 3.7. *Let assumption 3.6 holds. Then $\mathcal{X}_1^k \cap \mathcal{X}_2^k = \emptyset$ 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) corresponding to the nondominated value $\hat{y}^k$. Assume by contradiction that $\mathcal{X}_1^k \cap \mathcal{X}_2^k \neq \emptyset$. Then $x \in \mathcal{X}_1^k \cap \mathcal{X}_2^k$ exists and it 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). ∎

Proposition 3.8. *Let assumptions 2.4 and 3.6 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

$$M_1 = y_1(\hat{x}^2) \qquad M_2 = y_2(\hat{x}^1)$$

and 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 $\frac{M_i - y_i^{id}}{\gamma_i}$ distinct values, so that the cardinality of the Pareto frontier, obtained as the combination of the two, is finite and at most

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

□

Proposition 3.9. *Let $\hat{y} \in \mathcal{Y}_N$ be a nondominated point of* (BOIP). *Then, problem*

$$\begin{array}{ll} \min & y(x) \\ \text{s.t.} & x \in \mathcal{X} \cap \{y_i(x) \leq \hat{y}_i - \epsilon_i\} \\ & x \in \mathbb{Z}^n \end{array} \tag{5}$$

with $i = 1$ or $i = 2$ is either infeasible or its solutions are efficient for (BOIP).

Proof. If problem (5) is infeasible there is nothing to prove. Let $i = 1$ and let $\bar{x}$ be an efficient solution for (5) (case $i = 2$ can be proven identically). By contradiction, assume that $\bar{x}$ is not efficient for (BOIP). Then, $\tilde{x} \in \mathcal{X} \cap \mathbb{Z}^n$ 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 (5). 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 (5). □

Theorem 3.10. *Let assumptions 2.4 and 3.6 hold. Algorithm 1 returns the complete Pareto frontier $\mathcal{Y}_N$ of* (BOIP) *after having addressed $2|\mathcal{Y}_N| + 1$ scalar objective integer programs.*

Proof. When node $(BOIP)^k$ is addressed in Algorithm 1, the integer program $(INLP)^k$ is built using a correct scalarization technique (see Section 2.1). Then, Problem $(INLP)^k$ is solved: if $(INLP)^k$ is infeasible, we conclude that $\mathcal{X}^k \cap \mathbb{Z}^n$ does not contain any efficient point and the node is pruned. Otherwise, from Proposition 3.9, we have that its solutions are efficient points, giving us a nondominated point $\hat{y}^k$. By Proposition 3.7 and Remark 3.3, the nondominated point $\hat{y}^k \in \mathcal{Y}_N$ cannot be detected again by addressing any subsequent node in the branching tree. Summarizing, whenever a node is addressed and an integer

program is solved, either we get a new nondominated point or we detect infeasibility and we prune the node. Therefore, since Algorithm 1 produces exactly two children for each nondominated point of (BOIP), we have that the branching tree has exactly $2|\mathcal{Y}_N| + 1$ nodes (including the root node) so that exactly $2|\mathcal{Y}_N| + 1$ integer programs will be addressed before finding the complete Pareto frontier. ∎

The total number of iterations of `FPA` is $O(|\mathcal{Y}_N|)$ that is in fact the order of complexity of any algorithm that aims to identify all $|\mathcal{Y}_N|$ nondominated points by solving a sequence of subproblems, as it must solve at least $|\mathcal{Y}_N|$ subproblems. We observe that, for the special case of linear BOIPs, a criterion space search algorithm that solves $2|\mathcal{Y}_N| - 1$ integer programs has been defined in [28]. So, in the linear case, this algorithm based on the augmented weighted Tchebycheff method, has a slightly better bound than our. However, it is not straightforward how to extend the augmented weighted Tchebycheff method to tackle nonlinear BOIPs.

3.3. γ and oracles

In this section we present classes of functions that easily satisfy the Assumptions 3.2 and 3.6.

As a first step we look for sufficient conditions to have $\gamma > 0$ and computable whenever $f(x) \neq f(z)$ for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ (Assumption 3.6).

Proposition 3.11. *Assume that $f : \mathbb{Z}^n \to \mathbb{Z}$. Then we get that $\gamma = 1$.*

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 3.12. *As a matter of example of functions satisfying the condition in Proposition 3.11 we have all the polynomials with integer coefficients and in particular $f(x) = c^\top x$ with $c \in \mathbb{Z}^n$ and $f(x) = x^\top Q x + c^\top x$ with $Q \in \mathbb{Z}^{n \times n}$ and $c \in \mathbb{Z}^n$.*

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

Proposition 3.13. *Let $f(x) = x^\top Q x + c^\top x$. Assume that $Q \in \mathbb{Q}^{n \times n}$ and $c \in \mathbb{Q}^n$, then $r \in \mathbb{N}$, $r \neq 0$ exists so that $\gamma = \frac{1}{r}$.*

Proof. Since $Q \in \mathbb{Q}^{n \times n}$ and $c \in \mathbb{Q}^n$ then $r \in \mathbb{N}$, $r \neq 0$ exists such that $rQ \in \mathbb{Z}^{n \times n}$ and $rc \in \mathbb{Z}^n$. The function $g(x) = x^\top (rQ) x + (rc)^\top x = r f(x)$ satisfies the assumption of Proposition 3.11, therefore we can write

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

or, equivalently,

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

□

Note that r is easily computable as the least common multiple of the denominators of the rational coefficients.

Polynomials in n variables seem to satisfy the condition. Check how to write polynomials and the assertion

Remark 3.14. *Of course Proposition 3.15 holds when Q is the zero matrix, i.e. when $f(x)$ is a rational linear function: $f(x) = c^\top x$, $c \in \mathbb{Q}^n$.*

Proposition 3.15. *Let $f(x) = \|Ax + b\|_2$ and assume that*

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

Assume that $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^n$, then $\gamma = \sqrt{\bar{v} + 1} - \sqrt{\bar{v}}$.

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

$$|f(x) - f(z)| \geq \sqrt{\bar{v} + 1} - \sqrt{\bar{v}}.$$

□

In order to satisfy Assumption 3.2, we need an oracle able to address Problem $(INLP)^k$:

$$\min_{\mathcal{S} \cap \mathbb{Z}^n} y_1(x) + y_2(x)$$

where $\mathcal{S}$ is obtained intersecting the original decision space $\mathcal{X}$ with a finite number of cuts of the type $\{x \in \mathbb{R}^n : y_i(x) \leq \alpha\}$, $i = 1, 2$, $\alpha \in \mathbb{R}$.
In particular, we restrict ourselves to convex objectives $y_i(x)$, so that $\mathcal{S} \subseteq \mathbb{R}^n$ is convex.
In Table 1, we report some classes of objective functions that can be considered when using integer programming solvers such as CPLEX [16], Gurobi [15], SCIP [14], Couenne [] or Bonmin [7], in order to deal with problem $(INLP)^k$. In particular,

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

Table 1: Classes of functions that satisfy Assumptions 3.2 and 3.6. In the table we denote with r the least common multiple of the denominators of the rational coefficients.

$y_i(x) =$	γ	oracle
$c^\top x$ with $c \in \mathbb{Z}^n$	1	*ILP*
$c^\top x$ 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*
$x^\top Q x + c^\top x$ with $Q \succeq 0$, $Q \in \mathbb{Z}^{n \times n}$, $c \in \mathbb{Z}^n$	1	*QCQIP*
$x^\top Q x + c^\top x$ 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*

3.4. Circumventing/Encompassing/Bypassing nonlinear cuts

We introduced the Frontier Partitioner Algorithm as a branch and cut algorithm: new nodes are built imposing cuts to the feasible set in the decision space (see Step 9 in Algorithm 1). More specifically, at a generic node k, the set $\mathcal{X}^k \cap \mathbb{Z}^n$ is intersected with the following set:

$$C = \{x \in \mathbb{R}^n : y_i(x) \leq \hat{y}_i^k - \epsilon_i\}, \tag{6}$$

being $i = 1$ or $i = 2$ and $\hat{y}^k$ the nondominated point found at node k. When $y_i(x)$ is convex nonlinear we are introducing a nonlinear cut, as the set in (6) 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

$$\{x \in \mathbb{Z}^n : Ax \leq b\} = \{x \in \mathbb{Z}^n : y_i(x) \leq \hat{y}_i - \epsilon_i\}. \tag{7}$$

Such a valid formulation can be obtained, at least theoretically, when $\mathcal{X}$ is compact and we deal with convex inequalities: in [10, 11] it is shown that the Chvátal–Gomory closure of a compact convex set is a rational polyhedron. However, from a practical point of view, it is not yet clear how to easily generate a valid formulation.

It is the purpose of this section to investigate on the use of linear inequalities within `FPA`.

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

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

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

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

The resulting `FPA`, may eventually not terminate with the entire Pareto frontier, as we are not guaranteed of cutting the current nondominated point. On the one hand, we loose the exactness of the method, but we have the advantage of dealing with linear constraints.

For the specific class of problems where the objectives are strictly convex quadratic forms, we can prove the following result:

Proposition 3.16. *Let $y_i(x) = x^\top Q x$, with $Q \succ 0$. 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

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

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

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

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

hence we get the result. □

Note that both C^1, C^∞ are defined by $2n$ linear inequalities. Hence in the specific case of problems with strictly convex quadratic forms as objective, one can think of generating $(BOIP)^k$ using these $4n$ linear inequalities. Again, the resulting `FPA` will be a heuristic approach, as we are not guaranteed of cutting the nondominated points found so far. However, we have the advantage of dealing with a quadratic integer programming problem that can be solved quite efficiently with respect to quadratically constrained quadratic integer programming instances (see, e.g., [8]).

4. Numerical results

To test the performance of our algorithm `FPA`, we considered biobjective integer linear instances (see Section 4.1) and biobjective integer quadratic instances (see Section 4.3). Algorithm `FPA` is implemented in Java and use `CPLEX 12.6` as integer programming solver. All our experiments were carried out on

(??)

(Intel Xeon processors running at 2.60 GHz...).

All running times were measured in CPU seconds. All tables presented in this section include the following data for the comparison...

Problem $(INLP)^k$ is built by using the Goal programming method with $q = 1$ (see Section 2.1 and in particular Remark 2.12). This means that at every node of our branch and cut we address the following integer program

$$\min_{x \in \mathcal{X}_i^k \cap \mathbb{Z}^n} y_1(x) + y_2(x), \tag{8}$$

for some $i = 1, 2$.

Confronto da diverse tecniche di scalarizzazione: noi contro di noi. Scelta di quella che performa meglio come tempi e confronto con altri

4.1. Numerical experiments on linear instances

In this section, we present a numerical comparison of algorithm `FPA` with the balanced box method proposed in [3], that in the following is denoted as `BBM`.

description of the instances

We considered in total 219 biobjective integer linear instances: $n1$ instances of the biobjective knapsack problem, $n2$ instances of the biobjective assignment problem and $n3$ instances of biobjective integer linear programming. The instances considered have been used in [3] and are available on `htt:\\...`.
Besides the tables of average running times, we visualize our results comparing the performance of `FPA` and `BBM` by performance profiles, as proposed in [12]. Given our set of solvers $\mathcal{S}$={`FPA`, `BBM`} 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.

4.1.1. Biobjective Knapsack Problem
Let $b \in \mathbb{Z}$ be the capacity of the knapsack, $a_i > 0, a_i \in \mathbb{Z}$ be the weight and $c_{1i}, c_{2i} \in \mathbb{Z}$ be the profits of the i-th object. The biobjective Knapsack Problem can be modeled as follows:

$$\begin{array}{ll} \max & (c_1^\top x,\, c_2^\top x) \\ \text{s.t.} & a^\top x \leq b \\ & x \in \{0,1\}^n. \end{array}$$

In the instances considered we have $c_{ij}, a_j \in [1, 1000]$, with $j = 1, \ldots, n;\ i = 1, 2$ and $b = \lceil 0.5 \sum_{j=1}^{n} a_j \rceil$. Note that the condition in Proposition 3.11 is satisfied, so that $\gamma = 1$.

Table - results with one thread

Performance profiles - results with one thread

4.1.2. Biobjective Assignment Problem
The biobjective assignment problem aims to obtain optimal assignments between a set of agents $j \in \{1, \ldots, n\}$ and a set of tasks $k \in \{1, \ldots, n\}$, where each assignment has non-negative costs c_{1jk}, c_{2jk}.

$$\begin{array}{lll} \min & \left(\sum_{j=1}^{n}\sum_{k=1}^{n} c_{1jk}, \right. & \left. \sum_{j=1}^{n}\sum_{k=1}^{n} c_{2jk}\right) \\ \text{s.t.} & \sum_{k=1}^{n} x_{jk} = 1 & j = 1, \ldots, n \\ & \sum_{j=1}^{n} x_{jk} = 1 & k = 1, \ldots, n \\ & x_{ij} \in \{0,1\}, & i = 1, \ldots, n;\ j = 1, \ldots, n. \end{array}$$

In the instances considered we have $c_i \in \mathbb{Z}^{n\times n}$ and $c_{ijk} \in [1, 20]$, $i = 1, 2$; $j = 1, \ldots, n$; $k = 1, \ldots, n$. Note that the condition in Proposition 3.11 is satisfied, so that $\gamma = 1$.

Table - results with one thread

Performance profiles - results with one thread

4.1.3. Biobjective Integer Linear Programming Problem

The generic biobjective integer linear programming problem is modeled as

$$\begin{aligned} \min \quad & (c_1^\top x,\, c_2^\top x) \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n. \end{aligned}$$

In the instances considered we have $c_i \in \mathbb{Z}^n$, $i = 1, 2$; $A \in \mathbb{Z}^{m\times n}$ 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, $j = 1, \ldots, n$; $i = 1, 2$. 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 and $\sum_{l=1}^{n} a_{kl}$. Note that the condition in Proposition 3.11 is satisfied, so that $\gamma = 1$.

Table - results with one thread

Performance profiles - results with one thread

4.2. Parallelization

It is nowadays common to use computers with multiple cores. This gives the possibility of dividing computations over multiple threads and consequently reducing the running times. Commercial integer programming solvers as `CPLEX` are designed to exploit the availability of multiple threads: in `CPLEX` users can choose the number of threads to use, by properly setting the parameter `CPX_PARAM_THREADS`. In [3], the authors discuss the benefits of using multiple threads and this motivated the following numerical experiment. We set `CPX_PARAM_THREADS` to in both our implementation of `FPA` and in the C++ implementation of `BBM`.

(??)

Table - results with p threads

Performance profiles - results with p threads

4.3. Numerical experiments on quadratic instances

The generic biobjective integer quadratic programming problem is modeled as

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

In the instances considered we have $Q \succ 0$, $Q_i \in \mathbb{Z}^{n\times n}$, $c_i \in \mathbb{Z}^n$, $i = 1, 2$; $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. In particular, Q_{ij} and c_{ij} is generated in the ranges $[-100, -1]$ with probability 0.2 and $[0, 100]$ with probability 0.8, $j = 1, \ldots, n$; $i = 1, 2$. The coefficients a_{kl} are generated in the ranges $[-100, -1]$ with probability 0.1, $[1, 100]$ 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 and $\sum_{l=1}^{n} a_{kl}$. Note that the condition in Proposition 3.11 is satisfied, so that $\gamma = 1$.

description of the instances

Table - results with one thread

5. Conclusions

6. Acknowledgments

The authors acknowledge Prof. Hadi Charkhgard for having kindly provided the code of the balanced box method [3].

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

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

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

[4] N. Boland, H. Charkhgard, and M. Savelsbergh. The l-shape search method for triobjective integer programming. *Mathematical Programming Computation*, 8(2):217–251, 2016.

[5] N. Boland, H. Charkhgard, and M. 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.

[6] N. Boland, H. Charkhgard, and M. 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.

[7] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, et al. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[8] 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.

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

[10] D. Dadush, S. S. Dey, and J. P. Vielma. On the chvátal-gomory closure of a compact convex set. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 130–142. Springer, 2011.

[11] D. Dadush, S. S. Dey, and J. P. Vielma. On the chvátal–gomory closure of a compact convex set. *Mathematical Programming*, 145(1-2):327–348, 2014.

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

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

[14] A. Gleixner, L. Eifler, T. Gally, G. Gamrath, P. Gemander, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, F. Serrano, Y. Shinano, J. M. Viernickel, S. Vigerske, D. Weninger, J. T. Witt, and J. Witzig. The scip optimization suite 5.0. Technical Report 17-61, ZIB, Takustr. 7, 14195 Berlin, 2017.

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

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

[17] D. Jones, M. Tamiz, et al. *Practical goal programming*, volume 141. Springer, 2010.

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

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

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

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

[22] G. Mavrotas and D. 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.

[23] K. Miettinen. Nonlinear multiobjective optimization, volume 12 of international series in operations research and management science, 1999.

[24] S. Moradi, A. Raith, and M. 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.

[25] A. Przybylski, X. Gandibleux, and M. 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.

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

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

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

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

[30] T. Stidsen, K. A. Andersen, and B. Dammann. A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science*, 60(4):1009–1032, 2014.

[31] J. Sylva and A. 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.