This is a pre-copyedited version of an article published in European Journal of Operational Research. The final version of this article is available online at: https://doi.org/10.1016/j.ejor.2015.06.072

Graphical exploration of the weight space in three-objective mixed integer linear programs

Maria João Alves* and João Paulo Costa

Faculty of Economics of University of Coimbra / INESC - Coimbra, Portugal

Abstract

In this paper we address the computation of indifference regions in the weight space for multiobjective integer and mixed-integer linear programming problems and the graphical exploration of this type of information for three-objective problems. We present a procedure to compute a subset of the indifference region associated with a supported nondominated solution obtained by the weighted-sum scalarization. Based on the properties of these regions and their graphical representation for problems with up to three objective functions, we propose an algorithm to compute all extreme supported nondominated solutions adjacent to a given solution and another one to compute all extreme supported nondominated solutions to a three-objective problem. The latter is suitable to characterize solutions in delimited nondominated areas or to be used as a final exploration phase. A computer implementation is also presented.

Keywords: Multiple objective programming, mixed integer linear programming, weighted-sum scalarization, weight space, extreme supported nondominated solutions.

1. Introduction

A multiobjective integer or mixed-integer linear programming (MOMILP) problem with $p \geq 2$ objective functions can be written as:

$$\left.\begin{array}{l} \max z_1 = f_1(x) = c^1 x \\ \quad \ldots \\ \max z_p = f_p(x) = c^p x \end{array}\right\} \text{Max } z = f(x) = Cx$$

$$\text{s.t.} \quad x \in X = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0,\ x_j \in \mathbb{N}_0,\ j \in I\}$$

* Corresponding author. Tel. +351 239 790 558
Email address: mjalves@fe.uc.pt

where A is the $m \times n$ technological coefficients matrix, being all constraints transformed into equations by introducing appropriate slack or surplus variables, and $b \in \mathbb{R}^m$ is the right-hand-side vector. $I \subseteq \{1,...,n\}$, $I \neq \varnothing$ is the set of indices of the integer variables, with n the total number of variables (decision variables plus slack/surplus variables). It is assumed that X is bounded and non-empty. C is the $p \times n$ objective matrix whose rows are the vectors $c^k \in \mathbb{R}^n$, $k=1,...,p$.

If all decision variables are integer, then the multiobjective problem is pure integer (MOILP), which is a special case of the multiobjective mixed-integer case. In what follows we will denote by MOMILP the general case, in which integrality constraints are imposed on all or a subset of the decision variables.

A feasible solution $x' \in X$ is *efficient* if and only if there is no other solution $x \in X$ such that $f_k(x) \geq f_k(x')$ for all $k = 1,...,p$ and $f_k(x) > f_k(x')$ for at least one k. Let X_E denote the set of all efficient solutions.

Let $Z \subset \mathbb{R}^p$ be the image of the feasible region X in the objective space such that $Z = \{z \in \mathbb{R}^p: \ z = Cx, \ x \in X\}$. If $x' \in X$ is efficient, $z' = f(x') = Cx'$ is a *nondominated* criterion point. Let Z_{ND} be the set of all nondominated points, $Z_{ND} = \{z' \in Z : z' = Cx', x' \in X_E\}$.

An important concept in MOMILP is the distinction between *supported* and *unsupported* nondominated/efficient solutions.

A nondominated point $z' \in Z_{ND}$ is *supported* if it is located on the boundary of the convex hull of Z (*conv Z*). An *unsupported* nondominated point is located in the interior of *conv Z* (it is dominated by some convex combination of supported nondominated points). A supported (unsupported) nondominated point corresponds to a supported (unsupported) efficient solution.

We can further distinguish two types of supported nondominated points:

a) **e**xtreme **s**upported **n**on**d**ominated points $z \in Z_{ND}$, which are vertices of *conv Z*; we will denote these nondominated points /efficient solutions as ESND solutions;
b) non-extreme supported nondominated points, which are located in the relative interior of a face of *conv Z*.

Let Z_{ESND} denote the set of all ESND points/solutions in the objective space and X_{ESND} the corresponding set in the decision space.

Supported nondominated solutions are optimal solutions to the weighted-sum scalarization program (P_λ) for some weight vector $\lambda \in \Lambda = \{\lambda \in \mathbb{R}^p: \lambda_k > 0,\ k=1,...,p,\ \sum_{k=1}^{p} \lambda_k = 1\}$:

$$\max \quad z_\lambda = \sum_{k=1}^{p} \lambda_k f_k(x) = \lambda C x \qquad (P_\lambda)$$

$$\text{s.t.} \quad x \in X$$

Λ is usually called the *weight space* and the set of weight vectors that lead to the same nondominated solution is referred to as an *indifference region* in the weight space. The weights of the objective functions are the parameters in the (P_λ) scalarization program and the variation of parameters enables to attain different supported nondominated solutions. However, there are multiple parameter values that lead to the same solution, i.e. an indifference set on the parameter's space (weight space) can be defined for each supported nondominated solution.

The set Λ can be decomposed into subsets $\Lambda(z')$, $\forall z' \in Z_{ND}$ such that z' is supported, where $\Lambda(z')$ denotes the indifference region of z' in the weight space. It represents the set of weight vectors λ that lead to z' through the optimization of (P_λ), i.e., $\Lambda(z') = \{\lambda \in \Lambda: \lambda z' \geq \lambda z,\ \forall z \in Z_{ND}\}$. Indifference regions in the weight space are convex polytopes (Przybylski et al., 2010).

The optimization of (P_λ) using the branch-and-bound method yields (at least) an ESND solution. If there are alternative optima, a further exploration of the branch-and-bound tree allows for computing non-extreme supported nondominated solutions. However, unsupported nondominated solutions are never obtained through (P_λ) even if a complete parameterization is attempted and all alternative solutions for a given $\lambda \in \Lambda$ are analysed. The ESND points allow for the whole decomposition of the weight space into subsets $\Lambda(z')$, $z' \in Z_{ESND}$, because these, and only these points $z' \in Z_{ESND}$, correspond to indifference regions $\Lambda(z')$ of dimension $p-1$ (the dimension of Λ) of a MOMILP problem. Therefore, $\Lambda = \bigcup_{z' \in Z_{ESND}} \Lambda(z')$. Non-extreme supported nondominated points are associated with indifference regions of lower dimension resulting from the intersection of the regions of ESND points (these properties can be found in Przybylski et al., 2010).

Although the supported nondominated solutions (or even only the ESND solutions) constitute a subset of all nondominated solutions of the problem, they can provide important insights about the whole nondominated set because they are on the boundary (and the ESND are the vertices) of the convex hull of all nondominated points (Özpeynirci and Köksalan, 2010). Indifference regions in the weight space also constitute useful information for the decision maker. He/she may be indifferent to all weight combinations inside one region because they give rise to the same nondominated point.

An interactive graphical exploration of the weight space in multiobjective linear programming (MOLP) problems with three objective functions has been proposed in the TRIMAP method by Clímaco and Antunes (1987). The use of the weight space as a valuable means to gather information obtained from different interactive methods, and its graphical representation to present the information to the decision maker, has likewise been considered in other interactive MOLP computational tools (Antunes et al., 1992; Alves et al., 2015). Also considering MOLP problems, Benson and Sun (2002) proposed a weight set decomposition algorithm to generate all extreme nondominated points.

The computation of indifference regions in other parameter spaces has also been addressed. Costa and Clímaco (1999) related reference points (using achievement scalarizing functions) and weights in MOLP and defined indifference regions on the reference point space. Alves and Clímaco (2001) analysed the shape of indifference regions in the reference point space (in general, non-convex regions) for all-integer MOILP problems and proposed an approach to define indifference sets of reference points as long as a *directional search* procedure (Alves and Clímaco, 2000) is performed.

Concerning MOMILP problems, Przybylski et al. (2010) and Özpeynirci and Köksalan (2010) have exploited the weight space to design algorithms intended to generate all ESND points. These two approaches are reviewed in the next section.

In the present work we focus on ESND solutions of MOMILP problems and the exploration of their indifference regions in the weight space. We propose an approach that is able to compute a subset of an indifference region using the branch-and-bound tree that solved the weighted-sum scalarizing program (P_λ) for a given weight vector $\lambda \in \Lambda$. Acting alone, this approach rarely calculates the entire indifference region for the corresponding ESND solution, and the obtained sub-region may be much smaller than the full region. However, indifference regions can be iteratively enlarged using some

properties, namely convexity. Accordingly, we have developed a procedure that merges and expands sub-regions by building the convex hull of joined or disjoined sub-regions of the same solution. An indifference region can be enlarged not only from a merging process but also as a result of properties that relate adjacent regions of different solutions. We explore these properties for three objective problems, proposing an algorithm to compute all ESND solutions adjacent (in the weight space sense) to a known ESND solution or even to compute all ESND solutions of a three-objective problem. These features can naturally be applied to problems with two objective functions, but we will omit this case herein as it is straightforward. We also present a computer implementation in which the indifference regions are graphically depicted.

The rest of the paper is organized as follows. In section 2 the related work is reviewed. Section 3 introduces the technique to compute an indifference sub-region for an ESND solution and an illustrative example is shown. Section 4 gives the main principles to explore the weight space in MOMILP problems with three objective functions and describes the algorithms to compute the ESND solutions adjacent to a given solution and to compute all ESND solutions. Section 5 presents an overview of the computational implementation, illustrating the previous features using two examples, and presents computational experiments. The paper ends with some concluding remarks and future work in section 6.

2. Related work

Przybylski et al. (2010) proposed a recursive algorithm for finding Z_{ESND} which is based on the following idea: in each iteration, the weight space is completely decomposed with the solutions computed so far and the common facets of the regions are explored in order to compute new solutions and update the weight space decomposition. Let $S \subseteq Z_{ESND}$ denote the set of ESND points known at a given iteration. For each $z' \in S$ a *super-region* $\Lambda^+(z')$ is defined such that $\Lambda^+(z') = \{\lambda \in \Lambda: \lambda z' \geq \lambda z, \forall z \in S\}$. Then, the algorithm searches for new nondominated points at the boundaries of $\Lambda^+(z')$. For instance in three-objective problems, if z', $z'' \in S$ are adjacent in the current weight space decomposition, then $\Lambda^+(z') \cap \Lambda^+(z'')$ is a line segment; suppose that λ^1 and λ^2 are the extreme points of this edge; the algorithm investigates the edge by computing all the ESND points of the following bi-objective problem: $\max\{(f_1'(x), f_2'(x)) = (\lambda^1 Cx, \lambda^2 Cx) : x \in X\}$. Thus, a recursive

algorithm is developed. As new ESND points are added to S, the weight space decomposition is updated until all elements of Z_{ESND} have been found. At the end of the algorithm, the regions $\Lambda^+(z')$ are the real ones, i.e. $\Lambda^+(z') = \Lambda(z')$. The authors have further proved that a suitable initialization of the algorithm must contain the nondominated extreme points that optimize individually each objective function. This initialization enables to only explore the facets of each $\Lambda^+(z')$ that are not located on the boundary of Λ.

Özpeynirci and Köksalan (2010) proposed another algorithm with the same purpose of finding all points of Z_{ESND}. This algorithm does not use recursion, but it has a strong combinatorial component. The basic idea consists in introducing p dummy points in the objective space, $Z_m = \{m^k = Me^k, k = 1,\ldots,p\}$ where M is a large positive constant and e^k is the k^{th} unit vector. These points are infeasible and nondominated with respect to all points of Z_{ESND}. They have such characteristics that their indifference regions in the weight space touch all the boundary of Λ (where one of the weights is close to zero). The points m^k are incorporated into the search, which turns to be on $Z_{ESNDm} = Z_{ESND} \cup Z_m$. The authors prove that each point $z' \in Z_{ESND}$ is adjacent (in the weight space) to at least p points of Z_{ESNDm}. Denoting, as before, by S the set of ESND points identified so far, the algorithm successively selects different subsets of p points from $S \cup Z_m$ and, for each subset, defines the vector λ normal to the hyperplane that passes through these points. If all components of λ are positive, then the respective weighted-sum of the objectives (P_λ) is optimized in order to try to find a new ESND point or to conclude that the selected points define a facet of *conv* Z_{ESND}. The algorithm finishes when no more points or facets can be identified.

In both algorithms previously described, indifference regions in the weight space are not known during the intermediate stages of the algorithm since the weight space is decomposed using the current set of ESND points. This information is only available at the end, when all ESND solutions have been computed. Actually, the goal of these algorithms is not to compute indifference regions but rather to use this type of information as an instrumental means to generate all ESND solutions.

Indifference regions in the weight space are useful for an interactive search for new nondominated solutions. The decision maker may bypass all weight combinations within each region as all of them yield the same solution. With the purpose of obtaining this type

of information, we have developed an approach that is able to define a subset of an indifference region whenever an ESND solution is computed through the optimization of (P_λ). Although this is only a sub-region, it has the advantage that all the weight vectors included therein actually result in the same solution. This approach is described in the next section.

3. An approach to compute indifference sub-regions in the weight space for MOMILP problems

The approach we propose herein to compute indifference sub-regions in the weight space for MOMILP problems uses information provided by the *branch-and-bound* tree that solved (P_λ) combined with sensitivity analysis on the weighted-sum objective function. This analysis is performed in the terminal nodes (leaves) of the tree.

In MOLP, the indifference region corresponding to a basic efficient solution x' can be computed as follows (in, e.g., Yu and Zeleny, 1976): let x'_B be the basic variables and x'_N the non-basic variables of x'; let B and N be the sub-matrices of A corresponding to x'_B and x'_N, respectively; let C_B and C_N be the respective sub-matrices of C; the indifference region in the weight space associated with x' is defined by $\{\lambda \in \Lambda : \lambda(C_B B^{-1} N - C_N) \geq 0\}$ = $\{\lambda \in \Lambda : \lambda W \geq 0\}$ with $W = C_B B^{-1} N - C_N$ being the reduced cost matrix.

Consider now a MOMILP problem for which a weighted-sum program (P_λ) with $\lambda=\lambda^o$ has been optimized using the *branch-and-bound* method, yielding the efficient solution x^o. Let z^o be the corresponding non-dominated point: $z^o = Cx^o$. An indifference sub-region associated with this solution, say $\overline{\Lambda}(z^o)$, can be computed by inspecting the feasible terminal nodes of the *branch-and-bound* tree. The procedure is as follows:

Step 1. For each terminal node i of the *branch-and-bound* tree whose linear problem is feasible an indifference region R^i is computed as in MOLP. Let T denote the set of indices of the feasible terminal nodes. An intersection region of the R^i, for all $i \in T$, is considered: $R = \bigcap_{i \in T} R^i$

Step 2. Add to R the constraints resulting from the comparison of the weighted-sum value in the optimal node (node o) with respect to the other feasible terminal nodes of the tree (nodes $i \in T \backslash \{o\}$). Thus, $\overline{\Lambda}(z^o) = R \cap \{\lambda z^o \geq \lambda z^i, \ \forall i \in T \backslash \{o\}\}$, where $z^i = C x^i$ is the objective point for the solution x^i obtained in the node i.

Proposition 1. The set $\overline{\Lambda}(z^o)$ obtained by Steps 1 and 2 above is an indifference sub-region in the weight space for the nondominated point $z^o = Cx^o$ corresponding to the efficient solution x^o of the MOMILP problem.

Proof. The integer/mixed-integer linear program (Pλ) with $\lambda = \lambda^o$, say (P_{λ^o}), has been solved using the *branch-and-bound* method, yielding the solution x^o. Let the node o be the optimal node of the *branch-and-bound* tree. Each node i of the tree is associated with a linear sub-problem of (P_{λ^o}), say $(LP^i_{\lambda^o}): \{\max\ \lambda^o Cx : x \in X^i_{LP}\}$, where X^i_{LP} denotes the linear feasible region in the node i. Let T be the set of indices $\{i\}$ of the terminal nodes (leaves) of the tree for which $X^i_{LP} \neq \varnothing$. By the principles of the *branch-and-bound* method, none feasible solution $x \in X$ is excluded from $\bigcup_{i \in T} X^i_{LP}$. Let x^i be the current optimal solution of $(LP^i_{\lambda^o})$ for $i \in T$. The MOLP indifference region of x^i is $R^i = \{\lambda \in \Lambda: \lambda W^i \geq 0\}$, with W^i being the reduced cost matrix of node i. This means that x^i optimizes $(LP^i_{\lambda}): \{\max\ \lambda Cx : x \in X^i_{LP}\}$ for all $\lambda \in R^i$. Thus, $R = \bigcap_{i \in T} R^i$ is the set of the weight vectors λ that do not change any basic solution associated with the feasible terminal nodes of the current tree, i.e., for all $i \in T$, the corresponding x^i optimizes (LP^i_{λ}) for all $\lambda \in R$.

In addition, if, for a given $\lambda \in R$, the value of the objective function of (LP^o_{λ}) (node o, with solution x^o) is better than the value of the objective function of any other (LP^i_{λ}), $i \in T$, then the *branch-and-bound* tree remains unchanged with x^o optimal to (Pλ). This is valid for any $\lambda \in R$ such that $\lambda Cx^o \geq \lambda Cx^i$, $\forall i \in T \backslash \{o\}$. Hence, x^o is an optimal solution of (Pλ) for all $\lambda \in \overline{\Lambda}(z^o) = R \cap \{\lambda\ z^o \geq \lambda\ z^i,\ \forall i \in T \backslash \{o\}\}$, with $z^i = Cx^i$, which means that $\overline{\Lambda}(z^o)$ is an indifference sub-region in the weight space for (x^o, z^o). ■

In sum, Step 1 ensures that the included λ-vectors do not change the basic solutions associated with each feasible terminal node of the current tree. Step 2 restricts $\overline{\Lambda}(z^o)$ to λ-vectors for which the node o remains the optimal one for (Pλ), i.e. provides a weighted-sum value larger or equal to the values of all the other terminal nodes.

Due to the normalization $\lambda_1 + \lambda_2 + \ldots + \lambda_p = 1$, the weight space can be represented in a $p-1$ dimensional diagram. The following example illustrates the procedure described above using a small integer problem with 3 objective functions. The indifference sub-regions

are shown in the two-dimensional projection (λ_1, λ_2) in which the origin (λ_1, λ_2) = (0,0) represents the point where λ_3=1.

Example:

$$\begin{aligned}
&\max \quad z_1 = x_1 - x_2 \\
&\max \quad z_2 = x_1 + 2x_2 \\
&\max \quad z_3 = -x_1 + 2x_2 \\
&\text{s.t:} \quad x_1 + 6x_2 \leq 21 \\
&\qquad 14x_1 + 6x_2 \leq 63 \\
&\qquad x_1, x_2 \geq 0 \text{ and integer}
\end{aligned}$$

The problem's decision space is depicted in fig. 1, where all efficient solutions are marked with a larger dot and their variable values are shown. Solutions x^A, x^B, x^C and x^D are the extreme supported efficient solutions. Solutions x = (3, 1) and x = (3, 2) are unsupported, and x = (1, 3) and x = (2, 3) are non-extreme supported solutions.

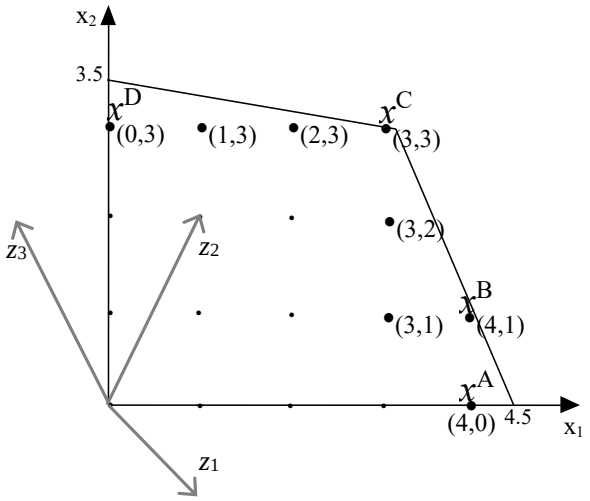
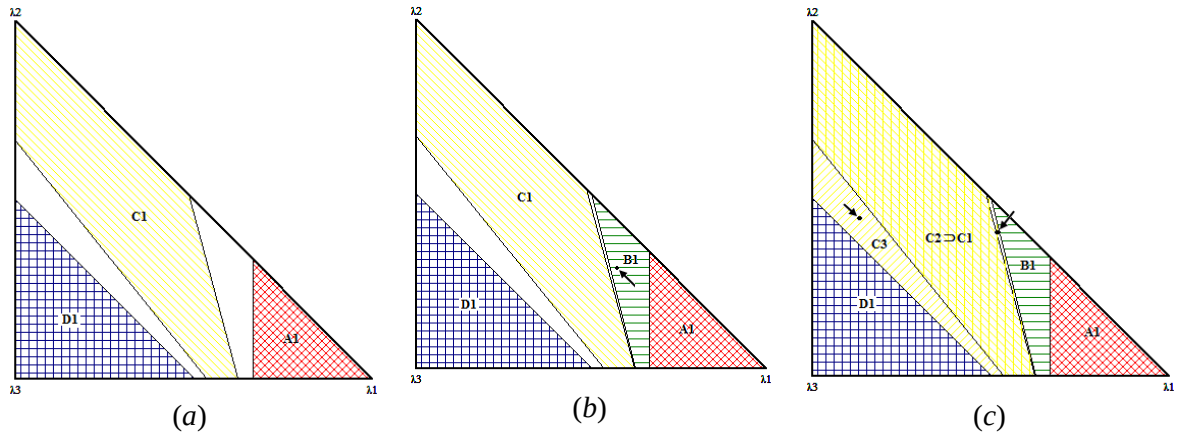


Fig. 1 – Representation of the *Example* problem in the decision space.

Firstly, consider that the weight vectors λ^1 = (0.99, 0.005, 0.005), λ^2 = (0.005, 0.99, 0.005) and λ^3= (0.005, 0.005, 0.99) are chosen in order to compute efficient solutions that optimize individually each objective function. These are solutions x^A, x^C and x^D, respectively. The respective indifference sub-regions in the weight space calculated by the procedure described above are the polygons A1, C1 and D1 shown in fig. 2(*a*).

Let us now consider that other weight vectors are selected from the unfilled areas of the weight diagram. If λ^3 = (0.57, 0.28, 0.14) is chosen, the point indicated by the arrow in fig. 2(*b*), then solution x^B is computed and the indifference sub-region B1 is obtained (see

fig.2(*b*)). Next, $\lambda^4 = (0.52, 0.4, 0.08)$ is chosen and, after that, $\lambda^5 = (0.13, 0.44, 0.43)$, which are indicated by arrows in fig. 2(*c*), respectively on the right and on the left of C1. Both weight vectors lead to solution x^C and the resulting indifference sub-regions are C2 (which contains C1) and C3, represented in fig. 2(*c*). As we can observe in fig. 2(*c*), the weight space is now completely filled, so $\{x^A, x^B, x^C, x^D\}$ constitutes the set of all ESND solutions of the problem. The whole indifference regions in the weight space of these solutions are, respectively, A1, B1, C1∪C2∪C3= C2∪C3 and D1.



(*a*) (*b*) (*c*)

Fig. 2 - Indifference (sub)-regions in the weight space computed for the problem of the *Example*.

Three indifference sub-regions were computed for solution x^C because this solution was obtained from different *branch-and-bound* trees. In order to better illustrate the procedure described above to compute the indifference sub-regions and, in particular, the case of solution x^C, we show in fig.3(*a*) and fig.3(*b*) the *branch-and-bound* trees that produced regions C2 and C3, respectively.

Consider the construction of C2. Let R_a^o and R_a^1 be the MOLP indifference regions corresponding to the nodes *o* and 1 of the tree in fig. 3(*a*), which have been computed according to step 1 of the procedure. In addition, step 2 defines the constraint $\lambda z^o \geq \lambda z^1$ $\Leftrightarrow 0\lambda_1 + 9\lambda_2 + 3\lambda_3 \geq 3\lambda_1 + 6\lambda_2 - 2\lambda_3 \Leftrightarrow -3\lambda_1 + 3\lambda_2 + 5\lambda_3 \geq 0$.

Thus, $C2 = R_a^o \cap R_a^1 \cap \{-3\lambda_1 + 3\lambda_2 + 5\lambda_3 \geq 0\}$.

$C3 = R_b^o$, where R_b^o is the MOLP indifference region of the node *o* of the tree in fig. 3(*b*) – step 1. No other feasible terminal node exists, so the step 2 is not applied.

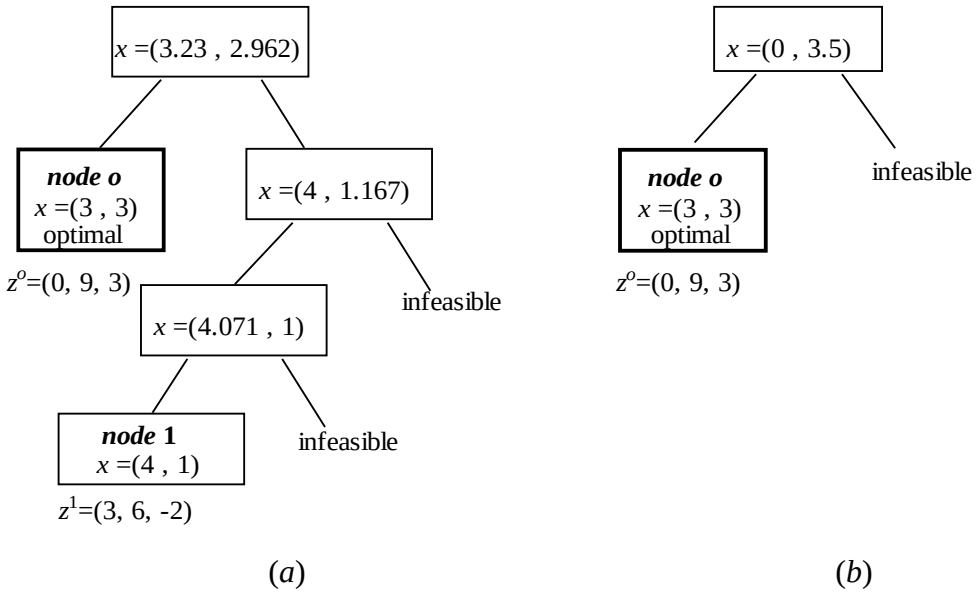


Fig. 3 – Branch-and-bound trees that (*a*) solved (P_λ) with λ^4=(0.52, 0.4, 0.08) producing region C2 and (*b*) solved (P_λ) with λ^5=(0.13, 0.44, 0.43) producing region C3.

4. Exploring the weight space in three-objective problems

The approach described above is able to compute a subset of an indifference region using information from the *branch-and-bound* tree that solved the weighted-sum scalarizing program (P_λ) for a given weight vector $\lambda \in \Lambda$. This approach may compute a small region when compared with the complete indifference region of the ESND solution. Fortunately, some properties enable to enlarge sub-regions when other (P_λ) are solved leading to the same solution or to neighbouring solutions. We will now present these properties.

Firstly, indifference regions in the weight space are always *convex*. This property enables to merge sub-regions computed for the same solution by defining the convex hull of their union. We have implemented a QuickHull algorithm for $\mathbb{R}^2$ (Bykat, 1978; Eddy, 1977) to perform this *merging* task for three-objective problems, in which the indifference regions are polygons. Figure 4 illustrates the merging process. We omit herein the bi-objective case as it is straightforward.

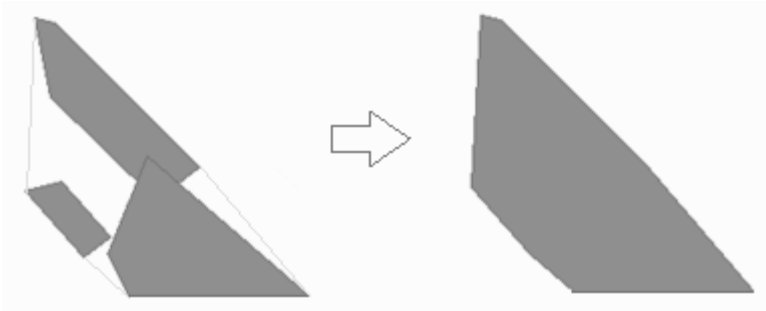


Fig. 4 – Merging indifference sub-regions for the same solution.

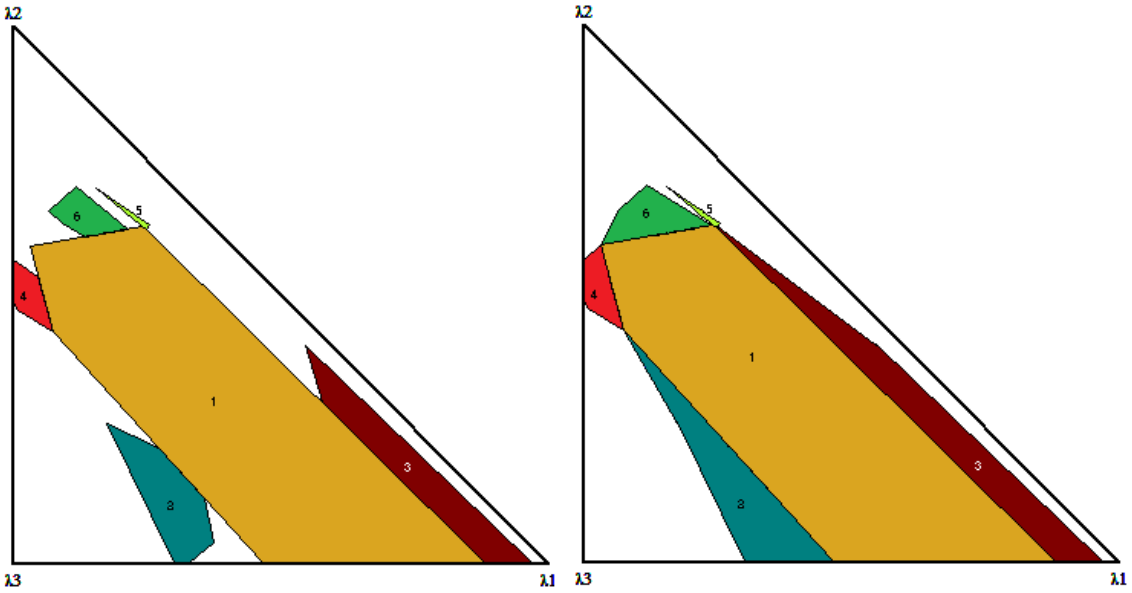
Secondly, suppose that we want to explore the surrounding area of an already known ESND solution. Let z^1 be its nondominated criterion point and $\overline{\Lambda}(z^1)$ the indifference sub-region currently known for z^1. If we choose a weight vector outside the region, but close to its limits, and optimize the respective weighted-sum, one of two situations may occur: z^1 is obtained again and the indifference sub-regions can be merged, redefining $\overline{\Lambda}(z^1)$; or another ESND solution is computed, say z^2, and the respective indifference sub-region $\overline{\Lambda}(z^2)$ is defined. If a weight vector for each edge of the current $\overline{\Lambda}(z^1)$ is investigated and no other property is used, the undesirable situation as the one illustrated in fig. 5(*a*) can arise: the polygon labelled with 1, $\overline{\Lambda}(z^1)$, has 6 edges; apart from the one that touches the boundary of the weight space, all the other 5 edges were explored by choosing a weight vector close to the midpoint of the edge and optimizing the respective weighted-sum; 5 different ESND solutions were obtained whose indifference sub-regions are shown in fig. 5(*a*); the figure seems to show that there are adjacencies to $\overline{\Lambda}(z^1)$ that have not been explored. Fortunately, the next proposition enables us to conclude that all ESND solutions adjacent to z^1 have already been found and their indifference regions can be enlarged as shown in fig. 5(*b*).

Proposition 2. Let $\lambda^0, \lambda^1, \lambda^2 \in \Lambda$ be three weight vectors that are located on the same line segment. Let the nondominated criterion points z^a and z^b optimize (P_λ) for both λ^0 and λ^1. Let z^a also optimize (P_λ) for λ^2. Then z^b also optimizes (P_λ) for λ^2.

Proof. To prove that z^b optimizes (P_λ) for λ^2, it is sufficient to prove that z^b provides the same weighted-sum value as z^a, i.e. $\lambda^2 z^b = \lambda^2 z^a$.
The equation of the line passing through $\lambda^0, \lambda^1, \lambda^2$ can be defined as $\lambda = \lambda^0 + t(\lambda^1 - \lambda^0)$ with t a parameter. Thus, $\lambda^2 = \lambda^0 + t^2(\lambda^1 - \lambda^0)$.

$\lambda^2 z^b = [\lambda^0 + t^2(\lambda^1 - \lambda^0)] z^b = \lambda^0 z^b + t^2\lambda^1 z^b - t^2\lambda^0 z^b = \lambda^0 z^a + t^2\lambda^1 z^a - t^2\lambda^0 z^a = [\lambda^0 + t^2(\lambda^1 - \lambda^0)]z^a = \lambda^2 z^a$. So, z^b is an alternative optimal solution to z^a when (P_λ) is optimized for λ^2. ■

Proposition 2 enables to conclude that, if an indifference polygon intersects another polygon in part of an edge, then the whole line segment that defines the edge belongs to both indifference regions (as illustrated in fig. 5, from (*a*) to (*b*)). Although formalized in a different way, Przybylski et al. (2010) proved a related result, showing that two adjacent indifference regions intersect in a maximal dimensional face.



(*a*) (*b*)

Fig. 5 – Using *Proposition* 2 to obtain (*b*) from (*a*).

Hence, when adjacencies are being explored, different situations may result in expanded indifference sub-regions. Consider that moving out from an edge of $\overline{\Lambda}(z^1)$ leads to solution z^2 with indifference sub-region $\overline{\Lambda}(z^2)$. Either $\overline{\Lambda}(z^2)$, $\overline{\Lambda}(z^1)$ or both may be enlarged as illustrated in fig. 6. In addition, if the same solution is found, $\overline{\Lambda}(z^1)$ grows by merging the two sub-regions and the adjacent regions may also be expanded, as illustrated in fig. 7: moving out from edge [AB] of $\overline{\Lambda}(z^1)$, we obtain again z^1 and a new indifference sub-region $\overline{\Lambda}'(z^1)$ is defined; then, $\overline{\Lambda}'(z^1)$ and $\overline{\Lambda}(z^1)$ are merged and, by Proposition 2, $\overline{\Lambda}(z^2)$ can be enlarged too.

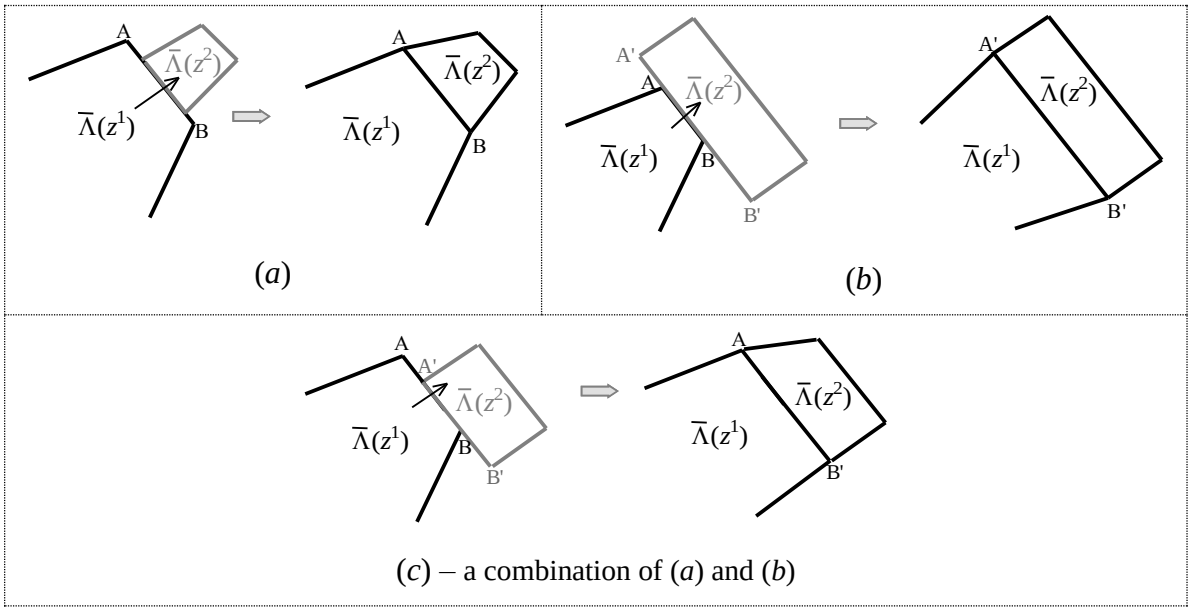


Fig. 6 – Examples of enlarging indifference sub-regions when the investigation of an edge leads to a different solution.

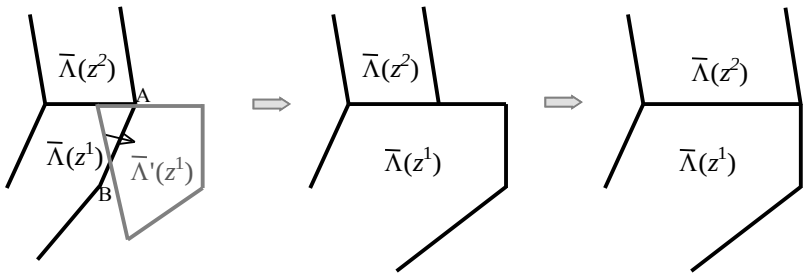


Fig. 7 – Example of enlarging indifference sub-regions when the investigation of an edge leads to the same solution.

The *Merge & Expand* iterative procedure to handle the merging of regions and the expansion impacts on the adjacent regions can be summarized as follows. Note that the expansion process repeats in a recursive way, because an enlarged region may produce a change in an adjacent region, which in turn changes another region adjacent to the latter, and so on:

Case I) If an edge of $\overline{\Lambda}(z^1)$ is investigated and a new solution z^2 is found (as in fig. 6), the edge of $\overline{\Lambda}(z^1)$ under analysis ([AB] in fig. 6) is added to $\overline{\Lambda}(z^2)$ and these are merged to form the convex hull. $\overline{\Lambda}(z^2)$ may be enlarged (fig. 6(*a*) and (*c*)) or not (fig. 6(*b*)). In any case, all the regions adjacent to $\overline{\Lambda}(z^2)$ are analysed to check whether they can be expanded (e.g. the expansion of $\overline{\Lambda}(z^1)$ in fig. 6(*b*) and (*c*)). Whenever a region is expanded, the other regions that intersect it in a modified

edge are then analysed. An iterative process occurs, which ends when no more region is modified.

Case II) If an edge of $\overline{\Lambda}(z^1)$ is investigated and the same solution is found (as in fig. 7), then the process starts by merging the indifference sub-regions of z^1. The expanding procedure continues by analysing the regions adjacent to the new joined $\overline{\Lambda}(z^1)$ following a similar scheme as in case I.

In the following we propose an algorithm for three-objective problems that computes the ESND solutions adjacent to a given ESND solution z^1 according to the following definition.

Definition 1 (Przybylski et al., 2010): Two ESND points z^1 and z^2 are *adjacent* if and only if $\Lambda(z^1) \cap \Lambda(z^2)$ is a polytope of dimension $p-2$.

The algorithm to find the ESND solutions adjacent to z^1 explores iteratively all *non-explored* edges of a sub-region $\overline{\Lambda}(z^1)$. This sub-region has been obtained from the branch-and-bound tree that optimized (P_λ) with a given λ (e.g. chosen by the user) that yielded z^1 (as described in Section 3). For each edge, a weight vector outside $\overline{\Lambda}(z^1)$, near and central to that edge, is selected. If the edge is not on the boundary of the weight space and the picked weight vector is *unknown*, the corresponding weighted-sum program (P_λ) is optimized and either a new solution (z^2) or the same one (z^1) is obtained. A weight vector is *known* if it belongs to the indifference region of another known solution, say z^3. The *Merge & Expand* procedure is then run, starting with z^2 (Case I above) if a new solution has been computed, z^1 (Case II) if the same solution has been obtained or z^3 (Case I) if the adjacent solution to z^1 on that edge is already known. The edge analysed is marked as *explored* and labelled with the index of the adjacent solution if a different solution has been found (the label is 0 in case of an edge on the boundary of the weight space). New edges defined by the *Merge & Expand* procedure are marked as *non-explored*. The steps of the algorithm to find all ESND solutions adjacent to z^1 are formalized in Algorithm 1, where *Adj* denotes the set of ESND criteria points found by the algorithm. Algorithm 1 and its subroutines are presented in a sufficiently general form so that they can also be used by Algorithm 2, presented afterwards, which computes all ESND solutions to the problem.

Initialize: Mark all edges of $\overline{\Lambda}(z^1)$ as *non-explored.*

Algorithm 1: *Adjacent-ESND* (z^1)

$Adj \leftarrow \varnothing$

While there is some *non-explored* edge of $\overline{\Lambda}(z^1)$ **do**

Step 1: Select a *non-explored* edge *i* of $\overline{\Lambda}(z^1)$, say $\overline{\Lambda}^i(z^1)$

If $\overline{\Lambda}^i(z^1)$ is on the boundary of Λ **then**

mark $\overline{\Lambda}^i(z^1)$ as *explored* with label 0

Else go to *Step 2*

End If

Step 2: Define a λ^i outside $\overline{\Lambda}(z^1)$, central and close to $\overline{\Lambda}^i(z^1)$ – call *Define_*λ^i ($\overline{\Lambda}^i(z^1)$)

Step 3: **If** $\lambda^i \notin \overline{\Lambda}(z^k), \forall k$ **then** // *unknown weight vector*

Optimize (P_λ) with $\lambda=\lambda^i$. Let (x^*, z^*) be the optimal solution and $\overline{\Lambda}(z^*)$ the respective indifference sub-region. Mark all edges of $\overline{\Lambda}(z^*)$ as *non-explored*.

If $z^* = z^1$ **then** // *Step 3.1: the same solution was obtained*

$\overline{\Lambda}(z^1) \leftarrow \overline{\Lambda}(z^1) \cup \overline{\Lambda}(z^*)$

Merge & Expand ($\overline{\Lambda}(z^1)$)

Else //*Step 3.2: another solution was computed*

Let k+1 be the index of the (new) obtained solution: $(x^{k+1}, z^{k+1}) = (x^*, z^*)$

If *Confirm_Adjacency* (z^1, z^{k+1}) **then**

$Adj \leftarrow Adj \cup \{z^{k+1}\}$

$\overline{\Lambda}(z^{k+1}) \leftarrow \overline{\Lambda}(z^*) \cup \overline{\Lambda}^i(z^1)$

Mark $\overline{\Lambda}^i(z^1)$ in $\overline{\Lambda}(z^1)$ as *explored* with label k+1

Mark $\overline{\Lambda}^i(z^1)$ in $\overline{\Lambda}(z^{k+1})$ as *explored* with label 1

Merge & Expand ($\overline{\Lambda}(z^{k+1})$)

Else go to *Step 2*

End If

End If

Else // *Step 3.3: the weight vector λ^i belongs to a known indifference region*

Let $\tilde{k}$ be the solution index such that $\lambda^i \in \overline{\Lambda}(z^{\tilde{k}})$

If *Confirm_Adjacency* $(z^1, z^{\tilde{k}})$ **then**

$Adj \leftarrow Adj \cup \{ z^{\tilde{k}} \}$

$\overline{\Lambda}(z^{\tilde{k}}) \leftarrow \overline{\Lambda}(z^{\tilde{k}}) \cup \overline{\Lambda}^i(z^1)$

Mark $\overline{\Lambda}^i(z^1)$ in $\overline{\Lambda}(z^1)$ as *explored* with label $\tilde{k}$

Mark $\overline{\Lambda}^i(z^1)$ in $\overline{\Lambda}(z^{\tilde{k}})$ as *explored* with label 1

Merge & Expand ($\overline{\Lambda}(z^{\tilde{k}})$)

Else go to *Step 2*

End If

End If

End While

Output: *Adj*, their indifference sub-regions $\{ \overline{\Lambda}(z^a) : z^a \in Adj \}$ and the complete indifference region of z^1, $\Lambda(z^1) = \overline{\Lambda}(z^1)$.

Now, let us detail how λ^i is defined in *Step 2* of the Algorithm 1 – procedure *Define_*λ^i – and its validation in *Confirm_Adjacency*.

The region $\overline{\Lambda}(z^1)$ is a polygon defined by a list of vertices (V_0, V_1,…,V_v) in anticlockwise order. The edges of the region $\overline{\Lambda}(z^1)$ are: [V_0V_1], [V_1V_2],…, [$V_{v-1}V_v$], [V_vV_0]. Let the edge under investigation $\overline{\Lambda}^i(z^1)$ be represented by [$V_{i0}V_{i1}$].

Procedure *Define_*λ^i ([$V_{i0}V_{i1}$]**)**

Considering the representation of the weight space in a projection (λ_1, λ_2), the vertex points V_{i0} and V_{i1} are: $V_{i0} = (\lambda_1^{i0}, \lambda_2^{i0})$ and $V_{i1} = (\lambda_1^{i1}, \lambda_2^{i1})$.

A point P' = (λ'_1, λ'_2) outside $\overline{\Lambda}(z^1)$ and close to the midpoint P of $\overline{\Lambda}^i(z^1)$ is defined as:

P' = P + $\varepsilon(\lambda_2^{i1} - \lambda_2^{i0}, \lambda_1^{i0} - \lambda_1^{i1})$ with P = $((\lambda_1^{i0} + \lambda_1^{i1})/2, (\lambda_2^{i0} + \lambda_2^{i1})/2)$ and ε a small positive value.

P' corresponds to $\lambda^i = (\lambda'_1, \lambda'_2, 1 - \lambda'_1 - \lambda'_2)$.

Output: λ^i

Since the vertices are in anticlockwise order, the expression above defines a point outside $\overline{\Lambda}(z^1)$. Besides that, an accurate value of ε must be used. Initially, ε is set to a predefined small positive value (e.g. 0.01). If a solution different from z^1, say z', is obtained for λ^i in Algorithm 1, either by optimizing the respective weighted-sum (*Step 3.2*) or because λ^i has been found to belong to $\overline{\Lambda}(z')$ already known (*Step 3.3*), then it is checked whether z' is really adjacent to z^1 in the *Confirm_Adjacency* test. In negative case, the algorithm decreases ε and returns to *Step 2*, calling again *Define_*λ^i. This process ensures that no gaps remain between indifference regions and the solutions yielded are surely adjacent to z^1.

Procedure *Confirm_Adjacency* (z^1, z'**)**

Let λ^{i0}, λ^{i1} be the weight vectors corresponding to the vertices V_{i0} and V_{i1}, respectively, that define the edge $\overline{\Lambda}^i(z^1)$ under exploration.

If $\lambda^{i0}z^1 = \lambda^{i0}z'$ and $\lambda^{i1}z^1 = \lambda^{i1}z'$ **then**

    Confirm_Adjacency ← true

Else

    Confirm_Adjacency ← false

End If

Output: *Confirm_Adjacency*

The *Merge & Expand* procedure starts with a region $\overline{\Lambda}(z)$ and then propagates the changes to the adjacent regions of $\overline{\Lambda}(z)$, continuing to the adjacent regions of the latter, and so on, in a recursive manner that stops when no more change has to be propagated. Therefore, the parameter $\overline{\Lambda}(z)$ of the routine *Merge & Expand* in Algorithm 1 just indicates the starting region. In order to provide a more formalized description of this procedure, let us first introduce some notation:

For each ESND point z' already known, the current definition of $\overline{\Lambda}(z')$ may include several subsets that have to be merged to form the respective convex hull. Let us denote by $\overline{\Lambda}_h(z')$, h =1,…,H' each subset of $\overline{\Lambda}(z')$ before de merging process.

Procedure *Merge & Expand*

Step 1: Select a region $\overline{\Lambda}(z') = \left\{\overline{\Lambda}_1(z'),..., \overline{\Lambda}_{H'}(z')\right\}$ such that $H' > 1$.
If $H' = 1$ for all known z′ **then** Stop.

Step 2: Apply the QuickHull procedure to $\left\{\overline{\Lambda}_1(z'),..., \overline{\Lambda}_{H'}(z')\right\}$ which joins the subsets and form its convex hull, obtaining $\overline{\Lambda}^{new}(z')$ with H'^{new}=1. $\overline{\Lambda}^{new}(z')$ is a polygon defined by a sequence of vertices (V_0, V_1,…, V_v) in anticlockwise order. Each edge of $\overline{\Lambda}^{new}(z')$ without correspondence in any $\overline{\Lambda}_h(z')$, h =1,…, H' is marked as *non-explored*; any edge of $\overline{\Lambda}^{new}(z')$ existing in some $\overline{\Lambda}_h(z')$ keeps its status and the label (if it is marked as *explored*). Let $\mathcal{L}$ be the list of solution indexes $l\neq 0$, $z^l \neq z'$ that were labels of explored edges in $\overline{\Lambda}_h(z')$, h =1,…,H' but do not figure as labels of edges of $\overline{\Lambda}^{new}(z')$ because the edges have changed. These z^l are adjacent solutions to z' not yet assigned to $\overline{\Lambda}^{new}(z')$.

Step 3: **For** each $l \in \mathcal{L}$ **do**

Search for an edge of $\overline{\Lambda}^{new}(z')$ that intersects $\overline{\Lambda}(z^l)$ by inspecting two consecutive vertices of $\overline{\Lambda}^{new}(z')$, say V_{i0}, V_{i1}, for which $\lambda^{i0}z' = \lambda^{i0}z^l$ and $\lambda^{i1}z' = \lambda^{i1}z^l$. Let this edge be denoted by $\overline{\Lambda}^{new_i}(z')$.

- Mark $\overline{\Lambda}^{new_i}(z')$ as *explored* with label l
- **If** $\overline{\Lambda}^{new_i}(z')$ does not coincide with any edge of $\overline{\Lambda}(z^l)$ **then**
  $\overline{\Lambda}(z^l) \leftarrow \overline{\Lambda}(z^l) \cup \overline{\Lambda}^{new_i}(z')$
  $H^l \leftarrow H^l + 1$
  End If
- Mark $\overline{\Lambda}^{new_i}(z')$ in $\overline{\Lambda}(z^l)$ as *explored* with label ' (the index of z')

End For
Return to *Step 1*.

When the *Merge & Expand* procedure is called from Algorithm 1, the first selection of a region $\overline{\Lambda}(z')$ such that $H' > 1$ is provided as parameter. This procedure can also be used alone to merge indifference regions for the same solution obtained from independent optimizations of weighted-sums.

Proposition 3. Algorithm 1 computes all different ESND criteria points adjacent to the ESND z^1 for a three-objective MOMILP problem.

Proof. Any point z found by the Algorithm 1 is adjacent to z^1 because the condition of *Definition* 1 is ensured by the *Confirm_Adjacency* test.

Suppose that there exists an ESND point z^b adjacent to z^1 that was not found by the Algorithm 1. By definition, $\Lambda(z^1) \cap \Lambda(z^b)$ is a polytope of dimension p–2, i.e., an edge.

Let us denote this edge of $\Lambda(z^1)$ by $\Lambda^i(z^1)$. Since the algorithm only finishes after all edges of $\Lambda(z^1)$ have been explored, suppose that $\Lambda^i(z^1)$ has been marked as *explored* with label *a*, which means that z^a was the adjacent solution found from the investigation of this edge. So, either $\Lambda(z^b) = \Lambda^i(z^1)$ of dimension p-2 instead of p-1, and thus z^b is not an ESND point (Przybylski et al., 2010 – Proposition 4), or $\Lambda(z^b)$ includes $\Lambda^i(z^1)$ and, at least, another weight vector, say $\lambda^3 \notin \Lambda^i(z^1)$, belonging to the interior of either $\Lambda(z^1)$ or $\Lambda(z^a)$. Consider λ^1, λ^2 two different weight vectors of $\Lambda^i(z^1)$. Assume that $\lambda^3 \in \Lambda(z^a)$. So, λ^1, λ^2 and λ^3 belong to the indifference region of z^a. Consider the following system of linear equations with variables z: $\begin{cases} \lambda^1 z = \lambda^1 z^a \\ \lambda^2 z = \lambda^2 z^a \\ \lambda^3 z = \lambda^3 z^a \end{cases}$. Since the matrix of the system is full rank, then the system has only one solution: $z = z^a$. Thus, there is no other solution whose indifference region shares three weight vectors with $\Lambda(z^a)$. Hence, $z^b = z^a$. Analogously, if $\lambda^3 \in \Lambda(z^1)$ it can be concluded that $z^b = z^1$. This contradicts the hypothesis that z^b was not found. ■

The algorithm to find *all* ESND criteria points (Z_{ESND}) of a three-objective MOMILP problem is a straightforward extension of Algorithm 1, in which adjacent solutions for all ESND points are explored. This is formalized in Algorithm 2.

Algorithm 2

// *Find Z_{ESND} starting with an empty set*
Choose an arbitrary $\lambda^1 \in \Lambda$.
Optimize (P_λ) with $\lambda = \lambda^1$. Let (x^1, z^1) be its optimal solution and $\overline{\Lambda}(z^1)$ the obtained indifference sub-region.
Mark all edges of $\overline{\Lambda}(z^1)$ as *non-explored*.
$Z_{ESND} = \{z^1\}$
$NA = \{z^1\}$ *// set of solutions not yet analysed*
While $NA \neq \emptyset$ **do**
  Select a $z' \in NA$
  $NA \leftarrow NA \setminus \{z'\}$
  $Adj \leftarrow$ ***Adjacent-ESND*** **(z')**
  For each $z^a \in Adj$ such that $z^a \notin Z_{ESND}$ **do**
    $NA \leftarrow NA \cup \{z^a\}$
  End For
  $Z_{ESND} \leftarrow Z_{ESND} \cup Adj$
End While

Output: All solutions of Z_{ESND} and their complete indifference regions in the weight space, $\Lambda(z)$ for $\forall z \in Z_{ESND}$.

This work has been mainly developed for three-objective problems. We have also implemented analogous algorithms for two-objective problems, although they are omitted

herein because they are much simpler. Some of these features can also be extended to more than three objective functions. The way indifference sub-regions are computed (presented in Section 3) is valid for any number of objective functions. The Quickhull procedure (Bykat, 1978, Eddy, 1977) used in the *Merge & Expand* procedure is specific for $\mathbb{R}^2$ but an extension to higher dimensions could be implemented, e.g. using the algorithm of Barber et al. (1996). Other instructions in this procedure would need to be adapted for higher dimensions in order to deal with faces of dimension $p-2$ (instead of edges) of $(p-1)$-dimensional polytopes. Also, faces of dimension $p-2$ would be iteratively explored in Algorithm 1 (requiring a suitable definition of λ^i in *Define_*λ^i) and the *Confirm_Adjacency* procedure would need to test the equality of the weighted-sum value of the two candidate solutions for $p-1$ linearly independent weight vectors. However, the major purpose of these features is to allow the decision maker to perform a graphical exploration of the weight space and this would hardly be achieved in problems with more than three objective functions.

5. Computational implementation

The procedures described in Sections 3 and 4 were implemented in Delphi XE5 for Windows. The graphical representation of the indifference regions in the weight space is available for problems with two or three objective functions, as well as the algorithms to compute the ESND solutions adjacent to a previously computed solution or to compute all ESND solutions of the problem. We will denote these algorithms by *Adjacent-ESND* and *All-ESND*, respectively. In case of the *All-ESND* algorithm, the user can run it after computing one or several ESND solutions or from the beginning when no solution has been computed (as in Algorithm 2). In the latter situation, the implemented algorithm starts by computing a solution that optimizes the weighted-sum (P_λ) with equal weights: $\lambda_k = 1/p$, $k = 1,\ldots,p$.

Note that, in a computer implementation with floating-point arithmetic, roundoff errors may introduce small differences between $\lambda^I z^1$ and $\lambda^I z'$ for two adjacent criterion points z^1 and z' and λ^I a weight vector common to both indifference regions of z^1 and z'. Therefore, for numerical reasons, the *Confirm_Adjacency* procedure, which checks if $\lambda^I z^1 = \lambda^I z'$, $I = i_0, i_1$, considers a tolerance δ (a positive number close to 0), i.e. $\left|\lambda^I z^1 - \lambda^I z'\right| \le \delta$, $I = i_0, i_1$, where λ^{i0} and λ^{i1} are the weight vectors corresponding to the

vertices of the edge $\overline{\Lambda}^i(z^1)$ whose analysis led to z'. Even considering a small value for δ, in problems with a large number of ESND points it may exist some solution between z^1 and z' with a very narrow indifference region. So, we can state that the implemented algorithm calculates all adjacent solutions with a tolerance δ. In the results reported herein we have considered a relative tolerance given by $\delta = 10^{-5} \times \left|\lambda^I z^1\right|$ when $\left|\lambda^I z^1 - \lambda^I z'\right| \leq \delta$ is checked.

As noticed before, the main purpose of these tools is to provide a graphical exploration of the weight space in an interactive decision process, namely exploring neighbourhoods of solutions (using the *Adjacent-ESND*). This process may culminate with the *All-ESND* algorithm after computing several ESND solutions (either by isolated optimizations of weighted-sums or by the *Adjacent-ESND*) in problems with a small or moderate number of ESND solutions.

To illustrate these features, we present two examples, the first one using an integer problem and the second one using a mixed-integer problem. Next, we provide some insight into the evolution of the performance of the algorithm when the number of variables (integer and/or continuous) or the number of constraints is increased. The experiments were done in a computer with an Intel Core i7-2600K CPU@3.4GHz and 8 GB RAM. The weight space pictures were taken directly from the software.

5.1. Example 1

Consider the following three-objective integer problem with 10 binary variables and 6 constraints:

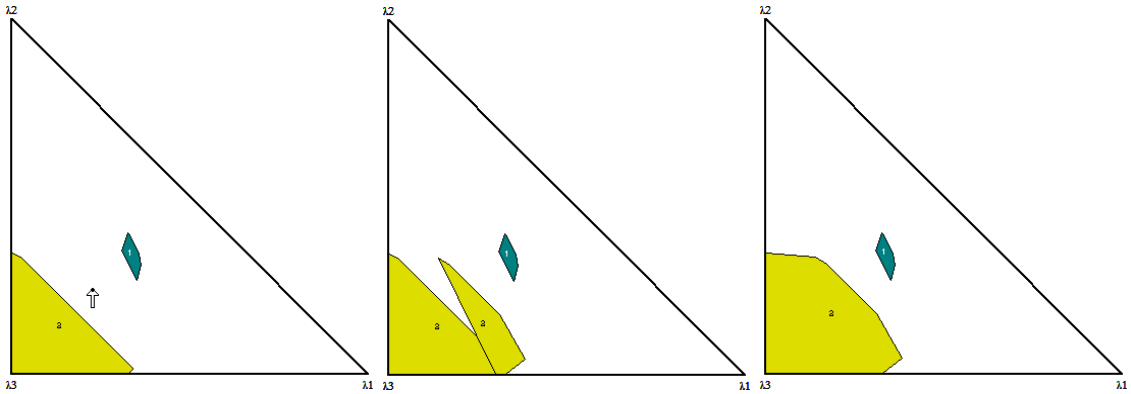
$$\begin{aligned} &\text{Max } z = Cx \\ &s.t.\ Ax \leq b \\ &\quad x_j \in \{0,1\},\ j = 1,...,10 \end{aligned}$$

$$\text{with } C = \begin{bmatrix} 60 & 19 & 0 & 0 & 0 & 73 & 18 & 29 & 96 & 72 \\ 66 & 11 & 33 & 27 & 50 & 48 & 0 & 72 & 78 & 39 \\ 0 & 3 & 17 & 25 & 86 & 49 & 31 & 15 & 51 & 93 \end{bmatrix},$$

$$A = \begin{bmatrix} 7 & 0 & 0 & 22 & 35 & 12 & 0 & 6 & 6 & 32 \\ 8 & 15 & 0 & 22 & 0 & 16 & 35 & 49 & 23 & 46 \\ 23 & 0 & 0 & 16 & 41 & 23 & 0 & 20 & 0 & 15 \\ 49 & 16 & 0 & 26 & 0 & 21 & 36 & 6 & 13 & 46 \\ 28 & 43 & 10 & 11 & 38 & 17 & 22 & 1 & 32 & 4 \\ 11 & 47 & 44 & 10 & 37 & 32 & 34 & 1 & 44 & 46 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 386 \\ 144 \\ 108 \\ 152 \\ 248 \\ 250 \end{bmatrix}.$$

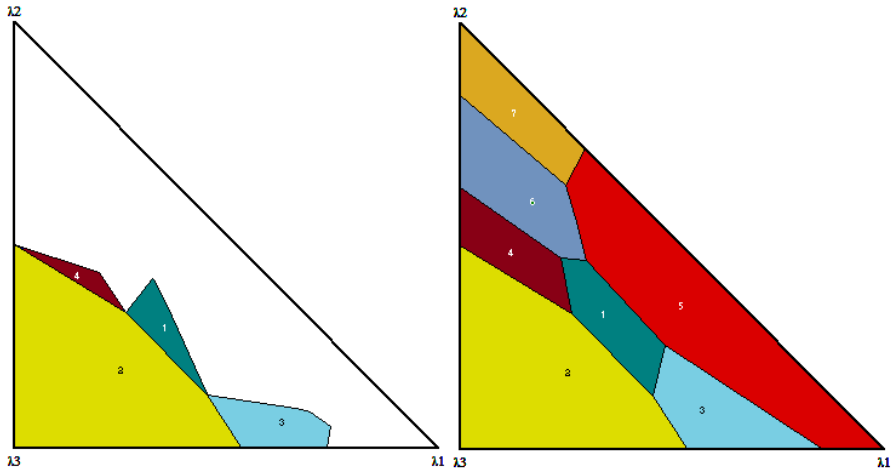
Suppose that some weighted-sums with different weight vectors are firstly optimized and the corresponding sub-indifference regions are computed: considering the weight vector $\lambda^1 = (1/3, 1/3, 1/3)$, the nondominated solution $z^1 = (301, 314, 296)$ is obtained; next, $\lambda^2 = (0.1, 0.1, 0.8)$ is chosen, which leads to $z^2 = (259, 275, 352)$. The corresponding indifference sub-regions are shown in fig. 8(a) (polygons labelled as 1 and 2, respectively).

The user is offered the possibility of inputting the weight values through a dialog box or selecting weights by clicking on a point in the unfilled area of the weight-space graph. Let us suppose that, using the latter option, the user points the weight vector given by the arrowhead in fig. 8(*a*), which corresponds to $\lambda^3 = (0.229, 0.234, 0.537)$. The optimization of this weighted-sum returns the already known solution z^2 and another indifference sub-region for this solution is computed - fig. 8(*b*). The two sub-regions known for solution 2 are merged, building the one presented in fig. 8(*c*).



(*a*) (*b*) (*c*)

Fig. 8 – *Example* 1: (*a*) computing solutions 1 and 2; (*b*) another indifference sub-region of solution 2; (*c*) merging the indifference sub-regions of solution 2.



(*a*) (*b*)

Fig. 9 – *Example* 1: (*a*) computing the ESND solutions adjacent to sol. 2 and (*b*) computing all extreme supported nondominated solutions through *All-ESND*.

Now, suppose that the decision maker wants to explore the neighbourhood of solution z^2. The *Adjacent-ESND* algorithm is called and the new solutions $z^3 = (320, 292, 282)$ and $z^4 = (270, 320, 311)$ are produced. The corresponding sub-regions are shown in fig. 9(*a*). At this stage, if *All-ESND* is executed, then three additional nondominated solutions are obtained whose objective points are $z^5 = (330, 336, 225)$, $z^6 = (276, 349, 265)$ and $z^7 = (277, 358, 221)$. The weight-space becomes completely filled, as can be seen in fig. 9(*b*). Hence, this problem has 7 extreme supported nondominated solutions.

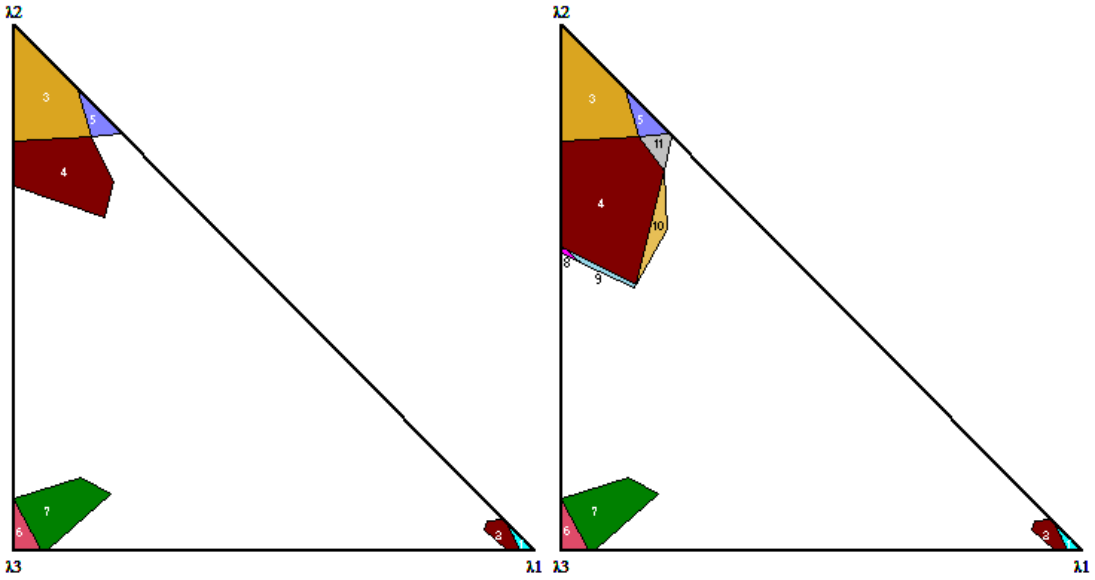
5.2. Example 2

Let us now consider a multiobjective mixed-integer problem with 3 objective functions, 20 variables (10 binary and 10 continuous variables) and 10 constraints. The data of the problem is provided in the *Appendix*.

Suppose that the decision maker wishes to start the search by computing the nondominated solutions that optimize individually each objective function and explore their neighbourhood by looking for the ESND solutions adjacent to each of these solutions. In order to ensure that nondominated solutions are generated from the individual optimizations, zero weights are replaced with small positive weights. Thus, the weight vector $\lambda^1 = (0.99, 0.005, 0.005)$ is firstly considered and solution 1 is obtained, which has one adjacent solution: solution 2 (fig. 10(*a*)). Then, $\lambda^2 = (0.005, 0.99, 0.005)$ is considered and solution 3 is obtained; its adjacent solutions are solutions 4 and 5 (fig. 10(*a*)). $\lambda^3 = (0.005, 0.005, 0.99)$ leads to solution 6, which has one adjacent solution: solution 7 (fig. 10(*a*)).

Assuming that, next, the decision maker wants to know the adjacent solutions to solution 4, the *Adjacent-ESND* algorithm is called for this solution. It returns solutions 8, 9, 10 and 11 and the corresponding indifference sub-regions shown in fig. 10(*b*). It is worth noting that, when all adjacent solutions are computed to a given solution, we get to know the complete indifference region for that solution. Therefore, we can see in fig. 10(*b*) the complete indifference regions for solutions 1, 3, 4 and 6.

Finally, if the decision maker wants to know all ESND solutions of the problem and calls the *All-ESND* algorithm, a total of 49 nondominated solutions become known whose objective values are presented in the *Appendix*. Fig. 11 shows the resulting decomposition of the weight space.



(*a*) (*b*)

Fig. 10 - *Example* 2: (*a*) computing the solutions that optimize individually each objective function and their neighbours; (*b*) computing the ESND solutions adjacent to solution 4.

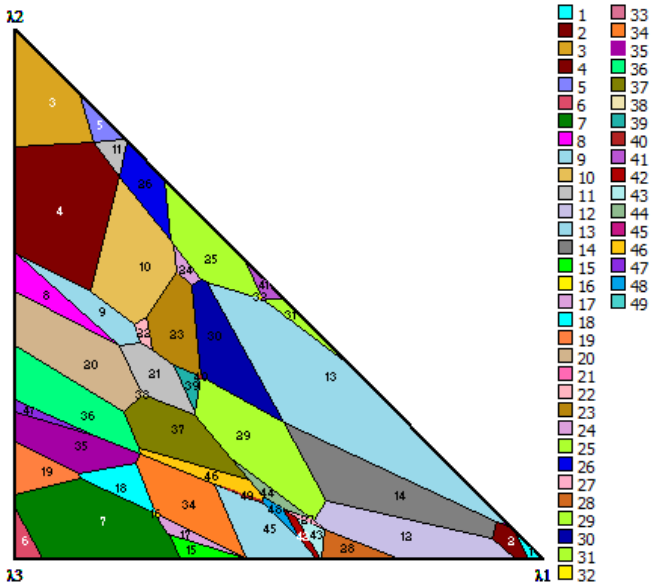


Fig. 11 – *Example 2*: Indifference regions in the weight space for all ESND solutions of the problem.

5.3. Computational experiments

Preliminary results have shown that the computational effort to compute the adjacent solutions of a given solution may vary substantially from one solution to another, even within the same problem. Therefore, in order to provide some general indications on the evolution of the computational effort with the increase of the number of variables and/or constraints, we have opted to show results of computing all ESND solutions of each problem (using Algorithm 2), although this is not the main purpose of this work. All computational times reported herein include the time spent not only in the search but also in graphical and display routines, including showing solution values, drawing indifference regions and updating other bar/scatter charts concerning the objective space.

For these computational experiments we used mixed-integer problems with continuous and binary variables. We considered the problem of *Example 2* (in Appendix) without bounds on the continuous variables as a basis to generate other instances. This problem (with 10 binary, 10 continuous variables and 10 constraints) is denoted by *Prob0*.

Firstly, we considered the Group 1 of problems in which the number of binary variables (B) was increased to 20, 30 and 50, keeping the data of *Prob0* for the existent variables, number of constraints and their right-hand-sides. These are *Prob1.1*, *Prob1.2* and *Prob1.3*, respectively. *Prob1.1* was created from *Prob0* by generating coefficients for the new ten binary variables, *Prob1.2* was created from *Prob1.1*, and so on. The coefficients in the objective functions (c_{ij}) and in the constraints (a_{ij}) are integer numbers that were

randomly generated in the following ranges (where *P* means probability): $\begin{cases} c_{ij} = 0 & P = 0.2 \\ 1 \le c_{ij} \le 100 & P = 0.8 \end{cases}$, $\begin{cases} a_{ij} = 0 & P = 0.2 \\ 1 \le a_{ij} \le 50 & P = 0.8 \end{cases}$. These rules had already been observed in *Prob0* and were also used in the problems of the next groups.

In Group 2 the number of continuous variables (C) of *Prob0* was increased to 20, 30 and 50, while keeping the number of binary variables equal to 10. These instances are *Prob2.1*, *Prob2.2* and *Prob2.3*, respectively. *Prob2.1* was created from *Prob0* and each of the following ones was created from its predecessor.

Group 3 increased simultaneously the number of binary and continuous variables, joining the data of problems in Groups 1 and 2 to form three instances with 20B+20C, 30B+30C and 50B+50C variables (*Prob3.1*, *Prob3.2* and *Prob3.3* respectively). As before, the number of constraints and their right-hand-sides remained the same.

In order to test problems with different feasible ranges, Group 4 changed the right-hand-sides (RHS) of the constraints of the problems in Group 3. The RHS were randomly generated between 10*n* and 90*n* (*n* being the number of decision variables): $n = 40$ in *Prob4.1*, $n = 60$ in *Prob4.2* and $n = 100$ in *Prob4.3*. Problems in Group 5 increased the number of constraints to 20, 30, 50, respectively to each problem of Group 4. The RHS were generated as in Group 4 and the same rules of the other groups were applied to generate the a_{ij} coefficients. All the constraints are of the type "≤". Finally, for illustrative purposes, we also report a few results for problems with 2 objective functions. We considered the problems of Group 3 without the third objective function. This is Group 3_2.

In the *Examples 1* and *2* presented above, we considered $\varepsilon = 0.01$ as the basic (predefined) step for setting λ^i in the procedure *Define_*λ^i (called in *Step 2* of Algorithm 1). If a higher value for ε is used, positive and negative effects may exist in the number of iterations: on one hand, the number of times that the algorithm finds the same solution (*Step 3.1*) may decrease; on the other hand, more often the reduction of ε may be required in order to verify adjacency of different solutions (*Steps 3.2* and *3.3*), which may increase the number of iterations. For the computational experiments we considered two values of ε: 0.01 and 0.05. The results are shown in Table 1, which include the number of ESND solutions of each problem ($|Z_{ESND}|$), the total number of weighted-sum optimizations (TOpt), the time

(in seconds) spent to compute all solutions and the average number of optimizations per solution ($\overline{\text{Opt}}$) given by the ratio TOpt/ $|Z_{ESND}|$.

Table 1. – Results of the computational experiments

						ε = 0.05			ε = 0.01		
		n									
Group	Prob.	(B)	(C)	*m*	$\|Z_{ESND}\|$	TOpt	time [s]	$\overline{\text{Opt}}$	TOpt	time [s]	$\overline{\text{Opt}}$
	Prob0	10	10	10	57	134	1.4	2.4	149	1.6	2.6
G 1	*Prob1.1*	20	10	10	98	827	15.5	8.4	894	17.3	9.1
	Prob1.2	30	10	10	96	1190	25.6	12.4	1352	29.8	14.1
	Prob1.3	50	10	10	81	2850	74.3	35.2	3629	94.1	44.8
G 2	*Prob2.1*	10	20	10	68	93	0.9	1.4	99	1.0	1.5
	Prob2.2	10	30	10	96	149	1.9	1.6	150	1.7	1.6
	Prob2.3	10	50	10	142	197	2.2	1.4	204	2.2	1.4
G 3	*Prob3.1*	20	20	10	132	274	5.9	2.1	293	6.3	2.2
	Prob3.2	30	30	10	205	1379	49.7	6.7	1552	54.4	7.6
	Prob3.3	50	50	10	312	2282	98.3	7.3	2756	98.7	8.8
G 4	*Prob4.1*	20	20	10	105	159	1.8	1.5	167	1.8	1.6
	Prob4.2	30	30	10	181	204	1.6	1.1	206	1.7	1.1
	Prob4.3	50	50	10	217	378	7.8	1.7	397	8.9	1.8
G 5	*Prob5.1*	20	20	10	399	795	27.2	2.0	826	28.3	2.1
	Prob5.2	30	30	30	292	629	29.4	2.2	679	30.6	2.3
	Prob5.3	50	50	50	598	1666	142.3	2.8	1846	156.3	3.1
G 3_2	*Prob3.1_2*	20	20	10	14	16	0.1	1.1	15	0.1	1.1
(*p=2*)	*Prob3.2_2*	30	30	10	21	43	0.2	2.0	43	0.2	2.0
	Prob3.3_2	50	50	10	25	36	0.2	1.4	37	0.2	1.5

In Group 1 we observe that the increase of the number of binary variables (keeping the right-hand-sides of the constraints, which make them tight to many variables) does not reflect a growing trend of the number of ESND solutions. However, the number of optimizations increases significantly and a similar trend is observed in the computational effort. On the other side, the increase of the number of continuous variables in Group 2 slightly increases the number of ESND solutions with a small increment in the computational effort. The number of optimizations keeps almost proportional to the number of solutions (about 1.5). The computational time is short, e.g. 2.2 seconds for computing and depicting 142 solutions in *Prob2.3*. In problems of Group 3, where binary and continuous variables increase simultaneously, the number of ESND solutions increases to about twice the respective number in Group 2. The computational effort is comparable to Group 1 but with a lower ratio of the number of optimizations per solution. These results clearly show that the computational effort grows very quickly with the increase of the number of integer variables but not with the number of continuous variables.

Groups 4 and 5 analyse the impact of changing the constraints. In Group 4, only the RHS of the constraints have been changed with respect to Group 3. They are, in general, larger than the previous ones, thus defining less restrictive constraints. As can be observed in Table 1, the number of ESND solutions is only slightly smaller than in Group 3, but the computational effort has been drastically reduced, being comparable to Group 2. Hence, we observe that constraints have a significant impact on the performance of the algorithm. The increase of the number of constraints in Group 5 considerably raises the number of solutions. The number of optimizations per solution increases steadily; each optimization requires more time.

Concerning the setting of ε, the computational effort to compute all ESND solutions was lower with $\varepsilon = 0.05$ than with $\varepsilon = 0.01$ in all three-objective problems.

Finally, the problems with two objective functions of Group 3_2 present a very small number of ESND points (about 10% of the respective number in the problems with three objective functions). The computational effort is also highly reduced. The average time per ESND solution in the larger problem is 0.008 seconds against 0.3 seconds in the corresponding three-objective problem (*Prob3.3*).

6. Conclusions

In this paper we have proposed a procedure to compute subsets of indifference regions in the weight space for MOMILP problems. We have also proposed algorithms to compute all extreme supported nondominated solutions (ESND) adjacent to a given solution or even all ESND solutions to problems with up to three objective functions. These algorithms are based on a graphical exploration of the weight space. A computational implementation has been developed to take the most of these procedures. Two examples have been shown to illustrate the proposed features. Computational experiments have also been presented.

Indifference regions give useful information to the decision maker allowing him/her to be indifferent to all weight combinations inside one region as those combinations give rise to the same nondominated solution. The graphical visualization of these regions supports the decision maker in the interactive exploration of the weight space, in particular for the exploration of neighbourhoods of solutions. Although the main aim is not to generate all solutions to the problem, an algorithm to compute all ESND solutions

is also available, which is still based on the graphical exploration of the weight space. It is suitable to characterize solutions in delimited nondominated areas or to be used as a final exploration phase in problems with a moderate number of extreme nondominated solutions.

As future work, we intend to study other approaches with the purpose of defining larger subsets of the indifference regions within the weighted-sum optimization step. We also aim to address the exploration of unsupported efficient solutions in the "neighbourhood" of supported efficient solutions.

Acknowledgements

This work has been partially supported by project EMSURE-Energy and Mobility for Sustainable Regions (CENTRO-07-0224-FEDER-002004) and by Fundação para a Ciência e a Tecnologia under project grant UID/MULTI/00308/2013.

References

Alves, M.J., & Clímaco, J. (2000). An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, 124(3), 478-494.

Alves, M.J., & Clímaco, J. (2001). Indifference sets of reference points in multiobjective integer linear programming. *Journal of Multi-Criteria Decision Analysis*, 10(4), 177-189.

Alves, M.J., Antunes, C.H., & Clímaco J. (2015). Interactive MOLP Explorer - a graphical-based computational tool for teaching and decision support in multi-objective linear programming models. *Computer Applications in Engineering Education* 23(2), 314-326.

Antunes, C.H., Alves, M.J., Silva, A.L., & Clímaco J. (1992). An integrated MOLP method base package - a guided tour of TOMMIX. *Computers and Operations Research*, 19(7), 609-625.

Barber, C. B., Dabkin, D. P. & Huhdanpa, H. (1996). The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4), 469-483.

Benson, H.P., & Sun, E. (2002). A weight decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. *European Journal of Operational Research*, 139(1), 26-41.

Bykat, A. (1978). Convex hull of a finite set of points in two dimensions. *Information Processing Letters*, 7(6), 296-298.

Clímaco, J., & Antunes, C.H. (1987). TRIMAP - an interactive tricriteria linear programming package. *Foundations of Control Engineering*, 12, 101-119.

Costa, J.P., & Clímaco, J. (1999). Relating reference points and weights in MOLP. *Journal of Multi-Criteria Decision Analysis*, 8(5), 281-290.

Eddy, W. F. (1977). A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software (TOMS)*, 3(4), 398-403.

Özpeynirci, O., & Köksalan, M. (2010). An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer problems. *Management Science*, 56(12), 2302-2315.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010). A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3), 371-386.

Yu, P.L., & Zeleny, M. (1976). Linear multiparametric programming by multicriteria simplex method. *Management Science*, 23(2), 159-170.

Appendix

Problem of the *Example2*, with 3 objective functions, 20 variables (10 binary and 10 continuous) and 10 constraints:

$$\begin{aligned} &\text{Max } z = Cx \\ &s.t.\ Ax \le b \\ &\quad x_j \in \{0,1\},\ j = 1,\ldots,10 \\ &\quad 0 \le x_j \le 1,\ j = 11,\ldots,20 \end{aligned}$$

with $C = \begin{bmatrix} 60 & 65 & 0 & 27 & 10 & 73 & 18 & 29 & 0 & 0 & 9 & 47 & 0 & 28 & 0 & 40 & 0 & 31 & 48 & 29 \\ 33 & 11 & 66 & 0 & 50 & 48 & 0 & 72 & 78 & 39 & 20 & 95 & 23 & 37 & 29 & 24 & 0 & 89 & 0 & 0 \\ 0 & 3 & 17 & 70 & 0 & 5 & 31 & 15 & 51 & 93 & 88 & 0 & 50 & 46 & 0 & 74 & 89 & 0 & 71 & 64 \end{bmatrix}$,

$$A=\begin{bmatrix} 19 & 25 & 6 & 0 & 37 & 42 & 0 & 0 & 43 & 24 & 14 & 38 & 12 & 0 & 0 & 49 & 16 & 27 & 27 & 39 \\ 0 & 25 & 30 & 0 & 0 & 28 & 38 & 5 & 48 & 19 & 0 & 35 & 15 & 30 & 4 & 26 & 23 & 26 & 24 & 38 \\ 29 & 44 & 3 & 3 & 14 & 20 & 29 & 41 & 33 & 30 & 31 & 2 & 0 & 0 & 44 & 41 & 0 & 35 & 0 & 46 \\ 29 & 37 & 19 & 31 & 2 & 28 & 10 & 45 & 0 & 3 & 24 & 49 & 0 & 38 & 2 & 18 & 0 & 48 & 22 & 0 \\ 3 & 21 & 0 & 35 & 22 & 49 & 26 & 44 & 50 & 3 & 13 & 21 & 24 & 47 & 35 & 6 & 43 & 29 & 13 & 50 \\ 27 & 23 & 38 & 2 & 25 & 0 & 0 & 27 & 2 & 11 & 47 & 49 & 0 & 0 & 11 & 6 & 40 & 43 & 0 & 28 \\ 42 & 40 & 44 & 30 & 0 & 43 & 20 & 5 & 0 & 25 & 20 & 0 & 44 & 16 & 24 & 15 & 0 & 0 & 0 & 0 \\ 33 & 44 & 46 & 15 & 24 & 8 & 9 & 31 & 20 & 40 & 9 & 44 & 33 & 26 & 14 & 38 & 41 & 44 & 7 & 4 \\ 1 & 40 & 12 & 24 & 5 & 45 & 0 & 5 & 20 & 21 & 4 & 0 & 17 & 34 & 0 & 42 & 37 & 26 & 38 & 17 \\ 33 & 0 & 24 & 45 & 0 & 40 & 0 & 43 & 42 & 7 & 0 & 32 & 3 & 0 & 46 & 0 & 26 & 0 & 40 & 44 \end{bmatrix} \quad b=\begin{bmatrix} 285 \\ 481 \\ 478 \\ 483 \\ 215 \\ 375 \\ 635 \\ 605 \\ 644 \\ 425 \end{bmatrix}.$$

Objective values of the 49 ESND solutions of the problem:

solution #	z_1	z_2	z_3	**solution #**	z_1	z_2	z_3
solution 1	417.308	384.462	352.231	solution 25	372.347	548.633	307.548
solution 2	415.231	418.846	424.923	solution 26	297.260	578.225	350.411
solution 3	226	589	388	solution 27	388.069	362.069	502
solution 4	263.463	579.276	422.276	solution 28	401.5	336	484
solution 5	291	579.875	347.250	solution 29	400	446.083	466.833
solution 6	172.191	173.574	648.957	solution 30	402	509.458	393.083
solution 7	230.420	205	645.720	solution 31	398.693	526.608	269.402
solution 8	264	539	477.628	solution 32	394.555	528.533	294.633
solution 9	269.362	546.085	467.809	solution 33	376.665	544.171	311.985
solution 10	279.362	575.002	421.975	solution 34	315.140	311	597.240
solution 11	294.512	578.963	350.463	solution 35	233.320	378	608.120
solution 12	413.724	412.724	452	solution 36	235.276	390.276	603.000
solution 13	409.745	507.234	378.723	solution 37	329.383	403.149	558.915
solution 14	413.106	442.319	442.532	solution 38	323.793	468.793	517.000
solution 15	283.240	216	621.840	solution 39	357.787	474.362	483.936
solution 16	280.383	219.149	622.915	solution 40	366	503.792	441.417
solution 17	286.952	234.095	619	solution 41	391.965	533.560	262.489
solution 18	262.383	303.149	619.915	solution 42	387	359	504.070
solution 19	198.5	283	635	solution 43	387.580	359	503.280
solution 20	262	467	551.256	solution 44	397.862	441.862	471.000
solution 21	327	475.125	510.750	solution 45	318.060	311	594.960
solution 22	329	538.5	437	solution 46	329.320	400	560.120
solution 23	348.532	534.596	426.660	solution 47	233.383	381.149	606.915
solution 24	351.766	546.131	379.163	solution 48	369	359	526.884
				solution 49	321.483	332.483	586