European Journal of Operational Research 306 (2023) 286–307

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

New $\epsilon-$constraint methods for multi-objective integer linear programming: A Pareto front representation approach

Mariana Mesquita-Cunha*, José Rui Figueira, Ana Paula Barbosa-Póvoa

CEGIST - Centre for Management Studies, Instituto Superior Técnico, Universidade de Lisboa, Portugal

ARTICLE INFO

Article history:
Received 26 July 2021
Accepted 24 July 2022
Available online 29 July 2022

Keywords:
Multiple objective programming
Integer linear programming
Generation methods
Representation methods

ABSTRACT

Dealing with multi-objective problems by using generation methods has some interesting advantages since it provides the decision-maker with the complete information about the set of non-dominated criterion vectors (Pareto front) and a clear overview of the different trade-offs of the problem. However, providing many solutions to the decision-maker may also be overwhelming. As an alternative approach, showing a representative set of the Pareto front may be advantageous. Choosing such a representative set is by itself also a multi-objective problem that must consider the number of alternatives to present, the uniformity, and/or the coverage of the representation, to guarantee its quality. This paper proposes three algorithms for the representation problem for multi-objective integer linear programming problems with two or more objective functions, each one of them dealing with each dimension of the problem (cardinality, coverage, and uniformity). Such algorithms are all based on the ϵ-constraint approach. In addition, the paper also presents strategies to overcome poor estimations of the Pareto front bounds. The algorithms were tested on the ability to efficiently generate the whole Pareto front or a representation of it. The uniformity and cardinality algorithms proved to be very efficient both on binary and on integer problems, being amongst the best in the literature. Both coverage and uniformity algorithms provide good quality representations on their targeted objective, while the cardinality algorithm appears to be the most flexible, privileging uniformity for lower cardinality representations and coverage on higher cardinality.

1. Introduction

Most real-world problems, although being multi-objective by their very nature, are frequently modelled as single objective problems mainly due to the lack of multi-objective tools and/or the computational complexity of solving multi-objective models. However, the manipulation of objectives in order to amalgamate them into a single objective function tends to oversimplify the problem and the conflicting nature among objectives. Indeed, a single objective model may not represent appropriately the decision-maker's (DM) real preferences and objectives (Branke, Deb, Miettinen, & Slowiński, 2008). The aforementioned disadvantages together with the rapid rise of computational performance, both in terms of hardware and software, are making multi-objective models, and in particular Multi-Objective Optimization (MOO), more favoured approaches. Nevertheless, multi-objective integer and mixed integer linear programming (MOILP and MOMILP), which arise in many real-world applications (such as logistics and production problems), have received less attention when compared to binary and continuous problems (Alves & Clímaco, 2007).

Depending on the stage at which the DM is involved to ascertain his/her preferences, multi-objective methods can be classified into *a priori* methods, interactive methods, and *a posteriori*/generation methods. In *a priori* methods, the DM states her/his preferences at the begining and the problem is then solved by aggregating the objective functions into a single one. This approach, not only carries many of the disadvantages presented above for the single objective models but also, most of the time, in complex functions, the DM struggles to provide parameters as, for example, the weights for each objective. In interactive methods, the DM iteratively states and adjusts preferences based on the results previously obtained. The drawback with this approach is two-folded: (1) it may take a long time for a satisfactory trade-off between objectives to be found; and (2) the DM, by never seeing the Pareto front, also never clearly understands the relation between objectives and their impact on the solutions. *A posteriori* or generation methods, which involve the DM only after finding the Pareto front, are the most advantageous. This approach clearly allows for defining the trade-off between all objectives, providing the DM with a full view

* Corresponding author.
E-mail address: mariana.cunha@tecnico.ulisboa.pt (M. Mesquita-Cunha).

https://doi.org/10.1016/j.ejor.2022.07.044

of the problem and all the tools for making a better-informed decision. However, not only have they become more computationally heavy, but they also risk overwhelming the DM with too many solutions to analyse.

The *a posteriori* methods frequently require significant computational effort and time to compute the efficient/non-dominated set. Hence, apart from usually small and linear problems, the non-dominated set is not practical to compute in a reasonable amount of time. As a result, there are two classes of generation methods, one that aims to determine the set of non-dominated criterion vectors (the so-called Pareto front), and the other which focuses on obtaining a set of points representative of the non-dominated set without being overwhelming for the DM (Alves & Clímaco, 2007; Kidd, Lusby, & Larsen, 2020). To address the latter, the representation problem must be considered, which, in itself is a multi-objective problem with three objective functions, namely (1) cardinality, the number of alternatives presented to the DM; (2) coverage, how well the Pareto front is being represented by the set of solutions; and (3) uniformity, how well those alternatives are spread through the Pareto front (Sayın, 2000).

In this work, we propose three multi-objective *a posteriori* algorithms to solve MOILP/ MOMILP with more than two objectives with strategies to overcome poor quality approximations for the bounds of the Pareto front, that can be used to generate the whole Pareto front or to compute a representation of it. To this end, the algorithms we put forward tackle each of the three objective functions of the representation problem.

Using a generation method on a MOILP/MOMILP is often more challenging than on a multi-objective linear program since the former has a non-convex feasible region, which implies that there may exist unsupported non-dominated criterion vectors. Chalmet, Lemonidis, and Elzinga (1986) proposed a modified weighted-sum method to produce, apart from the set of supported non-dominated criterion vectors, the set of unsupported non-dominated criterion vectors. This was achieved by adding constraints to the problem that bound the values of the objective functions. Although all non-dominated points can be obtained with a full parametrization of the weights added to each objective function, it can lead to an extensive and computationally demanding optimization problem.

Several other authors also proposed approaches based on a sequential reduction of the feasible region (Klein & Hannan, 1982; Masin & Bukchin, 2007; Sylva & Crema, 2004; 2007). Klein and Hannan (1982) propose a reduction of the feasible region by sequentially adding constraints that eliminate points dominated by the previously found non-dominated criterion vector. This strategy finds points spaced by at least a fixed amount (provided by a parameter), for each objective. However, it may skip interesting solutions from the DM's preferences point of view, since only one of the objectives is considered as the objective function to be optimized. Sylva and Crema (2004) presented a variation of Klein and Hannan (1982)'s method where all the objectives are included in the objective function by using weighting parameters. Sylva and Crema (2007) build upon the Sylva and Crema (2004) method by determining, at each iteration, the weights that maximize the infinity-norm distance to the set dominated by the previously found solutions. Consequently, uniformly spaced points are found. Masin and Bukchin (2007) proposed a very similar approach. All of these methods share the same main drawback, namely the fact that the problem size increases whenever a non-dominated criterion vectors is found, since the new constraints and, in some cases, new variables have to be added to the model. Hence, updating the search region whenever a new non-dominated criterion vectors is obtained is very important (Ceyhan, Köksalan, & Lokman, 2019; Dächert, Klamroth, Lacour, & Vanderpooten, 2017; Klamroth, Lacour, & Vanderpooten, 2015). Although focused on computing the whole Pareto front, Klamroth et al. (2015) present two strategies to incrementally update the problem's search region on this type of generation methods, redundancy elimination and redundancy avoidance, considering the geometrical properties of the search region and decreasing the number of redundant computations. Dächert et al. (2017) use the neighbourhood relation between local bounds of the search region to update it, providing a significantly more efficient strategy than the ones proposed in Klamroth et al. (2015). Ceyhan et al. (2019) and Doğan, Lokman, and Köksalan (2022) propose algorithms to generate representations that guarantee a predefined coverage error. To that end, Ceyhan et al. (2019) present a first algorithm (SBA) that partitions the search space into subsets and searches for the worst-represented points in each of these subsets, retaining the one with maximum coverage gap; a second algorithm (TDA), which eliminates from the search region the space around each non-dominated criterion vector corresponding to the desired coverage gap; and a third algorithm (SPA) which requires a desired cardinality from the DM and, by approximating the Pareto front to an L_p surface, predicts the location of the points providing the best coverage representation. However, while the second algorithm, TDA, requires the DM to provide a desired coverage threshold value, which is often very difficult, the third algorithm, SPA, only provides better results than the first algorithm on the lowest cardinalities, falling short in terms of coverage gap on higher cardinalities. Doğan et al. (2022) take as a baseline algorithm TDA and combine it with Dächert et al. (2017) search region update strategy, proposing algorithm TSGA. However, since these algorithms are sensitive to the scalarization model's weight parameters, Doğan et al. (2022) determine the model's weights by fitting an L_p surface and using an hyperplane tangent to the surface on the point that is at minimum Chebyshev distance from the ideal point. TSGA proved more efficient than SBA and TDA, proposed by Ceyhan et al. (2019). Kidd et al. (2020) also present a scalarization algorithm targeted for generating representations for bi-objective problems. To that end, an insertion method based on Voronoi cuts to partition the search region was developed. This method was proven to achieve simultaneously good coverage and uniformity for a given cardinality. The algorithm continues to insert solutions until a desired cardinality level is reached.

One of the most popular methods for solving *a posteriori* MOILP problems is the ϵ-constraint method. For constraining each objective, the method resorts to a virtual grid spaced, for each objective, by making use of an ϵ parameter. Laumanns, Thiele, and Zitzler (2006) applied the ϵ-constraint method as a generation method by dividing the objective space, after each iteration, using the values of the previous computed solution to create search boxes in the objective space. Additionally, after each new solution is found, a lexicographic optimization is performed to ensure getting non-dominated criterion vectors. Kirlik and Sayın (2014) developed a method based on the Laumanns et al. (2006) works, which makes use of a two stage model in order to guarantee non-dominated criterion vectors. It is a search approach based on the construction of rectangles, as in Laumanns et al. (2006), but it also removes rectangles in which no non-dominated criterion vector can be found. This approach makes the search less exhaustive when compared to the Laumanns et al. (2006) algorithm. Mavrotas (2009) adapted the original ϵ-constraint method by introducing slack variables in equality constraints and incorporating them in the objective function weighted by a factor which includes the range of each objective. To choose the ϵ vector, the range of each objective is divided into evenly spaced intervals generating a uniform grid. However, this approach produces several redundant points. To overcome this, Mavrotas and Florios (2013) extended the Mavrotas (2009) method by exploring the values of the slack variables in order to skip redundant iterations, improving computational efficiency. Zhang

and Reimann (2014) addressed the requirement to have the true nadir values, which are often difficult to compute (see Alves & Costa, 2009; Ehrgott & Ryan, 2003; Kirlik & Sayın, 2015). The proposed methodology skips the redundant iterations in a way that does not require more computations when using approximated nadir values. The disadvantage, however, is that it can only be used for the computation of the whole Pareto front and not for the generation of a representation. Ozlen, Burton, and MacRae (2014) proposed an adaption to a previously developed ϵ-constraint based algorithm, proposed in Özlen and Azizoğlu (2009), integrating information provided by the computed points, which allowed to skip redundant iterations. However, comparative analysis, reported in Zhang and Reimann (2014), proved the methodology by Zhang and Reimann (2014) to be more efficient than the proposed by Ozlen et al. (2014). Nikas, Fountoulakis, Forouli, and Doukas (2020) also address Mavrotas and Florios (2013) drawbacks but these authors present an algorithm that requires an array with as many entries as the integer size of the objective functions' ranges, which can generate memory issues. Furthermore, the presented results report dominated criterion vectors, requiring the obtained points to be filtered. Despite the fact that both Mavrotas and Florios (2013) and Nikas et al. (2020) allow for the computation of Pareto front representations, their quality is not addressed on both works.

Eusébio, Figueira, and Ehrgott (2014) address the representation problem using the ϵ-constraint method for bi-objective problems. Eusébio et al. (2014) propose two algorithms, one for coverage and another for uniformity. The former is an insertion based method: at each iteration, the algorithm searches for a criterion vector between the two most distant in the representation. The latter successively adds criterion vectors spaced by a predetermined step to the representation. Both methods terminate when the desired level of coverage and uniformity are met.

In this work, we take as a baseline Mavrotas and Florios (2013) and combine it with the strategies presented in Zhang and Reimann (2014), which allow, without having to calculate the whole Pareto front, to pass over redundant iterations and also permit to overcome the need of computing the true nadir points. In this way, we can address the aforementioned drawbacks of both works, while keeping their advantages. Furthermore, we develop three search strategy algorithms, one for coverage, another for uniformity, and the third one for cardinality, addressing each dimension of the Pareto front representation problem. The first two algorithms are based on the work by Eusébio et al. (2014) and are extended for MOILP problems with more than two objectives. The latter refines the virtual grid, introduced by Mavrotas (2009), after finding each solution, trying to verify the cardinality level by focusing the search on the feasible region. This work, by the very nature of its model formulation and strategy to look for new criterion vectors in the Pareto front, is independent of both the number of non-dominated solutions found and the objective functions ranges.

The remainder of this paper is organized as follows. Section 2 introduces the mathematical background, namely the type of problem addressed in this paper, the model used and the representation problem. Section 3 presents the proposed methodology, both the generic algorithm and the three proposed search strategies, as well as an illustrative example for each one of them. Section 4 provides the computational results for the strategies presented in Section 3, both the ones for computing the Pareto front as well as those for the representation problem. At last, Section 5 presents some concluding remarks and future work lines are put forward.

2. Mathematical background

This section presents the main concepts, definitions, and notation on multi-objective optimization, the $\epsilon-$constrain approach, and some fundamental concepts related to the representation of the whole set of criterion vectors.

2.1. The multi-objective integer programming problem

Consider the following multi-objective integer linear programming model.

$$\begin{array}{rcl} \max z_1(x) & = & (c^1)^\top x \\ \vdots & & \vdots \\ \max z_k(x) & = & (c^k)^\top x \\ \vdots & & \vdots \\ \max z_p(x) & = & (c^p)^\top x \\ \text{subject to:} & & x \in X. \end{array} \tag{P1}$$

where $x = (x_1, \ldots, x_j, \ldots, x_n)$ is an $n-$vector of non-negative and integer *decision variables*, $(c^k)^\top = (c^k_1, \ldots, c^k_j, \ldots, c^k_n)$ is an $n-$row vector composed of the *coefficients* of the decision variables in the *objective functions*, $k = 1, \ldots, p$ (we assume these coefficients are integer values or can be converted into integer), and X is the *feasible region* in the *decision space*, $\mathbb{Z}^n_{0+}$ (the set of non-negative integers). Let $Z = z(X)$ denote the image of the *feasible region* according to the objective functions. The set Z is called the *feasible region* in the *objective space*, i.e., $\mathbb{Z}^p$ (the set of integers) along with the order relation imposed by the objective functions in this set. Furthermore, assume the feasible region in the decision space is both bounded and not empty.

Problem 1 can be presented in a more compact form as follows.

$$\begin{array}{rcl} \text{“max”} z(x) & = & Cx \\ \text{subject to:} & & x \in X. \end{array} \tag{P2}$$

where “max” means that all functions are to be maximized, $z(x) = (z_1(x), \ldots, z_k(x), \ldots, z_p(x))$ is the vector of the p objective functions, and C is an $p \times n$ matrix, each row being composed of the coefficients of each objective function.

Definition 1. (Dominance) Let z' and z'' denote two criterion vectors, or points, in the objective space. Then, vector z' *dominates* z'', iff $z' \geq z''$ with $z' \neq z''$ (i.e., $z'_k \geqslant z''_k$, with at least a strict inequality, for $k = 1, \ldots, p$).

Definition 2. (Strict dominance) Let z' and z'' denote two criterion vectors, or points, in the objective space. Then, vector z' *strictly dominates* z'', iff $z' > z''$ with $z' \neq z''$ (i.e., $z'_k > z''_k$ for $k = 1, \ldots, p$).

Let z^* denote a feasible criterion vector, i.e., $z^* \in Z$. This vector is a *weakly non-dominated criterion vector* if and only if there is no other $z \in Z$ such that z strictly dominates z^*. One weakly non-dominated criterion vector is more interesting than another if its performance is better in at least one objective function. If there is no vector $z \in Z$, such that z dominates z^*, vector z^* is designated as a *non-dominated criterion vector*. Whenever z^* can be obtained as a weighted-sum of the p objective functions with strictly positive weighting factors, z^* is called a *supported criterion vector*. Otherwise, z^* is said to be an *unsupported criterion vector*. $N(Z)$ denotes the set of non-dominated criterion vectors, also called the *Pareto front*.

The same kind of concepts can be applied in the decision space. A feasible solution x^* is called an *efficient solution* if and only if we cannot find another $x \in X$ such that $Cx \geq Cx^*$ and $Cx \neq Cx^*$. A feasible solution x^* is a *weakly efficient solution* if and only if

we cannot find another $x \in X$ such that $Cx > Cx^*$. Furthermore, for any solution x^* corresponding to an inverse image of a *supported* (*unsupported*) criterion vector z^* is called an efficient *supported* (*unsupported*) solution. Finally, $E(X)$ denotes the set of all efficient solutions. For more details about these definitions see Ehrgott (2005) and Steuer (1986).

2.2. The $\epsilon-$constraint approach

This subsection is devoted to presenting an approach for identifying the Pareto front, based on the resolution of a sequence of problems of the following form.

$$\begin{aligned} \max_{x} \quad & \{z_q(x)\} \\ \text{subject to:} \quad & z_k(x) \geqslant \epsilon_k, \ k = 1, \ldots, p, \ k \neq q \\ & x \in X. \end{aligned} \tag{P3}$$

Theorem 1. *(Haimes, Lasdon, & Wismer, 1971) If x^* is an optimal solution of Problem 3, for some q, then x^* is a weakly efficient solution of Problem 1.*

Proof. See Chankong and Haines (2008), Ehrgott (2005), or Haimes et al. (1971). □

Algorithm 1 can be implemented for solving a sequence of Problem 3 versions. This algorithm requires as inputs Problem 2 data, namely matrix C and the feasible region X, and a small enough strictly positive parameter value, η. It provides as output $N(Z)$, the Pareto front or, if unable to compute all efficient solution, part of it. This algorithm makes use of four internal procedures:

Algorithm 1: Computing the Pareto front, $(N(Z))$.

```
Input: C, X, η;
Output: N(Z);
N̂(Z) ← {};
z* ← Ideal(P);
z^nad ← ApproxNadir(P);
Select q = 1 and build Problem P3;
ε_h ← z_h^nad, for h = 2, ..., p;
while (ẑ_2 < z*_2 and X ≠ ∅) do
    while (ẑ_3 < z*_3 and X ≠ ∅) do
        ...
        while (ẑ_{p-1} < z*_{p-1} and X ≠ ∅) do
            while (ẑ_p < z*_p and X ≠ ∅) do
                ẑ ← Solve(P3);
                N̂(Z) ← N̂(Z) ∪ {ẑ};
                ε_p ← ẑ_p + η;
            ε_{p-1} ← ε_{p-1} + η;
            ε_p ← z_p^nad;
        ...
        ε_3 ← ε_3 + η;
        ε_4 ← z_4^nad;
    ε_2 ← ε_2 + η;
    ε_3 ← z_3^nad;
N(Z) ← Filter(N̂(Z));
return(N(Z));
```

1. $Ideal(P)$: which computes the ideal point, $z^* = (z_1^*, \ldots, z_k^*, \ldots, z_p^*)$, through an individual maximization of each objective function.
2. $ApproxNadir(P)$: which computes the nadir point or an approximation of it, $z^{nad} = (z_1^{nad}, \ldots, z_k^{nad}, \ldots, z_p^{nad})$.
3. $Solve(P^\epsilon)$: which makes use of an integer linear programming solver to solve P^ϵ, and provides the criterion vector $\bar{z} = (\bar{z}_1, \ldots, \bar{z}_k, \ldots, \bar{z}_p)$.
4. $Filter(\hat{N}(Z))$: which makes the filtering of an auxiliary set, $\hat{N}(Z)$, which may contain weakly non-dominated criterion vectors, and provides as output the Pareto front, $N(Z)$.
5. From the first two calculations we identify the ranges of possible values, for each objective function, which are bounded from below by z_k^{nad} and from above by z_k^*, for $k = 1, \ldots, p$.
6. In each *while* loop the ϵ vector bounding the objective functions is updated for the following iteration.

The $\epsilon-$constraint approach consists of solving a sequence of versions of Problem 3 by successive increments or decrements of parameters ϵ_k, for $k = 1, \ldots, p$ with $k \neq q$. Any basic algorithm designed for implementing this approach (as for example Algorithm 1) suffers from three major drawbacks, independently of the solver used for optimizing a variation of Problem 3:

1. Due to the model structure, unnecessary versions of Problem 3 are solved, since it may find some weakly efficient solutions with no interest (i.e., disposable weakly efficient solutions). This is due to the shape of the model objective function and appears as a conclusion of Theorem 1.
2. Additionally, due to the model structure, a large amount of processing time may be required to compute the Pareto front (Ehrgott & Ryan, 2003). This is due to the lack of flexibility of the ϵ constraints (Ehrgott & Ruzika, 2008; Ehrgott & Ryan, 2003).
3. Finally, due to the difficulty of adjusting the ϵ vector, weakly efficient solutions of interest may be missed (Laumanns et al., 2006).

The third drawback was addressed by Laumanns et al. (2006). These authors show in detail the serious issues related to the *a priori* technical adjustment of the ϵ vector, and propose an adaptive based algorithm with an interesting scheme for determining the adequate values for the ϵ vector, when sequentially solving each variation of Problem 3.

The second drawback was reported and first addressed by Ehrgott and Ryan (2003). These authors proposed an elastic technique for changing the nature of ϵ type constraints. As a consequence, the following model was proposed (some improved versions of this model can be seen in Ehrgott & Ruzika 2008).

$$\begin{aligned} \max_{x, e^-, e^+} \quad & \left\{ z_q(x) - \sum_{k=1, k\neq q}^{p} p_k e_k^- \right\} \\ \text{subject to:} \quad & z_k(x) + e_k^- - e_k^+ = \epsilon_k, \ k = 1, \ldots, p, \ k \neq q \\ & e_k^-, e_k^+ \geqslant 0, \ k = 1, \ldots, p, \ k \neq q \\ & x \in X. \end{aligned} \tag{P4}$$

The following result did not change the nature of the output, persisting the first drawback.

Theorem 2. *(Ehrgott & Ryan (2003)) If $p_k > 0$, for $k = 1, \ldots, p$ with $k \neq p$, and (x^*, e^{-*}, e^{+*}) is an optimal solution of Problem 4, for some q, then x^* is a weakly efficient solution of Problem 1.*

Proof. See Ehrgott and Ryan (2003), or Ehrgott and Ruzika (2008) □

The first drawback was addressed by Mavrotas (2009) guaranteeing the efficiency of the obtained solution with the proposal of a new model based on the introduction of slack variables. Mavrotas and Florios (2013) proposed a new improvement for multi-objective integer linear programming which makes a slight change on the objective function. This change will not alter the theoretical results presented in Mavrotas (2009). The improved

model can be stated as follows.

$$\begin{aligned} \max_{x,s} \quad & \left\{ z_q(x) + \rho \sum_{k=1,\, k\neq q}^{p} 10^{k-1} \frac{s_k}{r_k} \right\} \\ \text{subject to:} \quad & z_k(x) - s_k = \epsilon_k, \ k = 1, \ldots, p, \ k \neq q \\ & s_k \geqslant 0, \ k = 1, \ldots, p, \ k \neq q \\ & x \in X. \end{aligned} \tag{P5}$$

where r_k is the width of the range values for each objective function, $k = 1, \ldots, p,\ k \neq q$.

The result of solving a version of the previous problem obeys to the following theoretical result.

Theorem 3. *(Mavrotas 2009) If (x^*, s^*) is an optimal solution of Problem 5, for some q and for ρ corresponding to a sufficiently small number (usually between 10^{-3} and 10^{-6}), then x^* is an efficient solution of Problem 1.*

Proof. See Mavrotas (2009). □

Problem 5 will be used in our algorithms.

2.3. On the representation of the Pareto front

A discrete representation of the Pareto front, $R(N)$, is a finite subset of the Pareto front, $N(Z)$. Many studies have focused on the quality of representations, in terms of how well the subset captures the characteristics of the full set (see Audet, Bigeon, Cartier, Le Digabel, & Salomon, 2021; Faulkenberg & Wiecek, 2010; Sayın, 2000). These studies proposed dimensions of interest in their evaluation of the representation. Sayın (2000) proposes three criteria:

1. *Coverage*: How well does the representation cover all regions of the objective space Z included in $N(Z)$.
2. *Uniformity*: How diverse and equally spaced are the points in the representation, i.e., how spread are the points (criterion vectors) included in the representation.
3. *Cardinality*: The number of criterion vectors considered in the representation, $\Pi(R(N))$.

Coverage and uniformity are dependent on the distance between points in the representation. There are multiple distance metrics, the most common being as follows:

$$d(z, z') = \begin{cases} \left(|z_1 - z'_1|^t + \cdots + |z_k - z'_k|^t + \cdots + |z_p - z'_p|^t \right)^{1/t}, & t \geqslant 1, \\ \max \left(|z_1 - z'_1|, \ldots, |z_k - z'_k|, \ldots, |z_p - z'_p| \right), & t = \infty. \end{cases}$$

where, if $t = 1$ the distance metric used is the Manhattan distance, if $t = 2$ it corresponds to the Euclidean distance, and if $t = \infty$ the Chebyshev distance is applied. The higher the t value, the smaller the distance value measured. Considering a distance metric, the coverage error of the representation can be stated as follows:

$$\Gamma\big(R(N), N(Z)\big) = \max_{z \in N(Z)} \min_{z' \in R(N)} d(z, z'), \tag{1}$$

which corresponds to the maximum distance from $z \in N(Z)$ to its closest point in the representation $z' \in R(N)$. For the sake of simplicity, from hereinafter, $\Gamma(R(N))$ is used instead of $\Gamma(R(N), N(Z))$. A representation $R(N)$ is said to have coverage γ if $\Gamma(R(N)) \leqslant \gamma$. To increase coverage, the coverage error $\Gamma(R(N))$ must be minimized. Since the representation $R(N)$ is a finite subset of $N(Z)$, minimizing the maximum distance from $z \in N(Z)$ to its closest point in the representation $z' \in R(N)$ can be done, for bi-objective problems, by minimizing the maximum distance between every two consecutive points in the representation. Furthermore, by guaranteeing that all consecutive points in the representation $R(N)$ are distanced by a value lower or equal to γ, there is also the guarantee that $R(N)$ has, at most, a coverage error of γ.

The uniformity level corresponds to the minimum distance between any two points in the representation:

$$\Delta\big(R(N)\big) = \min_{z, z' \in R(N), z \neq z'} d(z, z') \tag{2}$$

A representation $R(N)$ is said to have uniformity δ if $\Delta(R(N)) \geqslant \delta$. To increase uniformity, $\Delta(R(N))$ should be maximized. As a consequence, the discrete representation Problem 6, can be stated as a three-objective optimization problem (Shao & Ehrgott, 2016):

$$\begin{aligned} & \min\ \Pi\big(R(N)\big) \\ & \max\ \Delta\big(R(N)\big) \\ & \min\ \Gamma\big(R(N)\big) \\ & \text{subject to: } R(N) \subset N(Z), 2 \leqslant |R(N)| < \infty \end{aligned} \tag{P6}$$

The objectives considered in Problem 6 are interrelated. Sayın (2000) noted that, by improving the representation coverage, cardinality would also increase. As uniformity increases, potentially, the opposite effect on cardinality may emerge. Furthermore, Kidd et al. (2020) proved that, for bi-objective problems, when comparing two representations, $R'(N)$ and $R''(Z)$, with the same cardinality, where $R''(Z)$ is an equidistant representation, then $\Gamma(R''(Z)) \leqslant \Gamma(R'(N))$ and $\Delta(R''(Z)) \geqslant \Delta(R'(N))$.

In this work, we make use of the Chebyshev norm to compute $\Gamma(R(N))$ and $\Delta(R(N))$. Although other metrics may be used, the choice is supported by three main aspects: (1) Chebyshev norm allows the decoupling of the distance for each criterion, in the sense that all coordinate distances will be at most the Chebyshev distance, i.e., if $|z_k - z'_k| \leqslant 1$ for all $k = 1, \ldots, p$, then $d_{l\infty}(z, z') \leqslant 1$; (2) From the previous fact, it also derives that both the coverage error and the uniformity level have a direct correspondence to the coordinate distances without requiring the combination of multiple coordinates; this allows greater control over the representation criteria; and (3) The combination of coordinates, a fundamental characteristic of the other metrics, may distort the distance metric, and consequently the coverage error and uniformity level, if the ranges of each criterion are significantly different (Sayın, 2000). Using the Chebyshev norm implies that, in more than two objectives, the minimization of the coverage error by minimizing the maximum distance between each two consecutive points in the representation can be decoupled for each criterion. Hence, by guaranteeing that in all criteria consecutive points are distanced by a value lower or equal to γ, there is also the guarantee that the coverage error of the $R(N)$ is, at most, γ.

3. Representation of the Pareto front

This section presents the three Grid Point Based Algorithms (GPBA) developed by us to generate a representation of the Pareto front for MOILP Problem 1: *GPBA-A*, *GPBA-B* and *GPBA-C*. Each algorithm targets one dimension of the discrete representation Problem 6 (coverage, uniformity, and cardinality) with different strategies for exploring the feasible region. The first two algorithms, *GPBA-A* and *GPBA-B*, are an extension of the ones proposed by Eusébio et al. (2014), aiming to improve coverage and uniformity, respectively. The third algorithm, *GPBA-C*, represents an improvement over Mavrotas and Florios (2013) algorithm, and concerns mostly cardinality. However, all algorithms solving Problem 5 proposed in this work share the same main procedure, detailed in Fig. 1.

1. The first step of the algorithm consists of the computation of the lower (pessimistic) and upper (optimistic) bounds of the Pareto front, respectively z^{nad} a and z^*. Although the optimistic value is typically easily computed by maximizing each objective

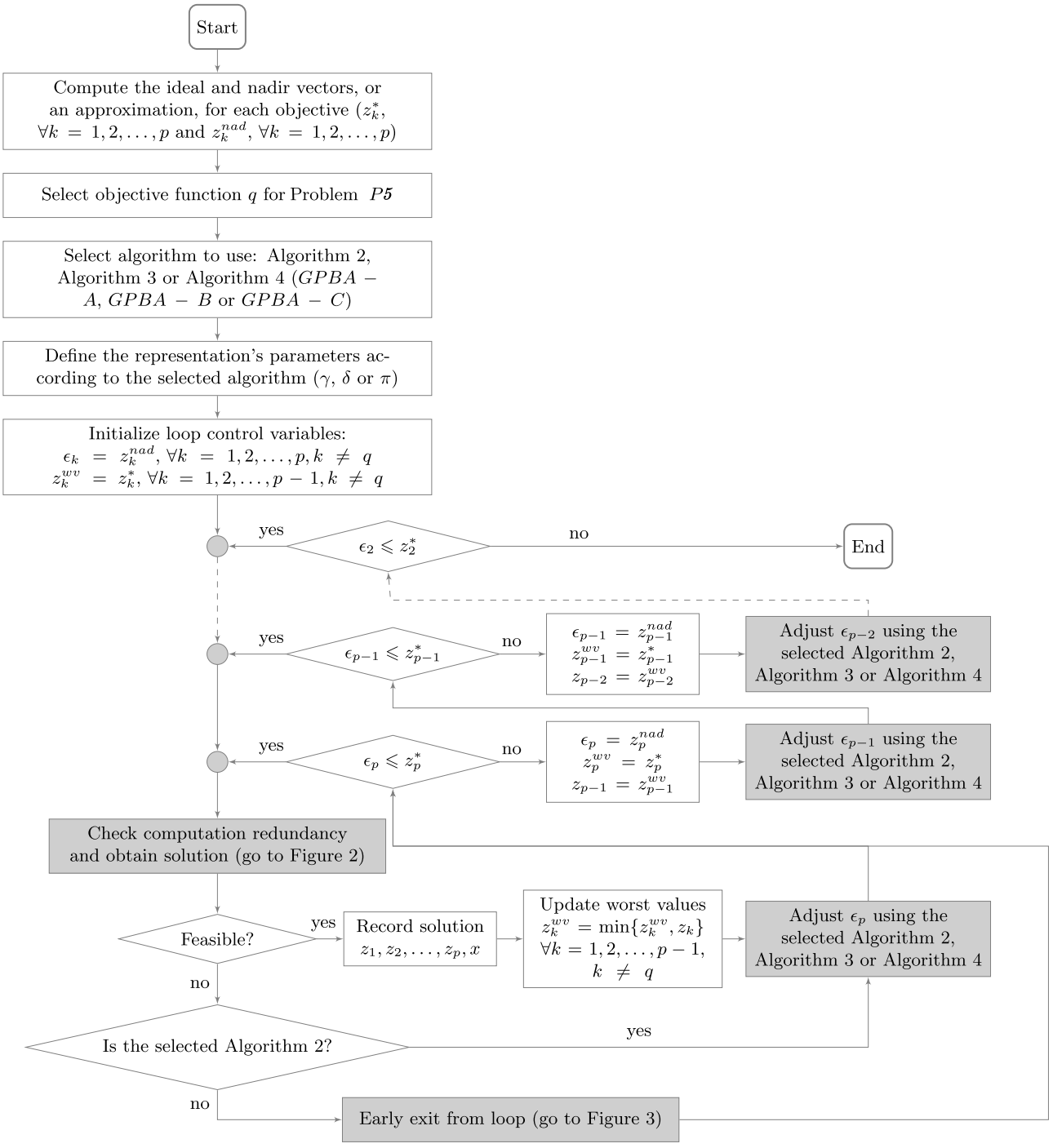


Fig. 1. Flowchart for the generation algorithm.

function separately, the inverse cannot always be done to obtain the pessimistic values. Many studies in the literature show the difficulty of obtaining good estimations for those bounds (see Alves & Costa, 2009; Isermann & Steuer, 1988). Consequently, nadir point overestimation is common. All three proposed methods can handle overestimated bounds, allowing any strategy to compute the pessimistic values for each objective function.

2. The objective function z_q to be directly considered in Problem 5 is then chosen, as well as the representation algorithm (*GPBA-A*, *GPBA-B* or *GPBA-C*). Depending on the representation algorithm, the desired coverage error, uniformity level or the maximum cardinality of the representation must be chosen.
3. The third step consists of the initialization of the main loop by enforcing the parameters ϵ_k, in Problem 5, to be the same as the pessimistic values for each objective with a small perturbation, resulting in Problem 5 being in its most relaxed form. Additionally, the relative worst values for each objective function, z_k^{wv}, are also initialized as the ideal values. z^{wv} is a vector that stores, for each objective function z_k, the worst value for that objective function obtained on the current iteration over the objective function's loop. That value is used as the result of that iteration, for objective function z_k, when adjusting the corresponding ϵ parameter.
4. The fourth step consists of the comparison of the problem's parameters against the results of the previously solved problem,

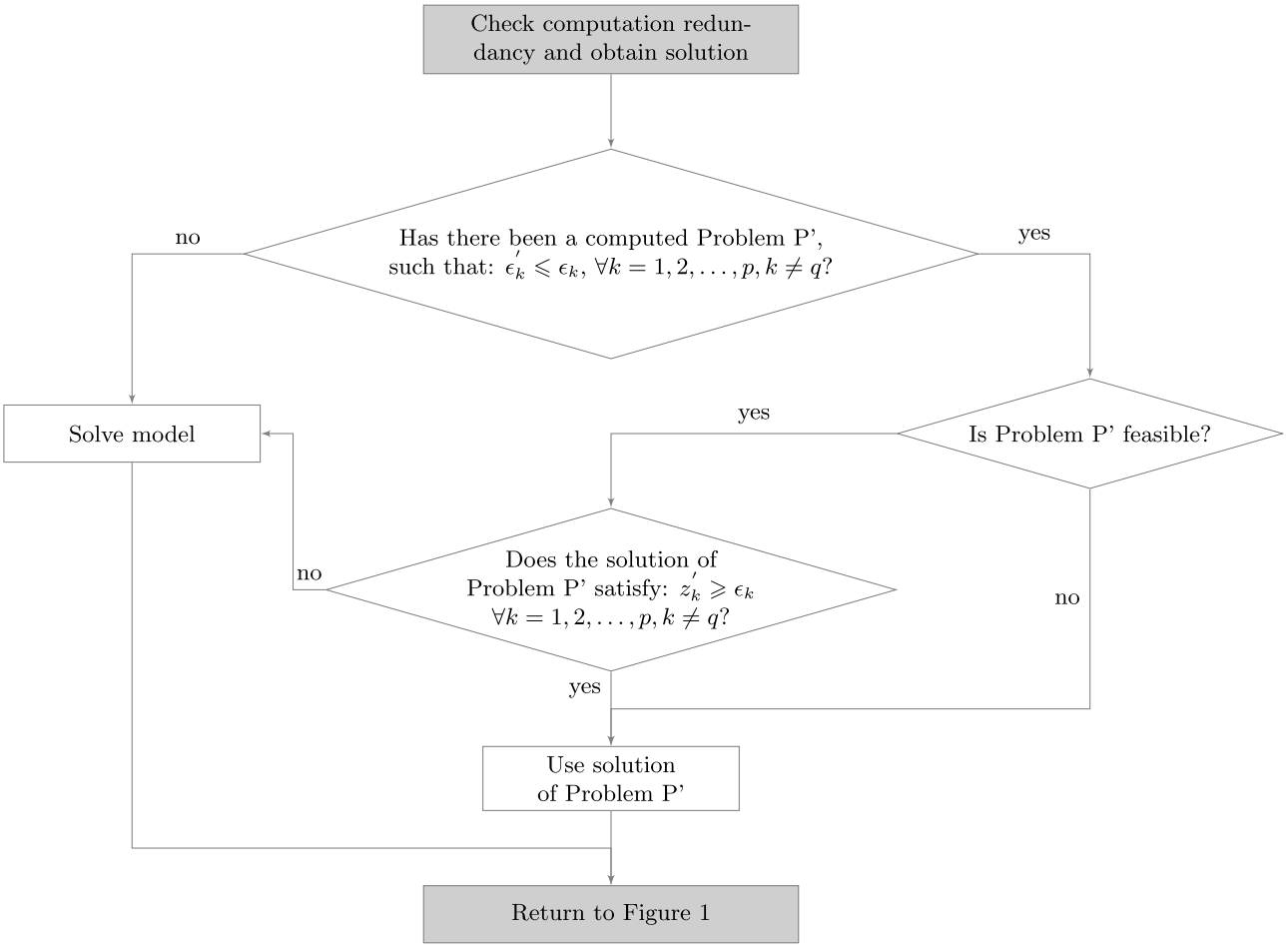


Fig. 2. Process to determine if the problem needs to be solved or can be skipped.

in order to check if a new criterion vector needs to be computed or if it is going to yield a redundant result. That procedure is shown in Fig. 2:

(a) Select from the list of previous iterations, for each $k = 1, \ldots, p,\ k \neq q$, a Problem P' such that the ϵ' vector is equal to ϵ or closer to the nadir point, i.e., $z_k^{nad} \leqslant \epsilon_k' \leqslant \epsilon_k$, for all $k = 1, \ldots, p,\ k \neq q$.
(b) If such a problem does not exist or, if it exists, its solution does not lay in the bounds of Problem 5 when using ϵ, then solve Problem 5 and return to the main procedure (see Fig. 1).
(c) If such a Problem P' exists but it is marked as infeasible, then return to the main procedure (see Fig. 1), considering the problem as an infeasible one.
(d) If such a Problem P' exists and it is marked as feasible, and its solution is within the bounds of Problem 5, when using ϵ, i.e., $z_k' \geqslant \epsilon_k$, for all $k = 1, \ldots, p,\ k \neq q$, then return to the main procedure (see Fig. 1) considering the solution of Problem P' as the solution of Problem 5.

5. If the obtained (or selected from the set of previously computed solutions) criterion vector is feasible, such a vector is saved and the new z_k^{wv} updated for all objective functions, with the exception of the function z_q and the innermost loop p. Then, the new ϵ_p parameter is readjusted according to the algorithm used. Otherwise, and in case the representation algorithm is not *GPBA-A*, i.e. not the coverage representation, the early exit loop is used (see Fig. 3).
6. The process is repeated until the optimistic values is reached in all loops, i.e., until Problem 5 reaches its most constrained form.

The aforementioned acceleration strategies, the redundancy checking (Fig. 2) and the early exit from loop (Fig. 3) procedures were originally proposed by Zhang and Reimann (2014). While the latter is only based on Proposition 1, the former resorts to both Proposition 1 and Proposition 2. Let $LP^{\epsilon s}$ denote a linear relaxation of Problem 5:

Proposition 1. *(Infeasibility) If Problem $LP^{\epsilon s}$ is infeasible, then Problem 5 is also infeasible.*

Proposition 2. *(Optimality) If x^* is the optimal solution of Problem $LP^{\epsilon s}$, and x^* is in the feasible region of Problem 5, then x^* is also the optimal solution of Problem 5.*

In the following subsections, we will detail the algorithms to compute the parameters ϵ_k, for Problem 5.

3.1. Coverage grid point based representation (GPBA-A)

When looking for a representation that privileges the coverage of the Pareto front, the algorithm will aim to represent all areas of the Pareto front. This may be achieved, as discussed in Section 2.3, by minimizing the maximum distance between any two consecutive points in the representation ($\Gamma(R(N))$), Eq. (1), i.e. the coverage error. The quality of the representation is controlled by the parameter γ, corresponding to the acceptable coverage error.

Parameter γ is sometimes difficult to determine. An alternative to directly conveying the acceptable coverage error is to define an acceptable cardinality in each objective (π_k, for all $k = 1, \ldots, p,\ k \neq q$) based on their range ($z_k^* - z_k^{nad}$, for all $k = 1, \ldots, p,\ k \neq q$). The ratio between the range and cardinality provides the

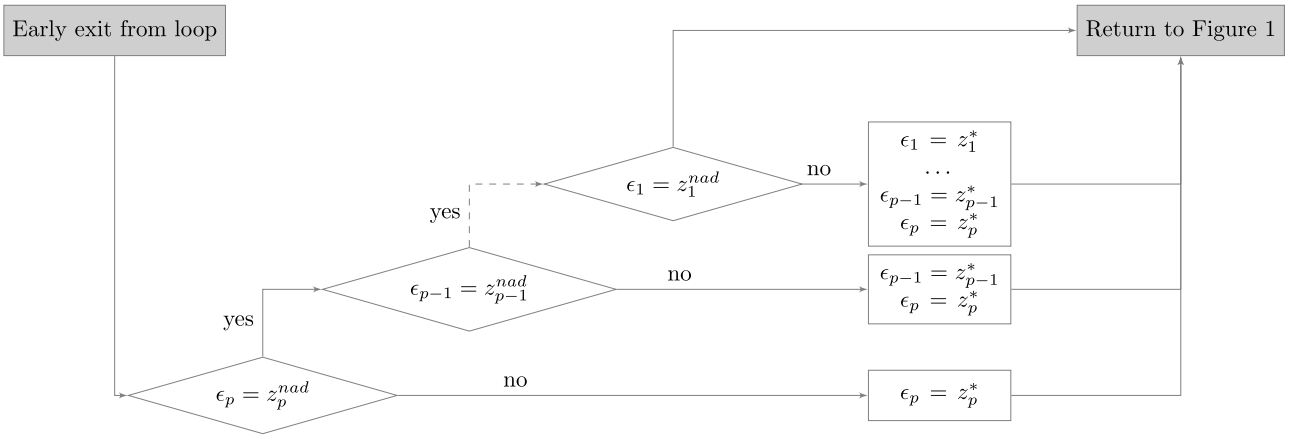


Fig. 3. Accelerated exit algorithm.

acceptable coverage error for each objective: $\gamma_k = (z_k^* - z_k^{nad})/\pi_k$. The sum of the specified cardinality in each objective will correspond to the approximate cardinality of the representation, i.e., $\Pi(R(N)) = \sum_{k=1,\, k\neq q}^{p} \pi_k$. Note that the cardinality of the representation $\Pi(R(N))$ is merely an approximation since the distribution of the Pareto front is unknown and it is not this parameter that controls the representation.

Algorithm 2 presents the method to adjust parameter ϵ_k when aiming to have a representation that privileges coverage. It performs a single run for every iteration and for each objective function z_k, i.e. for each loop, after solving a problem and obtaining, or not, a new criterion vector. The algorithm takes six inputs: (1) the desired coverage error over that loop, γ_k; (2) the value of the ϵ_k parameter of Problem 5 used in the current iteration; (3) the kth component of the criterion vector z, z_k, for the Problem 5 used in the current iteration; (4) the kth component of the ideal vector z^*, z_k^*; (5) the kth component of the approximation to the nadir vector z^{nad}, z_k^{nad}; and (6) the set D, corresponding to the ordered set of the discarded points for the current loop, which contains the points already obtained as well as the areas where redundant points will be obtained. The algorithm's outputs are the updated value of parameter ϵ_k to be used in the next iteration and the updated ordered set of discarded points, D, for the current loop. To this end, the first part of the algorithm updates D based on one of the following two cases:

Algorithm 2: Adjusting the parameter ϵ_k in *GPBA-A*.

Input: γ_k, ϵ_k, z_k, z_k^*, z_k^{nad}, D;
Output: ϵ_k, D;
if $(X^{\epsilon s} = \{\})$ **then**
  $D \leftarrow D \cup \{\lceil \epsilon_k \rceil, \lceil \epsilon_k \rceil + 1, \ldots, z_k^*\}$;
else
  $D \leftarrow D \cup \{\lceil \epsilon_k \rceil, \lceil \epsilon_k \rceil + 1, \ldots, z_k\}$;
if $(\epsilon_k = z_k^{nad})$ **then**
  $\epsilon_k \leftarrow z_k^*$;
else
  Let z_1 and z_2 be the most distant consecutive values in D;
  if $(d(z_1, z_2) \leqslant \gamma_k)$ **then**
    $\epsilon_k \leftarrow z_k^* + 1$;
    $D \leftarrow \{\}$;
  else
    $\epsilon_k \leftarrow (z_1 + z_2)/2$;
return(ϵ_k, D);

1. If Problem 5 solved in this iteration, when using ϵ_k, is infeasible (the feasible region is empty, $X^{\epsilon s} = \{\}$), then if the problem is more constrained it will be infeasible too, as stated in Proposition 1. As a result, all the points ranging from ϵ_k to the kth component of the ideal vector, z_k^*, can be discarded, and should be added to D for the current loop.
2. If Problem 5 solved in this iteration, when using ϵ_k, is feasible, all the points ranging between ϵ_k to the kth component of the obtained vector, z_k, can be discarded and should be added to D. This is based on Proposition 2.

Afterwards, using the updated set of discarded points, D, the algorithm computes the new value for parameter ϵ_k:

1. In the case where the current value of ϵ_k is the kth component of the pessimistic vector, z_k^{nad}, the problem solved in this iteration corresponds to the first extreme point. Hence, the following point to compute is the next extreme, the ideal value z_k^*.
2. Otherwise, D will have at least more than two points. As a result, the two consecutive points distancing the most from each other are drawn from set D. Then, on the one hand, if the distance between those points meets the desired coverage error, the loop is be exited. To do so, the set D is updated as empty and the ϵ_k parameter is set to a value higher than the ideal point value of objective function z_k, $z_k^* + 1$. On the other hand, if the distance between those points meets the desired coverage error, ϵ_k is set to find a point that minimizes the coverage error, i.e., a point at least in between the two most distant consecutive points, $(z_1 + z_2)/2$.

3.2. Uniformity grid point based representation (GPBA-B)

A representation that privileges uniformity is a representation which spreads points as uniformly as possible. By maximizing the minimum distance between two points in the representation, the uniformity level $\Delta(R(N))$ is increased. This representation is controlled by an acceptable uniformity level δ.

Similar to the acceptable coverage level, the acceptable uniformity level may be difficult to define. It can be derived from the range of each objective function values, $(z_k^* - z_k^{nad}$, for all $k = 1, \ldots, p$, $k \neq q$) and the acceptable cardinality in each objective (π_k, for all $k = 1, \ldots, p$, $k \neq q$). The ratio between the range and cardinality provides the acceptable uniformity level for each objective: $\delta_k = (z_k^* - z_k^{nad})/\pi_k$.

The procedure to determine ϵ_k, for the next iteration, is presented in Algorithm 3. It takes as input the defined uniformity level δ_k and the kth component of the obtained vector, z_k. The algorithm in each iteration adds δ_k to z_k, until ϵ_k is greater than the

Algorithm 3: Adjusting the parameter ϵ_k in *GPBA-B*.

Input: δ_k, ϵ_k;
Output: ϵ_k;
$\epsilon_k \leftarrow z_k + \delta_k$;
return(ϵ_k);

maximum value for objective function z_k, z_k^*. In that case the loop is exited because the problem becomes infeasible.

3.3. Cardinality grid point based representation (GPBA-C)

The cardinality representation we propose provides a range insensitive search strategy to distribute the desired number of criterion vector in the feasible region. When the desired cardinality is specified, it is very common that the ranges of the objectives in which it is based are overestimated. As a result, the final cardinality of the representation will be significantly lower than the originally considered, and in which the determination of parameters, such as uniformity level δ and coverage error γ, may have been based. The cardinality representation algorithm addresses this drawback by defining the search region in a way that intends to maintain the originally determined cardinality, and provides a balance between uniformity and coverage.

The cardinality representation algorithm starts by defining a uniform grid between the ideal and nadir points. Whenever a criterion vector is computed that skips a step on the grid, the grid is redefined from that point until the ideal value, considering only the remaining points left to compute. This results in a refinement of the search grid within the feasible region of the problem, and consequently in a range insensitive search strategy.

Algorithm 4 presents the procedure to adapt parameter ϵ_k. It takes six inputs: (1) the grid starting point for that loop, z_k^{start}; (2) the kth component of the ideal vector z^*, i.e. the grid end point for that loop, z_k^*; (3) the number of points to obtain from that grid, π_k'; (4) the position within the grid for this objective, i.e. the grid point number, i_k; (5) the kth component of the criterion vector z, z_k, for the Problem 5 used in the current iteration; and (6) the kth slack variable, s_k, for the Problem 5. For the first iteration, prior to entering Algorithm 4, the grid start point, z_k^{start}, corresponds to the nadir approximation vector, z_k^{nad}, the position within the grid, i_k, to zero, and the number of points to obtain, π_k', to the cardinality for that objective minus 1, $\pi_k - 1$. The algorithm outputs the new parameter ϵ_k and the updated grid parameters, z_k^{start}, π_k' and i_k. To that end, the following procedure is applied:

1. The first step determines if the criterion vector from the problem solved in the previous iteration, z_k, makes any of the next grid points redundant according to Proposition 2, i.e. if a step on the grid may be skipped. Accordingly, the grid step size, $step$, is computed by dividing the grid range by the number of grid points in that grid. The integer part of the division between the slack variable, s_k, and the grid step size, $step$, determines how many grid points may be skipped.
   (a) In case there are steps on the grid to skip, the grid is redefined. The new starting point for the grid, z_k^{start}, will become the corresponding component of the criterion vector obtained in the current iteration, z_k. The updated number of grid points for the new grid, π_k', corresponds to the number of points from the current grid subtracted the number of grid points already computed, i_k. The updated position within the grid, i_k, will be the first, at the beginning of the new grid. Finally, the step according to the newly defined grid is computed.
   (b) Otherwise, the grid is not updated and the position within the grid, i_k, is incremented.
2. The second step makes use of the grid parameters computed in the previous step, z_k^{start}, i_k and $step$, to update parameter ϵ_k. In case ϵ_k surpasses the ideal vector value z_k^*, the loop is exited and the grid parameters return to their initial values: the grid start point, z_k^{start}, corresponds to the nadir approximation vector, z_k^{nad}; the number of points to obtain, π_k', to the cardinality for that objective, π_k; and the position within the grid, i_k, to zero.

Algorithm 4: Adjusting the parameter ϵ_k in *GPBA-C*.

Input: z_k^{start}, z_k^*, π_k', i_k, z_k, s_k;
Output: ϵ_k, i_k, π_k', z_k^{start};
$step \leftarrow \max\left\{(z_k^* - z_k^{start})/\pi_k',\ 1\right\}$;
$b \leftarrow \lfloor |s_k/step| \rfloor$;
if $(b > 0)$ **then**
  $z_k^{start} \leftarrow z_k$;
  $\pi_k' \leftarrow \pi_k' - i_k$;
  $i_k \leftarrow 1$;
  $step \leftarrow \max\left\{(z_k^* - z_k^{start})/\pi_k',\ 1\right\}$;
else
  $i_k \leftarrow i_k + 1$;
$\epsilon_k \leftarrow z_k^{start} + i_k \times step$;
if $(\epsilon_k > z_k^*)$ **then**
  $z_k^{start} \leftarrow z_k^{nad}$;
  $\pi_k' \leftarrow \pi_k$;
  $i_k \leftarrow 0$;
return$(\epsilon_k, z_k^{start}, i_k, \pi_k')$;

3.4. An illustrative example

Consider the following numerical example, originally presented in Isermann and Steuer (1988):

$$
\begin{array}{lrrrrrrrrrrrrrrrrrrrrr}
\max z_1(x) & = & & 2x_1 & & & & & & - & 2x_4 & & & - & 2x_6 & - & 2x_7 & & \\
\max z_2(x) & = & - & 2x_1 & + & x_2 & + & 2x_3 & - & x_4 & + & x_5 & + & 2x_6 & - & x_7 & & \\
\max z_3(x) & = & - & x_1 & - & 2x_2 & & & - & 2x_4 & + & 3x_5 & + & x_6 & & & & \\
\text{subject to:} & & & x_1 & + & x_2 & + & 3x_3 & & & + & 3x_5 & + & 2x_6 & & & \leqslant & 61 \\
& & & & & 3x_2 & + & 2x_3 & + & 4x_4 & & & & & & & \leqslant & 72 \\
& & & 5x_1 & + & 3x_2 & & & & & + & 5x_5 & + & 4x_6 & + & 4x_7 & \leqslant & 76 \\
& & & 4x_1 & + & 2x_2 & & & + & 4x_4 & & & + & 4x_6 & & & \leqslant & 51 \\
& & & 5x_1 & + & 2x_2 & & & + & 3x_4 & + & x_5 & + & 4x_6 & & & \leqslant & 66 \\
& & & 2x_1 & + & 2x_2 & & & + & 4x_4 & + & 4x_5 & + & 4x_6 & + & 5x_7 & \leqslant & 59 \\
& & & 3x_1 & & & + & 2x_3 & & & + & 5x_5 & + & x_6 & + & 2x_7 & \leqslant & 77 \\
& & & x_1, & & x_2, & & x_3, & & x_4, & & x_5, & & x_6, & & x_7 & \in & \mathbb{Z}_0^+.
\end{array}
$$

This section illustrates the three representation algorithms, *GPBA-A*, *GPBA-B*, and *GPBA-C*. The first iteration over the outermost

loop is used and the results represented in Appendix A. Common to all representations is the need to compute both the ideal point and the nadir point, or, alternatively, an approximation of the nadir point. In this case, the ideal point is $z^* = (24, 49, 42)$, and the nadir approximation point, obtained through the minimization of each individual objective function, is $z^{nad} = (-28, -28, -48)$. Additionally, z_1 will be used as z_q, z_2 as the outermost loop, and z_3 as the innermost loop. Hence, all algorithms start with $\epsilon_2 = -28$ and $\epsilon_3 = -48$, as well as with the relative worst value for the outermost loop $z_2^{wv} = 49$.

3.4.1. Coverage grid point based representation (GPBA-A)

Assume the acceptable coverage error for the representation being computed is $\gamma = 15$, for all objective functions. The first iteration over the outermost loop will have $\epsilon_2 = -28$. Then, the iterations over the innermost loop, $k = 3$, are the following:

1. The first iteration over the innermost loop (Fig. A.4a) solves Problem 5 using $\epsilon_2 = -28$ and $\epsilon_3 = -48$. Point $z^1 = (24, 9, -14)$ is obtained and added to the representation $R(N) = \{z^1\}$. Next, the relative worst value for objective function z_2, z_2^{wv}, and the ordered set of discarded points, D, are updated to $z_2^{wv} = 9$ and $D = \{-48, -47, \ldots, -14\}$. Since the current value for ϵ_3 is z_3^{nad}, the new value for ϵ_3 is 42, corresponding to the ideal value.
2. Applying the flowchart on Fig. 2, Problem 5 needs to be solved for $\epsilon_2 = -28$ and $\epsilon_3 = 42$, and the point obtained is $z^2 = (0, 20, 42)$ (Fig. A.4b). Point z^2 is added to the representation, resulting in $R(N) = \{z^1, z^2\}$. Since $z_2^2 \geqslant z_2^{wv}$, the worst value for objective function z_2 does not need to be updated. z_3^2 is added to the set of discarded points $D = \{-48, -47, \ldots, -14, 42\}$. The two most distant consecutive values in D are -14 and 42, hence the new value of ϵ_3 is $(42 + (-14))/2 = 14$.
3. Again, applying the flowchart on Fig. 2, Problem 5 is solved for $\epsilon_2 = -28$ and $\epsilon_3 = 14$, obtaining $z^3 = (14, 13, 14)$ (Fig. A.4c), and updating the representation to $R(N) = \{z^1, z^3, z^2\}$. Again, the worst value for objective function z_2, z_2^{wv}, does not require updating, while the set of discarded points is updated to $D = \{-48, -47, \ldots, -14, 14, 42\}$. At this point the maximum distance between points D is 28. As a result, the new value for ϵ_3 is $(14 + (-14))/2 = 0$.
4. Problem 5 is solved for $\epsilon_2 = -28$ and $\epsilon_3 = 0$, obtaining $z^4 = (22, 6, 1)$ (Fig. A.4d), and updating the representation to $R(N) = \{z^1, z^4, z^3, z^2\}$. The worst value for objective function z_2, z_2^{wv}, needs to be updated $z_2^{wv} = 6$. The set of discarded points is updated to $D = \{-48, -47, \ldots, -14, 0, 1, 14, 42\}$. The new value for ϵ_3 is $(42 + 14)/2 = 28$.
5. Using $\epsilon_2 = -28$ and $\epsilon_3 = 28$, Problem 5 is solved and $z^5 = (8, 13, 29)$ is obtained (Fig. A.4e). criterion vector z^5 is then added to the representation, $R(N) = \{z^1, z^4, z^3, z^5, z^2\}$. The worst value for objective function z_2 remains the same, $z_2^{wv} = 6$. The set of discarded points is updated to $D = \{-48, -47, \ldots, -14, 0, 1, 14, 28, 29, 42\}$. At this point, the maximum distance between points in the set of discarded points is 14, which is less than γ. Hence, the loop stops, and the coverage error obtained for the first iteration over the outermost loop was 15.

For the next iteration over the loop of the objective function z_2, the value of $z_2^{wv} = 6$ is used as z_2 for the discarded vector of that loop.

3.4.2. Uniformity grid point based representation (GPBA-B)

Suppose the objective is to have a representation with at least a uniformity level of $\delta = 10$ over all objective functions. The algorithm begins with $\epsilon_2 = -28$, for the outermost loop, and the following iterations occur over the innermost loop:

1. In the first iteration (Fig. A.5a) Problem 5 is solved using $\epsilon_3 = -48$. Point $z^1 = (24, 9, -14)$ is obtained and added to the representation $R(N) = \{z^1\}$. The new ϵ_3 is updated according to the desired uniformity level as: $\epsilon_3 = z_3^1 + \delta_3 = -14 + 10 = -4$. Before the next iteration the worst value for objective function z_2, z_2^{wv}, is also updated to 9.
2. The second iteration (Fig. A.5b) uses $\epsilon_3 = -4$ to solve Problem 5 and point $z^2 = (24, 5, -3)$ is obtained and added to the representation, $R(N) = \{z^1, z^2\}$. The third component of the ϵ vector is updated accordingly, becoming $\epsilon_3 = 7$. Since $z_2^2 < z_2^{wv}$, the worst value for objective function z_2 is updated, resulting in $z_2^{wv} = 5$.
3. When solving Problem 5 with $\epsilon_2 = -28$ and $\epsilon_3 = 7$ (Fig. A.5c) point $z^3 = (18, 8, 9)$ is obtained and added to the representation, $R(N) = \{z^1, z^2, z^3\}$. The ϵ_3 is updated as $\epsilon_3 = z_3^3 + \delta_3 = 9 + 10 = 19$. Again, since $z_2^2 \geqslant z_2^{wv}$, the worst value for objective function z_2 is not updated.
4. In the fourth iteration (Fig. A.5d) Problem 5 is solved using $\epsilon_3 = 19$. The criterion vector is $z^4 = (12, 11, 21)$, which is added to the representation, $R(N) = \{z^1, z^2, z^3, z^4\}$. As a consequence, ϵ_3 is updated as $z_3^4 + \delta_3 = 21 + 10 = 31$. In this iteration, because $z_2^2 \geqslant z_2^{wv}$, the worst value for objective function z_2 does not need to be updated, remaining at $z_2^{wv} = 5$.
5. For the fifth iteration (Fig. A.5e) Problem 5 is solved using $\epsilon_3 = 31$. The resulting criterion vector is $z^5 = (6, 14, 33)$, which is added to the representation, becoming $R(N) = \{z^1, z^2, z^3, z^4, z^5\}$. Once more, z_2^{wv} is not updated since $z_2^{wv} < 14$. Then, ϵ_3 is updated following Algorithm 3. However, since the resulting ϵ_3 is 43, which is larger than $z_3^* = 42$, the loop is finished (see Fig. 1).

For the next iteration over the outermost loop, objective function z_2, the value of z_2^{wv} is used as z_2 to compute the next value of ϵ_2, $\epsilon_2 = z_2 + \delta_2 = 5 + 10 = 15$.

3.4.3. Cardinality grid point based representation (GPBA-C)

Consider a desired cardinality for the representation of 25 points, corresponding to a division of each constrained objective function z_k, $k = 1, \ldots, p$, $k \neq q$, into 5. The first iteration over the outermost loop, $k = 2$, starts with $\epsilon_2 = z_2^{nad} = -28$ and $z_2^{wv} = z_2^* = 49$, and then iterates over the innermost loop in the following way:

1. For the first iteration, in Fig. A.6a, ϵ_3 is -48, corresponding to the nadir approximation value for that objective, z_3^{nad}. Solving Problem 5 with those parameters, yields the criterion vector $z^1 = (24, 9, -14)$, which is added to the representation $R(N)$, and we update z_2^{wv} with 9. Since this is the first iteration, the grid start point z_3^{start} is -48, i.e., the nadir approximation value, the position within the grid, i_3, is 0, and the number of points to obtain with such a grid, c'_3, is equal to the objective's cardinality minus one, 4. The slack variable for Problem 5, s_3, is the difference between ϵ_3 and the third component of the criterion vector, i.e., $s_3 = -14 - (-48) = 34$. To determine whether a step in the grid may be skipped or not, we first compute the step size: $step = \max\{(42 - (-48))/4,\ 1\} = 22.5$. Since the number of steps to skip, b, is the integer part of $34/22.5 = 1.51$, then one step in the grid must be skipped. Consequently, the grid is refined from the obtained points, $z_3^{start} = z_3^1 = -14$, onwards, considering the number of points left to compute: $c'_3 = 4 - 0 = 4$, $step = \max\{(42 - (-14))/4,\ 1\} = 14$, $i_3 = 1$ (Fig. A.6b). Hence, the ϵ_3 parameter for the following iteration is $\epsilon_3 = -14 + 1 \times 14 = 0$.
2. For the second iteration, in Fig. A.6b, we use $\epsilon_3 = 0$ when solving Problem 5, leading to the vector $z^2 = (22, 6, 1)$, which is added to the representation, $R(N) = \{z^1, z^2\}$. The worst value

for objective function z_2 needs to be updated with 6, since $z_2^2 < z_2^{wv}$. As a result, the third slack variable for Problem 5 is $s_3 = 1 - 0 = 1$. Since the step size is 14, no point is skipped, and it can be determined by the integer part of $b = \lfloor 1/14 \rfloor = 0$. Hence the grid remains the same and the position within the grid is increased by one, $i_3 = 2$. Hence, the ϵ_3 parameter for the following iteration is $\epsilon_3 = -14 + 2 \times 14 = 14$.

3. For the third iteration, in Fig. A.6c, we solve Problem 5 with $\epsilon_3 = 14$, leading to vector $z^3 = (14, 13, 14)$, which is added to the representation, $R(N) = \{z^1, z^2, z^3\}$. The worst value for objective function z_2, z_2^{wv}, remains the same. The slack variable for Problem 5 is $s_3 = 0$, hence no point is skipped and the grid is kept. The position within the grid is increased by one, $i_3 = 3$, and ϵ_3 parameter is updated to $\epsilon_3 = -14 + 3 \times 14 = 28$.
4. For the forth iteration, in Fig. A.6d, we use $\epsilon_3 = 28$, which results in the criterion vector $z^4 = (8, 13, 29)$, and the representation $R(N) = \{z^1, z^2, z^3, z^4\}$. The worst value for objective function z_2 keeps its value $z_2^{wv} = 6$, because $z_2^4 \geqslant z_2^{wv}$. Since $z_3^4 - \epsilon_3 = 1$ corresponds to the slack variable, there are no redundant points in the grid, $b = \lfloor 1/14 \rfloor = 0$. As a result the grid remains the same and the position within the grid is increased by one, $i_3 = 4$. The ϵ_3 parameter is then updated to $\epsilon_3 = -14 + 4 \times 14 = 42$.
5. For the fifth and last iteration in this loop, in Fig. A.6e, we use $\epsilon_3 = 42$, which results in the criterion vector $z^5 = (0, 20, 42)$, updating the representation to $R(N) = \{z^1, z^2, z^3, z^4, z^5\}$. Since $\epsilon_3 = z_3^5$ no point is skipped and the position within the grid may be increased by one, $i_3 = 5$. Due to i_3 being greater than c_3', the updated value of $\epsilon_3 = -14 + 5 \times 14 = 56$ is greater than the ideal value for objective function z_3, $z_3^* = 42$. Hence, the loop will be exited and the grid control parameters for this objective, $k = 3$, will return to the values of the first iteration.

For the next iteration over the outermost loop, objective function z_2, the value of $z_2^{wv} = 6$ is used as z_2 to compute the next value of ϵ_2, using Algorithm 4. Because the slack variable $s_2 = -28 - 6 = -34$ and the step size to $step = \max\{(49 - (-28))/4, 1\} = 19.25$, one step is skipped, $b = \lfloor | -34/19.25 | \rfloor = 1$. The grid parameters are adapted to $z_2^{start} = z_2 = 6$, $c_2' = 4 - 0 = 4$ and $i_2 = 1$. The step size is updated to $step = \max\{(49 - 6)/4, 1\} = 10.75$ and, at last, $\epsilon_2 = 6 + 1 \times 10.75 = 16.75$.

3.5. Generating the entire Pareto front

As the three algorithms presented in the above section, *GPBA-A*, *GPBA-B*, and *GPBA-C*, are based on solving a sequence of variations of Problem 5, they can be used to generate the whole Pareto front under the condition that the objective function only takes integer values. Additionally the parameters that control each algorithm, the acceptable coverage error, the uniformity level, and the cardinality, must be chosen to ensure the production of the entire Pareto set.

When using algorithm *GPBA-A* to compute the entire Pareto front, the desired acceptable coverage error, γ, is 1, guaranteeing that the distance between any two consecutive points in the Pareto front representation is at most unitary. Since these algorithms are applied to problems with integer objective functions, if $\Gamma \leqslant 1$, then Γ is, in fact, equal to 1.

In case algorithm *GPBA-B* is used, the acceptable uniform level, δ, should be set to 1 to achieve the computation of the entire Pareto front. Setting $\delta = 1$ ensures the algorithm will search for the next solution as close to the previous as a unitary step in the objective space.

Finally, for the case of *GPBA-C*, when generating the entire Pareto front, the cardinality should be set for each objective as the respective range. As a result, a unitary step size is used for each objective function's loop and the algorithm restricts the search region for the next solution to a unitary distance in the objective space.

The three proposed algorithms were tested on multiple instances on their performance when generating the whole Pareto front. To that end, the above parameters were used. Results of those experiments are presented in Section 4.

4. Computational experiments

This section presents the design of the experiments, some implementation issues, the computational results for the identification of the whole Pareto front, some experiments on the representations of the Pareto front, and a few final comments and remarks.

The performance of the three proposed algorithms *GPBA-A*, *GPBA-B* and *GPBA-C* is compared with the works of Mavrotas and Florios (2013), Zhang and Reimann (2014) and Nikas et al. (2020), hereinafter referred to as *AUGM-2*, *S-AUGM* and *R-AUGM*, respectively. All algorithms, apart from *R-AUGM* were coded using PyCharm Community Edition 2020 for Windows Desktop, Python 3.7 as the interpreter and CPLEX 12.9 as the optimization solver. For *R-AUGM*, the code provided by the authors in https://github.com/KatforEpu/Augmecon-R was adapted regarding the instances. All experiments were performed in a workstation equipped with two processors Intel Xeon X5680 3.33GHz and 24GB of RAM, running Windows 10.

The code for the algorithms proposed in Section 3 as well as the problem instances used in this section are available in the following website: https://fenix.tecnico.ulisboa.pt/homepage/ist175325/apresentacao.

4.1. The design of the experiments

The generation of the instances used in these experiments was performed through a generator developed by the second author of this study[1]. This generator makes use of the random number generation procedure of NETGEN generator proposed in Klingman, Napier, and Stutz (1974) and requires as input the following elements:

- The number of variables (n), objective functions (p) and constraints (m).
- The ranges for the parameter values (coefficients of the constraints and the weights/profits of the objective functions). These parameters are randomly generated by using a uniform distribution with a specific seed number for each constraint/objective function.
- In this study, all seed numbers are increased by a fixed value, for each instance generated.
- The right-hand side of each constraint is bounded by half the sum of the constraint's coefficients.

The algorithm's performance is essentially compared in terms of solution quality and running time, or a proxy of the latter, such as the number of iterations per non-dominated criterion vector computed. As such, since both running time and its proxies are often not normally distributed, each problem type was tested on 30 instances. The choice of 30 instances was motivated by the conclusion of Coffin and Saltzman (2000) that, for those measures of performance, 30 to 40 observations would guarantee statistically significant results. Apart from the aforementioned observation, Coffin and Saltzman (2000) also provided other suggestions to draw meaningful conclusions from the comparison of algorithms

[1] For more details please contact José Rui Figueira at figueira@tecnico.ulisboa.pt

that were followed in this work, such as the use of sets of problems where certain parameters are varied, and the use of mean and variance of metrics for comparison. As a result, a total of 570 different uncorrelated instances were generated.

4.2. Implementation issues

All algorithms, aside from *AUGM-2*, can overcome poor nadir estimations, since the constraints' right-hand-side is computed based on the obtained solution and not beforehand. Hence, for those algorithms, in these experiments, we simply use the single minimization of each objective function individually as a nadir point. However, since the improvements on the constrained objective functions are weighted by the width of the objective's range, due to some solvers' sensitivity, the value of parameter ρ in Problem 5 may have to be adapted for some instances. That is, when the slack variable value is low, and it is divided by the width of the objective's range and multiplied by a small number ρ, this product may become smaller than the software's sensitivity. Hence, one way to go around this implementation issue is to increase the value of ρ. The consequence of not adapting parameter ρ is the failure to compute some non-dominated criterion vectors. For the results presented in this paper, $\rho = 10^{-3}$ was used for algorithm *AUGM-2* and $\rho = 10^{-2}$ for the remaining algorithms.

Algorithm *AUGM-2*, however, requires good quality nadir estimations. For such purpose, as suggested in Mavrotas and Florios (2013), the lexicographic pay-off table is used to provide an approximation of the nadir point. However, as it has been discussed in literature, lexicographic pay-off tables do not guarantee a good estimation of nadir points for more than two objective functions (see, e.g., Alves & Clímaco, 2007; Isermann & Steuer, 1988). For this implementation reason, in some instances, *AUGM-2* would not explore the whole Pareto front and, consequently, skip non-dominated criterion vectors. To prevent this from happening, as proposed by Mavrotas and Florios (2013), the objective functions' range provided by the lexicographic payoff table were expanded by 15%.

Algorithm *GPBA-A* requires recording the set of discarded points, D. There are as many D sets as the number of constrained objectives, i.e. $p-1$, and the set is temporary only existing while still in the same computation over the previous loop. The size of each set D can be at most the size of the integer size of the objective functions' ranges. This can create memory overflow issues. However, in the experiments described in the remaining of this paper, that problem never appeared. Nevertheless, if that issue arises, a better nadir point estimation than the single minimization of each objective function individually is required. In such case, the range provided by the lexicographic payoff table can be expanded by 10%/15%.

The implementation of *R-AUGM* provided by the authors of Nikas et al. (2020) requires the pre-computation of the objective functions' range to determine the number of grid points. That procedure was done using as nadir estimation the single minimization of each objective function individually as a nadir point.

4.3. Experiments on generating the whole Pareto front

To evaluate the computational and accuracy behaviour of different algorithms, when computing the Pareto front, experiments were performed on binary and integer instances, considering both single and multi-dimensional cases. Additionally, 180 tests were run on multi-objective general integer programming instances. This was done in order to, not only compare the algorithms, but also understand how changes in the number of objective functions, constraints, variables, and problem types affect the performance of the algorithms.

In the remaining of this section, analysis of the results for the aforementioned experiments are presented. This analysis is accompanied by summary tables, where underlined results show algorithms' drawbacks; and results in bold show algorithms' strengths. Algorithms are evaluated on the average and standard deviation over the number of unique non-dominated criterion vectors found ($|\overline{N(Z)}|$ and $\sigma_{N(Z)}$) and CPU time, as well as the average number of iterations computed for each non-dominated criterion vectors found ($\overline{iter/\hat{z}}$). Although the latter corresponds to a proxy of CPU time, it is included in this study because it is not correlated to the number of criterion vectors in the Pareto front neither with the implementation software while CPU time typically is. It therefore coveys better the influence of the remaining problem parameters on the algorithm's performance.

4.3.1. Multi-objective $\{0,1\}$−knapsack instances

The first set of problems studied were multi-objective binary knapsack instances with three and four objective functions, considering one constraint. The three-objective instances have 50, 75, and 100 variables, while the four-objective instances were generated with just 50 variables. Both the objective function's profits and the constraint's coefficients are drawn from a uniform distribution of type $U[1, 100]$. The initial seed numbers for the three-objective instances with 50 variables were 128, 888 and 6 for each objective function profit, and 40 for the constraint's coefficients. For the remaining three-objective instances, the seed numbers for the objective functions' profits were 47, 28 and 626, respectively, whilst the seed number 135 was used for the constraint's coefficients. Lastly, for the four-objective instances, the seed numbers of the objective functions' profits are 47, 28, 626 and 135, and 298 for the constraint's coefficients. All seed numbers are increased by 5 on each instance of each type. For each set of problems, 30 instances were generated. Table 1 presents experiments results for the above described instances.

Regarding the instances with three objectives, the computational time increases significantly with the number of criterion vectors in the Pareto front. However, the increase in the number of variables does not impact the number of iterations per non-dominated criterion vectors. All algorithms computed the same number of non-dominated criterion vectors. Algorithm *AUGM-2* is the least efficient method among the ones studied, both in terms of computational time and in terms of number of iterations per non-dominated criterion vector obtained. The reason for this is that, in this algorithm, iteration skipping and early exit mechanisms are only applied to the innermost loop. Furthermore, as explained in Section 4.2, *AUGM-2* has a smaller space to search since it is given a better nadir approximation than the other algorithms. *GPBA-A* also does not present competitive results since more iterations are performed per non-dominated criterion vector computed for this algorithm when compared to the others, which is also reflected in the computational time. *GPBA-A* presents this behaviour due to the fact that the accelerated early exit strategy cannot be applied to this algorithm. The results displayed by algorithms *AUGM-2* and *GPBA-A* justify the need and added value of the integration of the acceleration strategies (Figs. 2 and 3). *R-AUGM*, although presenting better results than *GPBA-A*, performs worse than the other proposed algorithms in terms of number of iterations done per non-dominated criterion vector. Consequently, algorithms *AUGM-2*, *GPBA-A* and *R-AUGM* were discarded from further analysis. Finally, the performances of *S-AUGM* and the other two proposed algorithms, *GPBA-B* and *GPBA-C*, are similar.

In the case of the four-objective instances, for the aforementioned reasons, only algorithms *GPBA-B* and *GPBA-C* are compared

Table 1
Multi-objective 0–1 knapsack instances.

p	n	Algorithm	$\lvert\overline{N(Z)}\rvert$	$\sigma_{N(Z)}$	$\overline{iter/\hat{z}}$	*CPU* (sec)	σ_{CPU}
3	50	*AUGM-2*	408.27	210.30	67.02	1083.47	599.83
		R-AUGM	408.27	210.30	1.99	183.02	77.82
		S-AUGM	408.27	210.30	**1.89**	58.36	34.51
		GPBA-A	408.27	210.30	14.37	394.56	238.80
		GPBA-B	408.27	210.30	**1.89**	**58.36**	**34.04**
		GPBA-C	408.27	210.30	**1.89**	58.61	34.64
	75	*AUGM-2*	1518.83	920.27	49.18	3523.38	1469.57
		R-AUGM	1518.83	920.27	1.89	961.57	449.15
		S-AUGM	1518.83	920.27	**1.82**	263.34	193.64
		GPBA-A	1518.83	920.27	11.69	1771.25	1378.51
		GPBA-B	1518.83	920.27	1.83	**260.03**	190.42
		GPBA-C	1518.83	920.27	1.83	260.18	189.67
	100	*AUGM-2*	3356.40	1628.35	42.07	9806.73	3775.94
		R-AUGM	3356.40	1628.35	1.84	2524.40	1011.70
		S-AUGM	3356.40	1628.35	**1.79**	666.86	411.67
		GPBA-A	3356.40	1628.35	9.28	4501.94	2674.97
		GPBA-B	3356.40	1628.35	**1.79**	**661.26**	427.21
		GPBA-C	3356.40	1628.35	**1.79**	662.42	427.22
4	50	*S-AUGM*	2655.43	1392.73	**4.74**	**9940.89**	7008.43
		GPBA-B	2655.43	1392.73	**4.74**	10104.69	7152.10
		GPBA-C	2655.43	1392.73	4.77	10032.92	7205.45

Table 2
Multi-objective multi-dimensional 0–1 knapsack instances.

p	n	m	Algorithm	$\lvert\overline{N(Z)}\rvert$	$\sigma_{N(Z)}$	$\overline{iter/\hat{z}}$	*CPU* (sec)	σ_{CPU}
3	25	2	*S-AUGM*	80.03	38.39	**1.91**	**8.06**	4.53
			GBPA-B	80.03	38.39	**1.91**	8.11	4.49
			GBPA-C	80.03	38.39	**1.91**	8.30	4.74
		3	*S-AUGM*	84.83	40.56	**1.92**	**8.95**	4.81
			GPBA-B	84.83	40.56	**1.92**	8.99	4.95
			GPBA-C	84.83	40.56	**1.92**	9.03	4.90
		4	*S-AUGM*	88.30	44.96	**1.92**	**9.67**	5.49
			GPBA-B	88.30	44.96	**1.92**	9.78	5.58
			GPBA-C	88.30	44.96	**1.92**	9.73	5.56
		5	*S-AUGM*	90.73	37.97	**1.92**	**9.94**	4.71
			GPBA-B	90.73	37.97	**1.92**	10.02	4.85
			GPBA-C	90.73	37.97	**1.92**	10.12	4.89

with *S-AUGM*. Analysis of the results shows for the three compared algorithms high sensitivity of the number of required iterations to the increase in the number of objectives. Furthermore, although all algorithms perform similarly, *GPBA-C* appears to have the lowest efficiency having performed, on average, the most iterations per non-dominated criterion vector found. This could be explained by possibly needing a readjustment of parameter ρ in Problem 5, as explained in Section 4.2. However, when comparing in terms of computational time, *GPBA-C* is slightly more efficient than *GPBA-B*.

4.3.2. Multi-objective multi-dimensional $\{0,1\}$−knapsack instances

For the multi-objective multi-dimensional binary knapsack problems, instances were generated varying the number of constraints between 2 and 5, with a fixed number of objectives at 3 and variables at 25. Both the objective function's profits and the constraint's coefficients follow a uniform distribution bounded between 1 and 100. The seed numbers used for the objective function's profits were 47, 63 and 728, while for each constraint equation's coefficient the values used were 626, 135, 28, 17, 758, respectively. Table 2 shows the results for algorithms *S-AUGM*, *GPBA-B* and *GPBA-C*. For the reasons described in Section 4.3.1, related to poor performance, algorithms *AUGM-2* and *GPBA-A* are not considered in this comparison.

Table 2 shows that the performance of the algorithms is not affected by the increase in the number of constraints. Additionally, the three tested algorithms, *S-AUGM*, *GPBA-B* and *GPBA-C*, are comparable in performance, finding the whole Pareto front in a short amount of time. *S-AUGM* appears to be slightly more efficient when comparing average computational time, however differences between algorithms represent, in the worst case, 1.8%.

4.3.3. Multi-objective integer knapsack instances

To test the algorithms on integer multi-objective knapsack instances, the instances generated in Section 4.3.1 with three objectives and 50 variables were solved using integer variables. Instance number 27, that uses seed numbers 158, 1018 and 136 for the objective function's profits and 170 for the constraint's coefficients, was discarded, having exceeded a ten-hour computational time limit, due to having many non-dominated criterion vectors. Hence, the results presented in Table 3 only consider 29 instances.

Although all algorithms are able to compute the whole Pareto front, their performance differs. Whilst *S-AUGM* takes less com-

Table 3
Multi-objective integer knapsack instances.

p	n	Algorithm	$\lvert\overline{N(Z)}\rvert$	$\sigma_{N(Z)}$	$\overline{iter/\hat{z}}$	*CPU* (sec)	σ_{CPU}
3	50	*S-AUGM*	3152.59	9061.66	1.98	**2590.92**	6666.79
		GPBA-B	3152.59	9061.66	**1.71**	2662.00	7017.92
		GPBA-C	3152.59	9061.66	**1.71**	2673.95	7051.64

Table 4
Multi-objective multi-dimensional integer knapsack instances.

p	n	m	Algorithm	$\lvert\overline{N(Z)}\rvert$	$\sigma_{N(Z)}$	$\overline{iter/\hat{z}}$	*CPU* (sec)	σ_{CPU}
3	25	2	*S-AUGM*	167.07	269.14	1.93	30.13	66.06
			GPBA-B	167.07	**269.14**	**1.89**	**28.72**	63.53
			GPBA-C	167.07	**269.14**	**1.89**	28.79	62.97
		3	*S-AUGM*	292.57	302.10	1.94	50.94	58.59
			GPBA-B	292.57	302.10	**1.88**	**28.72**	54.10
			GPBA-C	292.57	302.10	**1.88**	28.79	53.83
		4	*S-AUGM*	285.50	232.03	1.93	53.51	44.51
			GPBA-B	285.50	232.03	**1.92**	**49.55**	44.51
			GPBA-C	285.50	232.03	**1.92**	49.66	44.33
		5	*S-AUGM*	238.30	169.77	**1.94**	42.30	32.86
			GPBA-B	238.30	169.77	**1.94**	**40.04**	31.53
			GPBA-C	238.30	169.77	**1.94**	40.08	31.96

putational time than the other two algorithms, around 3% less than *GPBA-C*, it is also the least efficient method in terms of the number of iteration needed to compute each non-dominated criterion vector. This suggests that both *GPBA-B* and *GPBA-C* computational time performance could be bounded by code efficiency.

4.3.4. Multi-objective multi-dimensional integer knapsack instances

To assess the impact of the increase in dimensions on the algorithms' performance on integer instances, the instances from Section 4.3.2 were applied using integer type variables. Table 4 shows the results for this case.

Although the impact of the increase in dimension does not appear to be significant, the same trend as in Table 3 can be observed where *S-AUGM* is less efficient than *GPBA-B* and *GPBA-C*. In this case, *GPBA-B* and *GPBA-C* outperform consistently *S-AUGM*, both in terms of number of iterations solved per non-dominated criterion vector obtained and in terms of computational time. The latter indicator also suggests that *GPBA-B* is more efficient than *GPBA-C*. The worse performance of *S-AUGM* in integer instances, can probably be attributed to the ϵ-constraint model formulation used on *S-AUGM*, which is not using elastic constraints as proposed by Ehrgott and Ryan (2003). However, (Table 4) as the number of constraints increase, the performance difference between *S-AUGM* and the other algorithms appears to diminish.

4.3.5. Multi-objective general integer programming instances

Instances for general integer problems were generated, setting a uniform distribution ranging between 0 and 20 for the objective function's profits and the constraints' coefficients. The instances were generated for both three and four objectives with multiple constraints and 10, 15 and 20 variables. The generation of objective function's profits started with seed numbers 47, 63, 728 and 11, one for each objective function. The initial seed numbers used to generate the constraints' coefficients was 634, 17, 28, 626, 135, 34, 78, 55, 783, 945, 823, 362, 133, 92 and 41, one for each constraint equation. Seed numbers were increased by 5 on each instance of each type. Results on solving these instances using *S-AUGM*, *GPBA-B* and *GPBA-C* are shown in Table 5

Results on the three objective instances, show the same trend as in Section 4.3.4, in which *GPBA-B* and *GPBA-C* outperform *S-AUGM*, although not as significantly. The less significant difference in performance between *S-AUGM* and the remaining algorithms may be attributed to these problems having more constraints than the ones presented in the previous section, where the performance gap was lower as the number of constraints increased. Finally, *GPBA-B* is consistently the best performing algorithm among the three.

When looking at the four objective instances, the same trend is not apparent. On the contrary, *S-AUGM* performs better than *GPBA-B* and *GPBA-C* in all sets of problems, both in terms of computational time and in terms of number of iterations per non-dominated criterion vector. *GPBA-B*, for all sets of problems, requires the most iterations per non-dominated criterion vector, while *GPBA-C* takes the most time.

4.4. Experiments on representation

To assess the quality of the proposed algorithms for the representation problem, experiments were carried out on the binary knapsack problem instances, Section 4.3.1, varying the representation's control parameters according to the instance's Pareto front range. As a pre-computing stage for each instance, the acceptable uniformity level, δ_k, and the acceptable coverage level, γ_k, are defined as percentage of the range of the objective with the narrower range. The ranges are obtained using the results of Section 4.3.1. The cardinality control parameter required by algorithm $GPBA-C$ is the maximum between the cardinality obtained by the coverage representation and the uniformity representation. Tables 6, 7 and 8 present the results for these experiments for the instances with three objectives and 50, 75 and 100 variables, respectively. The same results are also depicted in the charts in Appendix B, on Figs. B.7, B.8 and B.9, respectively. Apart from the already introduced performance measures, the obtained representations are evaluated in terms of the final obtained coverage error, $\Gamma(R(N))$, the uniformity level, $\Delta(R(N))$, the cardinality, $\Pi(R(N))$ and the percentage of criterion vectors from the whole Pareto front that are not covered by the defined coverage level for that instance, *%Ncov.*. In this section, only the three proposed algorithms are compared

Table 5
General multi-objective integer linear instances.

p	n	m	Algorithm	$\lvert\overline{N(Z)}\rvert$	$\sigma_{N(Z)}$	$\overline{iter/\hat{z}}$	*CPU* (sec)	σ_{CPU}
3	10	5	*S-AUGM*	19.83	10.14	**1.80**	1.54	1.06
			GPBA-B	19.83	10.14	**1.80**	**1.37**	0.64
			GPBA-C	19.83	10.14	**1.80**	1.45	0.75
	15	10	*S-AUGM*	38.93	20.13	1.79	3.72	2.16
			GPBA-B	38.93	20.13	**1.78**	**3.36**	1.97
			GPBA-C	38.93	20.13	**1.78**	3.45	1.96
	20	15	*S-AUGM*	75.13	31.07	**1.73**	9.03	3.91
			GPBA-B	75.13	31.07	**1.73**	**8.74**	3.95
			GPBA-C	75.13	31.07	**1.73**	8.91	3.85
4	10	5	*S-AUGM*	30.67	13.31	**3.54**	**7.73**	4.71
			GPBA-B	30.67	13.31	3.58	7.75	4.76
			GPBA-C	30.67	13.31	3.55	7.86	4.75
	15	10	*S-AUGM*	107.77	74.05	**3.60**	**43.39**	37.44
			GPBA-B	107.77	74.05	3.64	44.23	39.23
			GPBA-C	107.77	74.05	3.63	44.70	38.86
	20	15	*S-AUGM*	311.30	175.06	**3.52**	**152.01**	94.09
			GPBA-B	311.30	175.06	3.56	156.39	97.36
			GPBA-C	311.30	175.06	3.55	156.43	97.56

Table 6
Representation for multi-objective 0–1 knapsack instances with three objectives and fifty items.

p	n	γ_k/δ_k	Algorithm	$\lvert\Pi(R(N))\rvert$	$\lvert\Gamma(R(N))\rvert$	$\lvert\Delta(R(N))\rvert$	$\lvert\%Ncov.\rvert$	$\overline{iter/\hat{z}}$	*CPU*(s)
3	50	30%	*GPBA-A*	15.87	**208.97**	41.97	**6.12**	1.42	0.89
			GPBA-B	**9.10**	316.63	**67.93**	13.59	**1.15**	**0.52**
			GPBA-C	11.77	225.97	57.27	12.02	1.22	0.66
		20%	*GPBA-A*	32.40	**174.13**	21.23	**8.33**	1.43	2.88
			GPBA-B	**16.13**	247.70	**47.53**	14.24	**1.18**	**1.29**
			GPBA-C	24.10	189.93	30.77	14.65	1.20	1.78
		15%	*GPBA-A*	41.07	**162.83**	17.37	**11.68**	1.45	2.75
			GPBA-B	**24.33**	235.67	**29.17**	19.82	**1.18**	**1.49**
			GPBA-C	28.77	181.87	22.43	20.88	1.19	1.65
		10%	*GPBA-A*	77.10	**125.03**	11.53	**12.01**	1.61	5.08
			GPBA-B	**41.67**	195.53	**20.37**	23.66	**1.19**	**2.10**
			GPBA-C	50.93	150.53	14.43	22.79	1.22	2.65
		5%	*GPBA-A*	155.23	**95.30**	6.87	**17.24**	2.11	13.64
			GPBA-B	93.47	149.93	**10.47**	32.42	**1.25**	5.19
			GPBA-C	**85.10**	133.30	10.10	40.59	1.27	**4.71**
		4%	*GPBA-A*	171.77	**90.60**	6.07	**21.98**	2.32	16.72
			GPBA-B	116.57	125.67	9.17	34.93	**1.27**	6.63
			GPBA-C	**88.70**	127.50	**10.80**	50.12	1.28	**5.04**
		3%	*GPBA-A*	215.43	**85.87**	5.00	**23.27**	2.85	25.98
			GPBA-B	149.90	117.10	7.97	37.70	1.33	9.02
			GPBA-C	**105.70**	118.27	**8.90**	57.87	**1.32**	**6.16**
		2%	*GPBA-A*	259.37	**77.60**	4.33	**23.92**	3.60	39.25
			GPBA-B	199.10	110.20	6.33	37.57	1.41	13.11
			GPBA-C	**114.63**	118.07	**8.73**	65.30	**1.33**	**6.48**
		1%	*GPBA-A*	331.57	**62.50**	3.47	**15.59**	6.05	92.94
			GPBA-B	277.17	79.10	4.57	27.17	1.57	21.71
			GPBA-C	**131.70**	109.83	**8.13**	66.81	**1.36**	**7.99**

since *S-AUGM* was developed exclusively to compute the whole Pareto front and not a representation of it.

On the one hand, in Tables 6, 7 and 8, as well as in Figs. B.7, B.8 and B.9 it is possible to observe that, apart from few exceptions, *GPBA-A* presents the best results on coverage error and *GPBA-B* on uniformity level. However, except in cases of lower acceptable coverage error/uniformity level, both algorithms show the worst performance in the other measure, i.e. *GPBA-A* on uniformity level and *GPBA-B* on coverage error. On the other hand, algorithm *GPBA-C* appears to be an algorithm with a more balanced performance. When lower values for the target cardinality are set, *GPBA-C* obtains a good coverage error, similar to *GPBA-A*, while not compromising uniformity as much. For higher values of target cardinality, *GPBA-C* fails to reach them, compromising on coverage. Nevertheless, the lower cardinality levels allow for a better uniformity. Additionally, even with lower cardinality than *GPBA-B*, in many situations *GPBA-C* is able to reach a better coverage error (e.g. see in Table 6 the 4% row, and in Table 7 the 3% and

Table 7
Representation for multi-objective 0–1 knapsack instances with three objectives and seventy-five items.

p	n	γ_k/δ_k	Algorithm	$\lvert\Pi(R(N))\rvert$	$\lvert\Gamma(R(N))\rvert$	$\lvert\Delta(R(N))\rvert$	$\lvert\%Ncov.\rvert$	$\overline{iter}/\hat{z}$	CPU(s)
3	75	30%	*GPBA-A*	17.67	**314.43**	52.30	**6.85**	1.37	1.05
			GPBA-B	**10.43**	402.67	**86.10**	10.33	**1.17**	**0.68**
			GPBA-C	13.00	336.80	73.73	11.06	1.23	0.78
		20%	*GPBA-A*	40.63	**260.77**	20.80	**6.94**	1.31	2.37
			GPBA-B	**19.57**	403.37	**41.60**	14.45	**1.15**	**1.19**
			GPBA-C	31.03	265.00	31.53	9.81	1.16	1.83
		15%	*GPBA-A*	51.57	**247.50**	15.43	**10.56**	1.30	3.09
			GPBA-B	**29.93**	307.03	**31.43**	15.46	**1.15**	**1.73**
			GPBA-C	38.80	252.07	25.20	13.61	**1.15**	2.15
		10%	*GPBA-A*	116.63	**176.97**	10.07	**7.39**	1.33	7.48
			GPBA-B	**55.07**	263.43	**19.60**	17.52	1.15	**3.15**
			GPBA-C	81.40	208.17	12.67	13.73	**1.14**	4.57
		5%	*GPBA-A*	288.97	**137.33**	4.90	**10.97**	1.56	22.08
			GPBA-B	**149.33**	190.20	**9.73**	23.02	**1.15**	**8.80**
			GPBA-C	174.10	168.40	8.23	22.45	1.17	10.45
		4%	*GPBA-A*	333.23	**132.40**	4.43	**14.47**	1.67	27.59
			GPBA-B	198.90	182.97	6.97	26.20	**1.17**	**11.91**
			GPBA-C	**193.63**	163.53	**8.07**	30.09	1.19	**11.91**
		3%	*GPBA-A*	497.73	**112.27**	3.17	**15.59**	1.92	47.75
			GPBA-B	281.23	149.53	**6.13**	30.90	**1.20**	17.01
			GPBA-C	**269.53**	145.37	5.53	35.78	1.22	**16.69**
		2%	*GPBA-A*	637.53	**100.63**	2.67	**22.05**	2.32	75.36
			GPBA-B	428.90	131.37	4.40	36.87	1.25	28.83
			GPBA-C	**310.00**	140.83	**4.83**	53.39	**1.23**	**20.14**
		1%	*GPBA-A*	977.70	**76.97**	2.03	**23.08**	3.64	195.85
			GPBA-B	741.07	107.60	2.87	37.43	1.39	58.56
			GPBA-C	**404.50**	131.40	**3.87**	66.87	**1.30**	**28.51**

Table 8
Representation for multi-objective 0–1 knapsack instances with three objectives and one hundred items.

p	n	γ_k/δ_k	Algorithm	$\lvert\Pi(R(N))\rvert$	$\lvert\Gamma(R(N))\rvert$	$\lvert\Delta(R(N))\rvert$	$\lvert\%Ncov.\rvert$	$\overline{iter}/\hat{z}$	CPU(s)
3	100	30%	*GPBA-A*	16.80	**393.43**	55.93	**6.04**	1.39	1.21
			GPBA-B	**9.80**	564.00	**118.43**	12.08	**1.17**	**0.83**
			GPBA-C	13.10	457.87	72.77	12.44	1.22	1.04
		20%	*GPBA-A*	42.77	**315.07**	24.80	**5.22**	1.27	2.70
			GPBA-B	**19.20**	483.30	**57.73**	13.36	**1.14**	**1.32**
			GPBA-C	35.43	340.33	28.20	9.07	1.15	2.31
		15%	*GPBA-A*	51.60	**297.30**	19.57	**8.45**	1.26	3.34
			GPBA-B	**29.97**	420.97	**38.80**	14.70	**1.14**	**2.02**
			GPBA-C	39.07	332.03	29.90	14.34	1.15	2.65
		10%	*GPBA-A*	126.97	**223.17**	9.90	**7.06**	1.25	8.82
			GPBA-B	**58.87**	356.37	**22.03**	16.43	**1.12**	**3.77**
			GPBA-C	93.10	260.50	13.93	10.96	**1.12**	5.95
		5%	*GPBA-A*	352.90	**161.10**	5.07	**8.87**	1.36	27.70
			GPBA-B	**167.50**	240.13	**10.27**	19.83	**1.12**	**11.12**
			GPBA-C	232.87	197.87	7.77	15.96	1.13	15.99
		4%	*GPBA-A*	411.80	**158.67**	4.77	**11.70**	1.40	33.01
			GPBA-B	**233.13**	206.87	**7.93**	20.91	**1.12**	**15.63**
			GPBA-C	253.77	192.90	8.10	21.77	1.14	17.50
		3%	*GPBA-A*	687.63	**136.83**	3.40	**11.10**	1.53	62.76
			GPBA-B	**349.17**	191.07	**6.03**	24.17	**1.13**	**24.03**
			GPBA-C	425.83	166.93	4.57	22.26	1.17	30.69
		2%	*GPBA-A*	945.80	**126.90**	2.53	**17.00**	1.76	101.85
			GPBA-B	585.87	159.00	**4.03**	30.41	**1.17**	43.01
			GPBA-C	**499.13**	161.00	**4.03**	38.58	1.19	**37.50**
		1%	*GPBA-A*	1706.87	**96.67**	1.77	**23.85**	2.57	285.73
			GPBA-B	1197.53	121.40	2.63	38.70	1.27	100.55
			GPBA-C	**759.87**	140.60	**3.23**	60.58	**1.25**	**60.76**

4% rows). Finally, although *GPBA-C* never reaches coverage errors as low as *GPBA-A*, it is much more efficient, both in number of models solved per non-dominated criterion vector in the representation, and in computational time. Regarding the percentage of criterion vectors from the Pareto front that are not covered by the defined coverage level, *%Ncov*., *GPBA-A* obtains the best mean values in all experiments and with relatively low errors. Nevertheless, both *GPBA-B* and *GPBA-C* perform similarly on this indicator, both less satisfactorily than *GPBA-A*. Moreover, comparing the performance of *GPBA-B* and *GPBA-C* in light of the aforementioned indicator two considerations can be made: (1) *GPBA-C* while having a better coverage error than *GPBA-B*, in some cases leaves many points not covered at the desired coverage error; and (2) the fact that *GPBA-B* has a relatively low number of points not covered at the desired coverage error when comparing to the coverage error suggests that the reason for its underperformance in terms of coverage error is that whole areas of the Pareto front are undercovered.

4.5. Final comments and remarks

All the three algorithms were tested on experiments set for computing the whole Pareto front and a representation of it. Regarding the computation of the whole Pareto front:

1. All the three algorithms outperformed the *AUGM-2* algorithm. The reason for this is that *AUGM-2* only skips redundant solution and applies the early exit mechanism in the innermost loop. Furthermore, the redundant solution skip strategy is less sophisticated than the one applied on the other algorithms since it does not consider all past solutions, but just the last one. These factors imply that *AUGM-2* requires more models to be solved per non-dominated criterion vector computed.
2. Algorithms *GPBA-B* and *GPBA-C* outperform *GPBA-A* and *R-AUGM*. *GPBA-A*, due to the strategy used to determine the ϵ vector, does not have the early exit mechanism, lagging behind the other algorithms in terms of number of models solved per non-dominated criterion vector computed.
3. Algorithms *GPBA-B* and *GPBA-C* performance is comparable with *S-AUGM*, being amongst the best in the literature. This was to be expected since the acceleration mechanisms are common to the three algorithms. However, probably due to the model structure used, *S-AUGM* falls behind on integer knapsack instances with a low number of constraints (less than four). But, for general multi-objective integer instances, *S-AUGM* appears to be more efficient.

Regarding the representation of the Pareto front, the three algorithms were not compared with others in literature since, to the best of our knowledge, these are the first ϵ-constraint based algorithms to target the representation problem for integer problems with more than two objectives. Hence, the analysis of the proposed algorithms performance on the representation of Pareto front showed that:

1. *GPBA-A* was the algorithm with the best coverage error, while having the worst uniformity level.
2. *GPBA-B* had the best uniformity level, presenting the worst coverage error.
3. Contrary to what is observed for *GPBA-A* and *GPBA-B*, *GPBA-C* behaviour changes depending on the target cardinality and shows a behaviour balanced between *GPBA-A* and *GPBA-B*'s performances in all dimensions of the representation problem. When lower cardinality targets are used, *GPBA-C* appears to privilege coverage. Nevertheless, when higher cardinality targets are set, *GPBA-C* is unable to increase the cardinality of the representation and compromises on coverage while improving uniformity. This hybrid behaviour seems very convenient, since *GPBA-C* is more efficient than *GPBA-A*.

5. Conclusions

This work addresses the Pareto front representation problem, contributing with methodologies to explore the feasible region of the objective space. This is relevant for most multi-objective problems, since the evaluation of the whole Pareto front may be overwhelming for the decision-maker, potentially hindering the decision process. A good representation of the Pareto front consists of as few points as possible, covers all regions of the objective space and spreads the points the most. To tackle this problem three algorithms are presented: one aiming at coverage, a second at uniformity and a third at cardinality. All algorithms are based on the ϵ-constraint method and are insensitive to the quality of nadir point estimation.

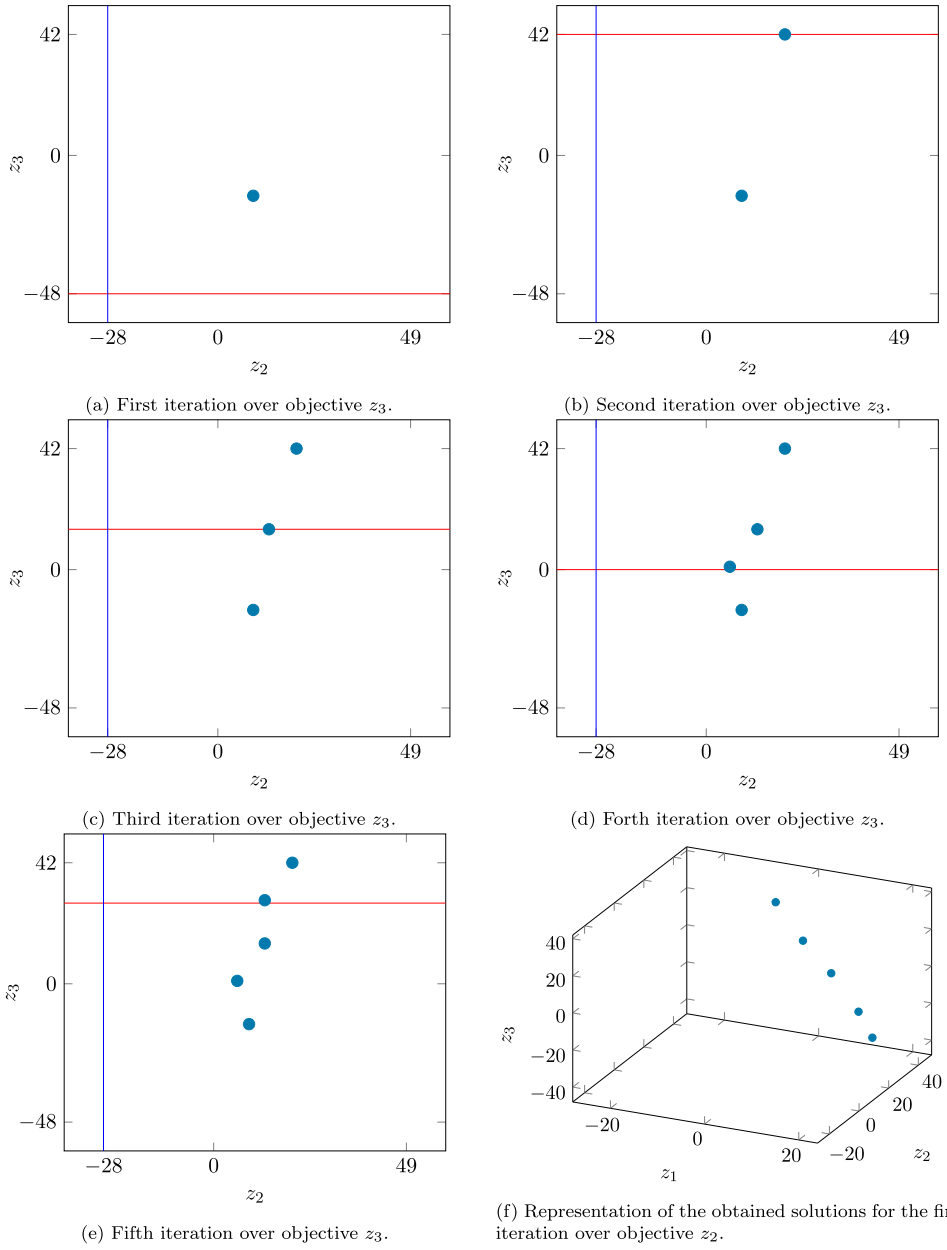
The algorithms were tested on the ability to efficiently obtain the Pareto front under 240 binary knapsack instances, 150 integer knapsack instances and 180 general problem instances, with three and four objectives. The algorithms that target uniformity and cardinality are demonstrated to be very efficient and among the best in literature. The algorithms were also tested on the quality of the obtained representation. Although the coverage and uniformity algorithms showed good results on their targeted measures, the cardinality algorithm for lower cardinalities presents a coverage error similar to the coverage algorithm, whilst not compromising much on uniformity and being much more efficient.

As future work we intend to study the incorporation of these representation strategies in interactive algorithms. Namely, we believe that the cardinality algorithm will provide a good basis in the sense that, in the first stage a good coverage of the Pareto front can be obtained and, at a later stage, as the DM's preferences are refined and the solution space more restricted, a representation of the selected area can be obtained with less effort by the coverage algorithm. Finally, we would like to apply these representation algorithms in real multi-objective problems that would benefit from being treated as such, without the computational complexity of computing the whole Pareto front and the difficulty of analysing it.

Acknowledgments

Mariana Mesquita-Cunha acknowledges the support by national funds through FCT, under the research grant SFRH/BD/149441/2019. José Rui Figueira acknowledges the support by national funds through FCT, under DOME research project, PTDC/CCI-COM/31198/2017. Ana Paula Barbosa-Póvoa and Mariana Mesquita-Cunha acknowledge the support by national funds through FCT, under the Data2Help research project, DSAIPA/AI/0044/2018, and the project 1801P.00740 PTDC/EGE-OGE/28071/2017 - LISBOA-01-0145-FEDER-028071. At last, this work is financed by national funds through the FCT - Foundation for Science and Technology, I.P., under the project UIDB/00097/2020. The authors would also like to thank Doctor Kostas Florios for the valuable comments and suggestions.

Appendix A. Figures of the illustrative example



(a) First iteration over objective z_3.

(b) Second iteration over objective z_3.

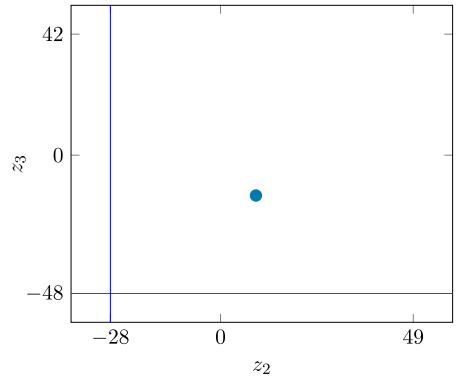
(c) Third iteration over objective z_3.

(d) Forth iteration over objective z_3.

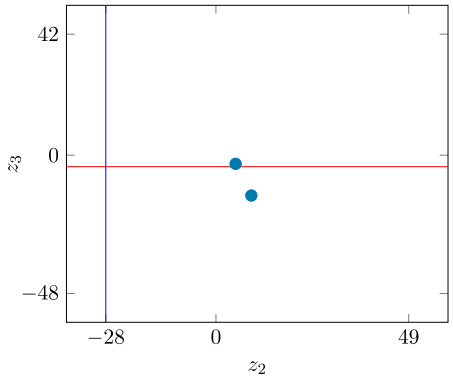
(e) Fifth iteration over objective z_3.

(f) Representation of the obtained solutions for the first iteration over objective z_2.

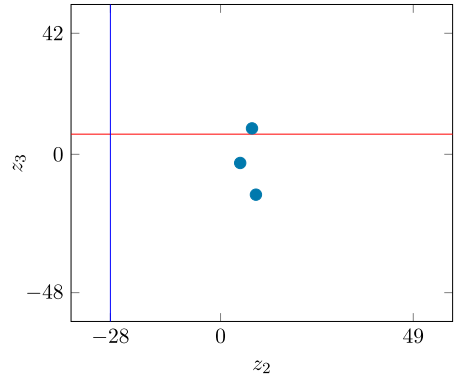
Fig. A.4. Illustrative example for the coverage representation, *GPBA-A*. The acceptable coverage error was defined as $\gamma = 15$ in each objective. Only the first iteration over the outermost loop is depicted. The blue line represents the ϵ_2 bound imposed over objective z_2, and the red line represents the ϵ_3 bound imposed over objective z_3.



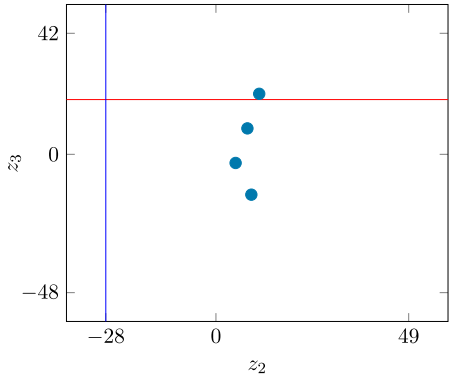
(a) First iteration over objective z_3.



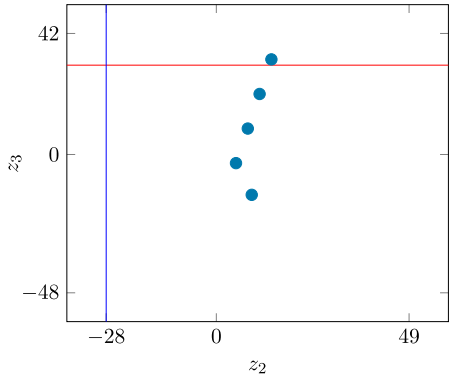
(b) Second iteration over objective z_3.



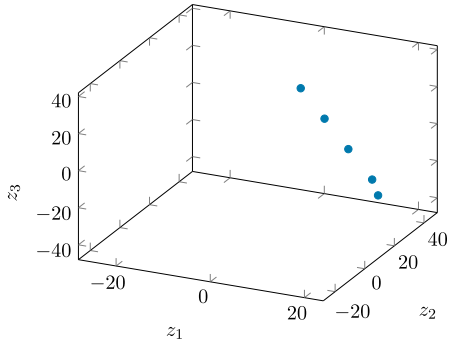
(c) Third iteration over objective z_3.



(d) Forth iteration over objective z_3.

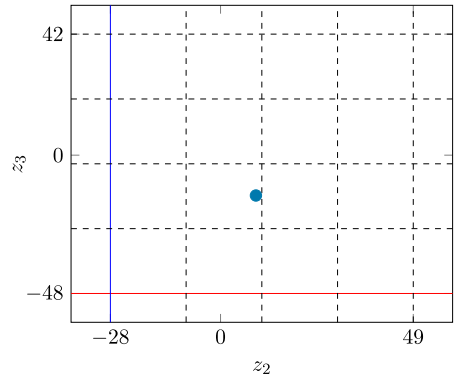


(e) Fifth iteration over objective z_3.

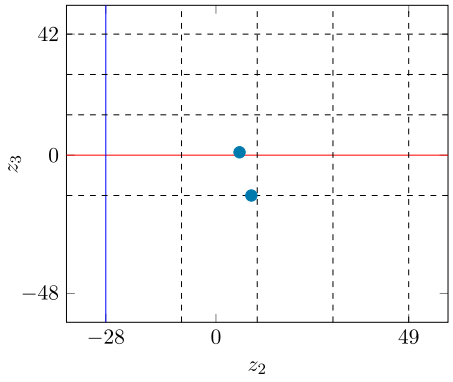


(f) Representation of the obtained solutions for the first iteration over objective z_2.

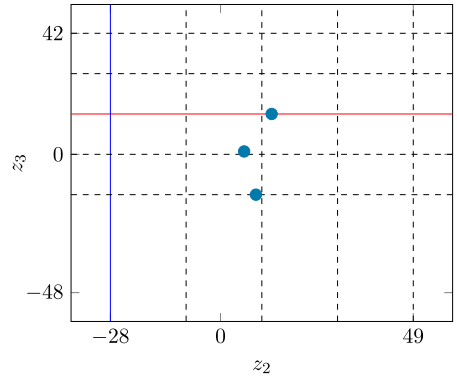
Fig. A.5. Illustrative example for the uniformity representation, *GPBA-B*. The uniformity level defined was $\delta = 10$ in all objectives. Only the first iteration over the outermost loop is depicted. The blue line represents the ϵ_2 bound imposed over objective z_2, and the red line represents the ϵ_3 bound imposed over objective z_3.



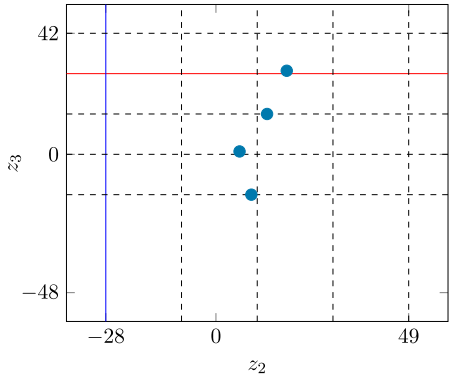
(a) First iteration over objective z_3.



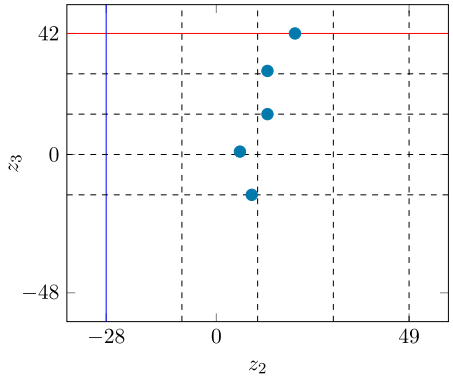
(b) Second iteration over objective z_3.



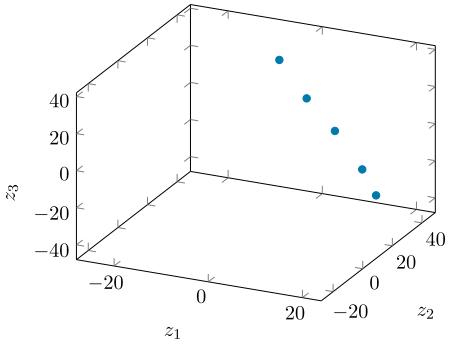
(c) Third iteration over objective z_3.



(d) Forth iteration over objective z_3.



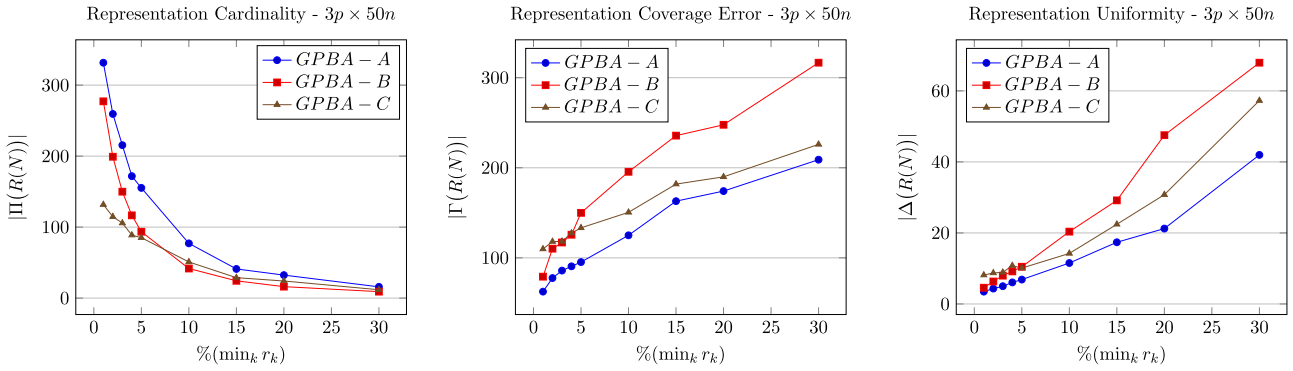
(e) Fifth iteration over objective z_3.



(f) Representation of the obtained solutions for the first iteration over objective z_2.

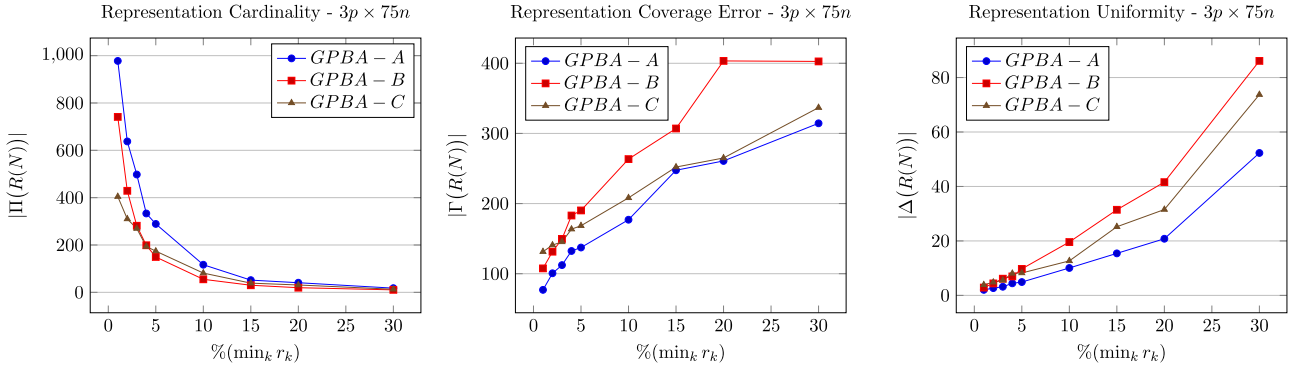
Fig. A.6. Illustrative example for the cardinality representation, *GPBA-C*. The desired cardinality defined was 5 in all objectives. Only the first iteration over the outermost loop is depicted. The blue line represents the ϵ_2 bound imposed over objective z_2, and the red line represents the ϵ_3 bound imposed over objective z_3.

Appendix B. Visualization of the results of experiments on representation



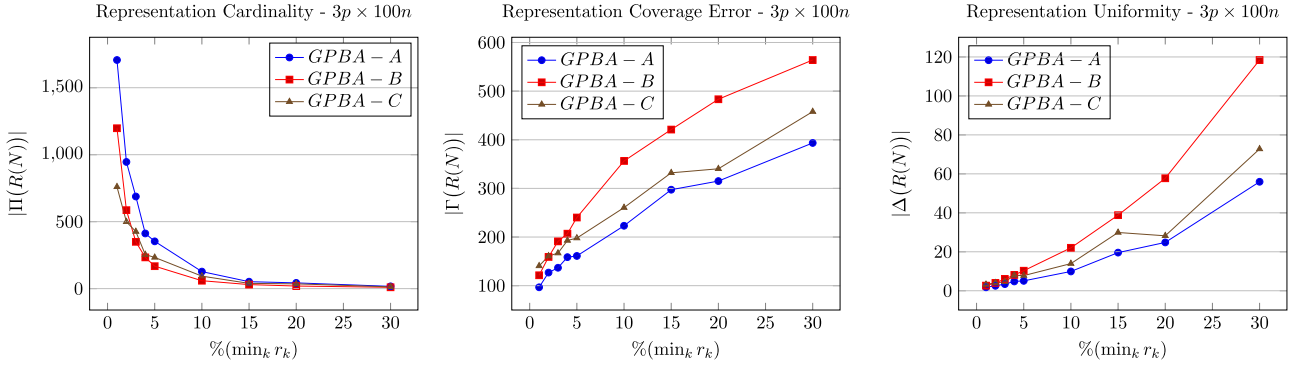
(a) Representation cardinality. (b) Representation coverage error. (c) Representation uniformity.

Fig. B.7. Impact of the control parameters' restriction on the representation problem's objectives for the proposed algorithms, when applying to the binary knapsack problem instances in Section 4.3.1 with fifty variables.



(a) Representation cardinality. (b) Representation coverage error. (c) Representation uniformity.

Fig. B.8. Impact of the control parameters' restriction on the representation problem's objectives for the proposed algorithms, when applying to the binary knapsack problem instances in Section 4.3.1 with seventy-five variables.



(a) Representation cardinality. (b) Representation coverage error. (c) Representation uniformity.

Fig. B.9. Impact of the control parameters' restriction on the representation problem's objectives for the proposed algorithms, when applying to the binary knapsack problem instances in Section 4.3.1 with one hundred variables.

References

Alves, M. J., & Clímaco, J. (2007). A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research, 180*(1), 99–115. https://doi.org/10.1016/j.ejor.2006.02.033.

Alves, M. J., & Costa, J. P. (2009). An exact method for computing the nadir values in multiple objective linear programming. *European Journal of Operational Research, 198*(2), 637–646. https://doi.org/10.1016/j.ejor.2008.10.003.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research, 292*(2), 397–422. https://doi.org/10.1016/j.ejor.2020.11.016.

Branke, J., Deb, K., Miettinen, K., & Slowiński, R. (2008). *Multiobjective optimization: Interactive and evolutionary approaches*. Berlin, Germany: Springer-Verlag. https://doi.org/10.1007/978-3-540-88908-3.

Ceyhan, G., Köksalan, M., & Lokman, B. (2019). Finding a representative nondominated set for multi-objective mixed integer programs. *European Journal of Operational Research, 272*(1), 61–77. https://doi.org/10.1016/j.ejor.2018.06.012.

Chalmet, L., Lemonidis, L., & Elzinga, D. (1986). An algorithm for the bi-criterion integer programming problem. *European Journal of Operational Research, 25*(2), 292–300. https://doi.org/10.1016/0377-2217(86)90093-7.

Chankong, V., & Haines, Y. Y. (2008). *Multiobjective decision making: Theory and methodology*. Mineola, NY, USA: Dover Publications Inc..

Coffin, M., & Saltzman, M. J. (2000). Statistical analysis of computational tests of algorithms and heuristics. *INFORMS Journal on Computing, 12*(1), 24–44. https://doi.org/10.1287/ijoc.12.1.24.11899.

Doğan, I., Lokman, B., & Köksalan, M. (2022). Representing the nondominated set in multi-objective mixed-integer programs. *European Journal of Operational Research, 296*(3), 804–818. https://doi.org/10.1016/j.ejor.2021.04.005.

Dächert, K., Klamroth, K., Lacour, R., & Vanderpooten, D. (2017). Efficient computation of the search region in multi-objective optimization. *European Journal of Operational Research, 260*(3), 841–855. https://doi.org/10.1016/j.ejor.2016.05.029.

Ehrgott, M. (2005). *Multicriteria optimization* ((2nd ed.)). Berlin, Germany: Springer-Verlag. https://doi.org/10.1007/3-540-27659-9.

Ehrgott, M., & Ruzika, S. (2008). Improved ε-constraint method for multiobjective programming. *Journal of Optimization Theory and Applications, 138*(3), 375. https://doi.org/10.1007/s10957-008-9394-2.

Ehrgott, M., & Ryan, D. M. (2003). The method of elastic constraints for multiobjective combinatorial optimization and its application in airline crew scheduling. In T. Tanino, T. Tanaka, & M. Inuiguchi (Eds.), *Multi-objective programming and goal programming: vol. 21* (pp. 117–122). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-540-36510-5_14.

Eusébio, A., Figueira, J., & Ehrgott, M. (2014). On finding representative nondominated points for bi-objective integer network flow problems. *Computers & Operations Research, 48*, 1–10. https://doi.org/10.1016/j.cor.2014.02.009.

Faulkenberg, S. L., & Wiecek, M. M. (2010). On the quality of discrete representations in multiple objective programming. *Optimization and Engineering, 11*(3), 423–440. https://doi.org/10.1007/s11081-009-9099-x.

Haimes, Y., Lasdon, L., & Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-1*(3), 296–297. https://doi.org/10.1109/TSMC.1971.4308298.

Isermann, H., & Steuer, R. E. (1988). Computational experience concerning payoff tables and minimum criterion values over the efficient set. *European Journal of Operational Research, 33*(1), 91–97. https://doi.org/10.1016/0377-2217(88)90257-3.

Kidd, M. P., Lusby, R., & Larsen, J. (2020). Equidistant representations: Connecting coverage and uniformity in discrete biobjective optimization. *Computers & Operations Research, 117*, 104872. https://doi.org/10.1016/j.cor.2019.104872.

Kirlik, G., & Sayın, S. (2015). Computing the nadir point for multiobjective discrete optimization problems. *Journal of Global Optimization, 62*(1), 79–99. https://doi.org/10.1007/s10898-014-0227-6.

Kirlik, G., & Sayın, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research, 232*(3), 479–488. https://doi.org/10.1016/j.ejor.2013.08.001.

Klamroth, K., Lacour, R., & Vanderpooten, D. (2015). On the representation of the search region in multi-objective optimization. *European Journal of Operational Research, 245*(3), 767–778. https://doi.org/10.1016/j.ejor.2015.03.031.

Klein, D., & Hannan, E. (1982). An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research, 9*(4), 378–385. https://doi.org/10.1016/0377-2217(82)90182-5.

Klingman, D., Napier, A., & Stutz, J. (1974). NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science, 20*(5), 814–821. https://doi.org/10.1287/mnsc.20.5.814.

Laumanns, M., Thiele, L., & Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research, 169*(3), 932–942. https://doi.org/10.1016/j.ejor.2004.08.029.

Masin, M., & Bukchin, Y. (2007). Diversity maximization approach for multiobjective optimization. *Operations Research, 56*(2), 411–424. https://doi.org/10.1287/opre.1070.0413.

Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation, 213*(2), 455–465. https://doi.org/10.1016/j.amc.2009.03.037.

Mavrotas, G., & Florios, K. (2013). An improved version of the augmented ϵ-constraint method (AUGMECON2) for finding the exact Pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation, 219*(18), 9652–9669. https://doi.org/10.1016/j.amc.2013.03.002.

Nikas, A., Fountoulakis, A., Forouli, A., & Doukas, H. (2020). A robust augmented ε-constraint method (AUGMECON-R) for finding exact solutions of multi-objective linear programming problems. *Operational Research*. https://doi.org/10.1007/s12351-020-00574-6.

Özlen, M., & Azizoğlu, M. (2009). Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research, 199*(1), 25–35. https://doi.org/10.1016/j.ejor.2008.10.023.

Ozlen, M., Burton, B. A., & MacRae, C. A. G. (2014). Multi-objective integer programming: An improved recursive algorithm. *Journal of Optimization Theory and Applications, 160*(2), 470–482. https://doi.org/10.1007/s10957-013-0364-y.

Sayın, S. (2000). Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming, 87*(3), 543–560. https://doi.org/10.1007/s101070050011.

Shao, L., & Ehrgott, M. (2016). Discrete representation of non-dominated sets in multi-objective linear programming. *European Journal of Operational Research, 255*(3), 687–698. https://doi.org/10.1016/j.ejor.2016.05.001.

Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation and application*. New York, NY, USA: John Wiley & Sons. https://doi.org/10.1002/oca.4660100109.

Sylva, J., & Crema, A. (2004). 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. https://doi.org/10.1016/S0377-2217(03)00255-8.

Sylva, J., & Crema, A. (2007). A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research, 180*(3), 1011–1027. https://doi.org/10.1016/j.ejor.2006.02.049.

Zhang, W., & Reimann, M. (2014). A simple augmented ϵ-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research, 234*(1), 15–24. https://doi.org/10.1016/j.ejor.2013.09.001.