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.2022.02.047

A new exact method for linear bilevel problems with multiple objective functions at the lower level

Maria João Alves[1,3], Carlos Henggeler Antunes[2,3]

[1] CeBER and Faculty of Economics, University of Coimbra, Portugal
[2] Department of Electrical and Computer Engineering, University of Coimbra, Portugal
[3] INESC Coimbra, Portugal
mjalves@fe.uc.pt; ch@deec.uc.pt

Abstract

In this paper we consider linear bilevel programming problems with multiple objective functions at the lower level. We propose a general-purpose exact method to compute the optimistic optimal solution, which is based on the search of efficient extreme solutions of an associated multiobjective linear problem with many objective functions. We also explore a heuristic procedure relying on the same principles. Although this procedure cannot ensure the global optimal solution but just a local optimum, it has shown to be quite effective in problems where the global optimum is difficult to obtain within a reasonable timeframe. A computational study is presented to evaluate the performance of the exact method and the heuristic procedure, comparing them with an exact and an approximate method proposed by other authors, using randomly generated instances. Our approach reveals interesting results in problems with few upper-level variables.

Keywords: Multiple objective programming; Linear bilevel optimization; Semivectorial bilevel problem; Multiobjective simplex method.

1 Introduction

Bilevel programming is useful to model optimization problems with a hierarchical relation between two decision makers (the leader and the follower), who make decisions sequentially in a non-cooperative manner. The two decision makers control different sets of variables aiming to optimize their own objective functions. The leader commits to a strategy before the follower, who then optimizes his/her own objective function within the options restricted by the leader's decision. However, the follower's decision affects the leader's objective function value and even his/her feasible options, so the leader must anticipate the reaction of the follower. Sequential decision-making processes often appear in the management of decentralized organizations and policy making. For instance, in a road network design, the aim of the leader may be the minimization of investment and operational costs, but he/she has to incorporate in his/her decision the traffic pattern resulting from the travelers' decisions, who want to minimize travel time, gasoline consumption, among other objectives (follower's problem). Another common problem in the context of transportation policy is the toll-setting problem, where the upper level decision maker is an authority that wants to set tolls for a network of highways to maximize its revenues, while and the drivers (the lower-

level decision makers) want to minimize their travel time and cost. Many practical applications in the energy sector have also been reported in the literature including, for instance, in electricity retail markets involving demand response: bilevel programs allow to capture the sequence of decisions regarding the price announcement by the electricity retailer (the leader, who wants to maximize profit) and the consumers' reaction involving changes in the energy use to optimize cost and comfort (Alves and Antunes, 2018). As these examples illustrate, the players may have multiple objective functions. For the interested reader, we refer to (Sinha, Malo and Deb, 2018) and (Alves, Antunes and Costa, 2019) for the description and references of different application areas using bilevel multiobjective models.

Multiple objective functions add further theoretical, methodological and computational complexities to the (single-objective) bilevel problem (BP), which is an already difficult problem. The BP is strongly NP-hard even when all functions involved are linear (Dempe, 2002). Dealing with multiple objectives at the lower level is particularly challenging due to the existence of a set of lower-level efficient (Pareto optimal) solutions for each leader's decision. Bilevel problems with a single objective function at the upper level and multiple objective functions at the lower level (BPMOLL) have also been called *semivectorial* bilevel problems.

In this paper, we address the linear BPMOLL. A general-purpose exact method to solve this type of problems is proposed, which is based on the search of efficient extreme solutions of an associated multiobjective linear programming (MOLP) problem with many objective functions.

1.1 Related literature on bilevel problems with multiple objective functions at the lower level

The BPMOLL was firstly addressed by Bonnel (2006) and Bonnel and Morgan (2006), who provided first order necessary conditions to the problem in the former study and a penalty method in the latter one, considering weakly-efficient solutions at the lower level. Dempe and Mehlitz (2020) investigated the relationship between a (general) BPMOLL and its replacement by a scalar BP obtained by applying the weighted-sum scalarization to the multiobjective lower-level problem and interpreting the scalarization parameters (i.e., the weights) as new upper-level variables. Still considering weakly-efficient solutions, Ankhili and Mansouri (2009), Zheng and Wan (2011), Zheng, Chen and Cao (2014) and Ren and Wang (2016) proposed solution approaches based on penalty methods for the all linear BPMOLL, or at least the lower level is a MOLP problem.

Calvete and Galé (2011) also addressed bilevel problems with MOLP lower-level problems. They proved that the feasible region of the BPMOLL (the inducible region) is given by the union of faces of the polyhedron defined by all constraints. Assuming that the upper-level objective function is quasiconcave, they concluded that there is an extreme point of this polyhedron that is the optimal solution to the problem. Both an enumerative exact algorithm and a genetic-based algorithm were proposed. The exact method (for the linear BPMOLL) is an extension of the *k-th best* algorithm by Bialas and Karwan (1984) for the single-objective case. The method searches for extreme points of the constraint polyhedron until a feasible one is obtained, i.e., a solution that is efficient to the lower-level problem; since the algorithm computes an

ordered sequence of points, the first one that belongs to the inducible region solves the problem. Due to the well-known difficulties of exact methods in solving even medium-sized bilevel problems, a genetic algorithm was also proposed which explores extreme points by combining bases of the constraint polyhedron.

Lv and Wan (2014) reformulated the linear BPMOLL as a nonlinear bilevel program, where the weighted-sum scalarization is used to aggregate the lower-level objective functions and the weights are variables comprised in the upper-level variable set. Then, an optimal-value-function approach was proposed to deal with the problem.

All the aforementioned studies adopt the *optimistic* formulation of the BPMOLL, which assumes that the follower accepts any efficient solution of the lower-level problem. Therefore, solving the BPMOLL corresponds to optimizing the upper-level objective function over the inducible region, which is composed by the solutions that satisfy all the constraints and are efficient to the lower-level problem. The *pessimistic* formulation, which has been mainly discussed in the context of single-objective bilevel problems, assumes that the leader is risk-averse and prepares for the worst case. This means assuming that the follower chooses the worst solution to the leader among his/her efficient solution set (for each leader's decision) and the aim of the leader is to optimize the upper-level objective function over this set of 'worst' solutions. The optimal pessimistic solution to the BPMOLL is even more difficult to calculate than the optimistic one. Other type of solutions can also give useful information to the leader about the risk he/she takes from a particular decision. A discussion on the assumptions and implications of optimistic vs. pessimistic approaches in BPMOLL was firstly presented in (Alves, Antunes and Carrasqueira, 2015). An overview of different solution concepts and perspectives of development in BPMOLL and BP with multiple objective functions at both levels are provided in (Alves, Antunes and Costa, 2019) and (Alves, Antunes and Costa, 2021).

In the present work we consider the optimist formulation of the linear BPMOLL (LBPMOLL). Since the proposed approach relies on computing all efficient extreme solutions of an associated MOLP problem, a brief literature review of algorithms to generate the whole set of extreme efficient (or nondominated) points in general MOLP problems is presented in the next subsection.

1.2 Vector-maximum algorithms in MOLP

Although the development of algorithms to calculate all efficient extreme points of MOLP problems (vector-maximum algorithms) is longstanding, there are very few efficient software implementations reported in the literature. Evans and Steuer (1973) developed a multiobjective simplex method that finds the set of all efficient extreme solutions and the set of unbounded efficient edges for MOLP problems, which has a computer implementation called ADBASE. According to Schechter and Steuer (2005), ADBASE has been the dominant computer code for computing all efficient extreme points since its inception (in 1974) with more than 100 citations, and it has undergone several revisions over the years. Benson (1998) proposed an algorithm that works in the objective space, which computes all nondominated

extreme points of the MOLP problem. It employs an outer approximation technique, so only when the algorithm terminates the points in the approximation set are guaranteed to be nondominated. In (Benson, 1998), the algorithm was tested in random problems with up to 3 objective functions, 20 variables and 15 constraints. More recently, Rudloff, Ulus and Vanderbei (2017) proposed a parametric simplex algorithm than can be seen as a variant of the Evans-Steuer algorithm, but which does not aim to find the set of all efficient extreme solutions. In each iteration, only a subset of efficient nonbasic variables is chosen and, for each of these entering variables, only a single pivot is picked to determine the leaving variable. The proposed algorithm was implemented in Matlab and was compared with Benson's algorithm, using the Matlab code *bensolve* (Löhne and Weißing, 2017); it was also compared with the Evans-Steuer algorithm, for which the authors developed a Matlab implementation to use the same test environment for the three algorithms. In the computational experiments using randomly generated problems with 4 objective functions, *bensolve* was more efficient than the other algorithms for the degenerate problems (with 10/30 to 30/10 variables/constraints), but the Evans-Steuer algorithm was the best one for the non-degenerate problems (with 30/50 to 50/30 variables/constraints).

It should be remarked that methods aiming to generate all efficient extreme points are difficult to implement and the computational burden to compute all these solutions may be severe, since the number of efficient extreme points increases significantly with the problem size. Developing an effective computer implementation of a vector-maximum method is a demanding task, because it requires an implementation from scratch involving the design of bookkeeping and backtracking schemes with several theoretical and numerical concerns (namely for degenerate bases). These implementations can hardly make use of commercial solvers as modules, unless LP subproblems are solved separately.

1.3 Main contributions

In this paper, we develop a general-purpose exact method to compute the optimistic optimal solution to the LBPMOLL, which explores vertices of the constraint polyhedron that are efficient to the lower-level problem. We call this method *EEPSM – efficient extreme points search method*. This method is based on the property that the optimal solution to the LBPMOLL is an efficient solution of an associated single-level MOLP problem with many objective functions. This property will be stated and proved in this paper. The idea of vertex enumeration is shared with the work of Calvete and Galé (2011) but, while their algorithm works with infeasible solutions and stops when the first feasible solution is found, our approach works with feasible solutions only. Therefore, it can be interrupted at any moment, yielding a solution that may be non-optimal but which is surely feasible. Based on this principle, a local search heuristic procedure is also developed to deal with problems in which the complete vertex enumeration becomes impracticable. The heuristic ensures a local optimum solution, so we call it *LOH - local optimum heuristic*. The *LOH* can be further parameterized to define the extent of the neighborhood for the local search.

A computational study is presented to evaluate the performance of the *EEPSM* and the *LOH*, comparing them with another two algorithms (one exact and one approximate) using randomly generated instances.

The comparison will be made with the algorithms of Calvete and Galé (2011), in our opinion the most representative and comprehensive approaches for LBPMOLL which have been implemented computationally and tested in a large number of instances. The other approaches proposed in the literature for LBPMOLL do not report on computational experiments, only providing very small illustrative examples, which are not suitable for algorithm assessment. Most of these examples are used by the different authors (Ankhili and Mansouri, 2009), (Zheng and Wan, 2011), (Zheng, Chen and Cao, 2014), (Lv and Wan, 2014), (Ren and Wang, 2016).

Our main contribution in this work is a new exact method (*EEPSM*) for the LBPMOLL, a field with very few practical developments. This method is computationally interesting for problems with relatively few upper-level variables, since the number of objective functions of the associated single-level MOLP problem depends on the number of upper-level variables. This is the main limitation of the method. The development of the *EEPSM* required the development of an effective vector-maximum algorithm for MOLP problems. This is a further contribution of this work to the multiobjective optimization area.

Another contribution is the heuristic procedure *LOH* based on the *EEPSM*, which has revealed its effectiveness in problems where the global optimum is difficult to guarantee within a reasonable computational time. The *EEPSM* and the *LOH* are compared with the *k-th best* algorithm and the genetic algorithm proposed by Calvete and Galé (2011), respectively.

The rest of paper is organized as follows. Section 2 states the problem, notation and definitions. Section 3 is devoted to the new exact method for LBPMOLL presenting: the fundamentals and the steps of the *EEPSM*; a numerical example comparing the search path of *EEPSM* and the *k-th best* algorithm; the main characteristics (advantages and disadvantages) of these two approaches. Section 4 presents the computational experiments, the comparison of the algorithms and discusses the results. Section 5 presents the heuristic procedure. Section 6 is devoted to its computational experiment, making comparisons of the *LOH* with: i) the *EEPSM*; ii) different *LOH* parameterizations; iii) the genetic algorithm of Calvete and Galé (2011). Conclusions are drawn in Section 7.

2 Linear bilevel problems with multiple objectives at the lower level

In this work, we focus on the Linear Bilevel Problem with Multiple Objective Functions at the Lower Level (LBPMOLL), considering that upper-level constraints only include upper-level variables. Let $x \in \mathbb{R}^{n_1}$ be the vector of upper-level variables and $y \in \mathbb{R}^{n_2}$ be the vector of lower-level variables; $F(x, y)$ is the upper-level objective function and $f_1(x, y), \cdots, f_p(x, y)$ are the p objective functions at the lower level. Since x is a fixed vector whenever $f_1(x, y), \cdots, f_p(x, y)$ are optimized, these objective functions can be expressed in terms of y only. The optimistic formulation is considered, which assumes that the follower is indifferent to the efficient solutions to the lower-level problem for a given x, i.e., the efficient solution that most benefits the upper-level objective function is taken. This is why the maximization at the upper level is formulated with respect to x and y. The LBPMOLL can be stated as follows:

$$\max_{x,y} \; F(x,y) = c^1 x + d^1 y$$
s.t.
$$\begin{aligned} & A^1 x \leq b^1 \\ & x \geq 0 \\ & y \in \Psi_{ef}(x) \end{aligned} \tag{1}$$

where $\Psi_{ef}(x)$ denotes the set of all efficient solutions to the multiobjective lower-level problem (2) for a given x, which we will denote by MOLL(x):

$$\begin{aligned} & \max_{y} f_1(y) = d_1^2 y \\ & \dots \\ & \max_{y} f_p(y) = d_p^2 y \end{aligned} \tag{2}$$

s.t.
$$\begin{aligned} & A^2 x + B^2 y \leq b^2 \\ & y \geq 0 \end{aligned}$$

In (1) - (2), $c^1 \in \mathbb{R}^{n1}$, $d^1 \in \mathbb{R}^{n2}$, $d_j^2 \in \mathbb{R}^{n2}, j = 1, \cdots, p$, $A^1 \in \mathbb{R}^{m1 \times n1}$, $b^1 \in \mathbb{R}^{m1}$, $A^2 \in \mathbb{R}^{m2 \times n1}$, $B^2 \in \mathbb{R}^{m2 \times n2}$, $b^2 \in \mathbb{R}^{m2}$.

Let S denote the *constraint region* of the LBPMOLL and $S(x)$ is the feasible region of MOLL(x) for a given x: $S = \{(x,y) \in \mathbb{R}^{n1} \times \mathbb{R}^{n2} : A^1 x \leq b^1, A^1 x + B^2 y \leq b^2, x \geq 0, y \geq 0\}$,

$S(x) = \{y \in \mathbb{R}^{n2} : \; B^2 y \leq b^2 - A^2 x, y \geq 0\}$.

It is assumed that S is nonempty and compact and, for each decision taken by the leader, the follower has some room to respond, i.e. $S(x) \neq \emptyset$ for each feasible x, and $S(x)$ is bounded.

A solution $y \in S(x)$ is *efficient* (Pareto optimal) to the MOLL(x) if and only if there is no other $y' \in S(x)$ that *dominates* y, i.e., such that $f_j(y') \geq f_j(y)$ for all $j = 1, \cdots, p$ and $f_j(y') > f_j(y)$ for at least one $j = 1, \cdots, p$. A point $z \in \mathbb{R}^{\mathrm{p}}$ on the objective function space corresponding to an efficient solution y, $z = \big(f_1(y), f_2(y), \cdots, f_p(y)\big)$, is called a nondominated point.

The set of all efficient solutions to the MOLL(x) is denoted by $\Psi_{ef}(x) = \{y \in S(x) : y \text{ is efficient to MOLL}(x)\}$.

The feasible region of the LBPMOLL is called *inducible region* and is defined as $IR = \big\{(x,y) \in S : y \in \Psi_{ef}(x)\big\}$. Solving (1) means finding a solution that maximizes the upper-level objective function $F(x,y)$ over IR.

Some relevant properties of the LBPMOLL are:

- The IR of the LBPMOLL is, in general, non-convex, as in the single-objective linear BP.
- Since MOLL(x) is a multiobjective linear problem, $\Psi_{ef}(x)$ is the union of faces of $S(x)$ and it is connected (Ehrgott, 2005).
- As in the single-objective linear BP with no upper-level constraints involving lower-level variables (i.e., with no coupling constraints), the IR of (1) is connected. This property is proved

in the next section, by showing the equivalence of IR with the efficient set of an associated MOLP problem.

- The IR consists of the union of faces of the constraint region S (Calvete and Galé, 2011); in the case of the formulation (1) with no coupling constraints, the IR is a set of connected faces because IR is connected. However, these are not necessarily maximal faces, as it will be shown in the example below.
- There is an extreme point (vertex) of S that is an optimal solution to the LBPMOLL (Calvete and Galé, 2011).

Example 1

Consider the following illustrative example with one upper-level variable (x) and two lower-level variables (y_1, y_2):

$$max\ F(x,y) = x + 2y_1 - y_2$$

s.t.

$$2 \le x \le 5$$

$$max\ f_1(y) = y_1 + 2y_2$$

$$max\ f_2(y) = -y_1 + y_2$$

s.t.

$$y_1 \le 6$$
$$y_1 + y_2 \le 10 - x$$
$$y_2 \le x$$
$$5y_1 - 2y_2 \ge 0$$
$$y_1, y_2 \ge 0$$

In order to better illustrate the behavior of the bi-objective problem at the lower level, Figure 1 shows the feasible region and the efficient solutions of MOLL(x) for three specific values of x: x =2, x =3 and x =4.5. Consider, for instance, the graph for x =2: $S(2) = \{y_1 \le 6, y_1 + y_2 \le 8, y_2 \le 2, 5y_1 - 2y_2 \ge 0\ ,\ y_1, y_2 \ge 0\}$; the solution that optimizes individually f_1 is P1, where (y_1, y_2)=(6, 2) and the solution that optimizes individually f_2 is P2, where (y_1, y_2)=(0.8, 2); thus, the set of efficient solutions of MOLL(2) is the line segment from P1 to P2, which belongs to the boundary of the constraint $y_2 \le x$ of S. For MOLL(3) a similar situation occurs: the set of efficient solutions is a line segment on the same face of S. For MOLL(4.5), the constraint $y_2 \le x$ is redundant; P1≡P2 is the point that optimizes simultaneously f_1 and f_2, which is given by the intersection of $y_1 + y_2 = 10 - x$ (with x = 4.5) and $5y_1 - 2y_2 = 0$. Thus, only P1≡P2 is efficient for MOLL(4.5), where (y_1, y_2)=(1.571, 3.929).

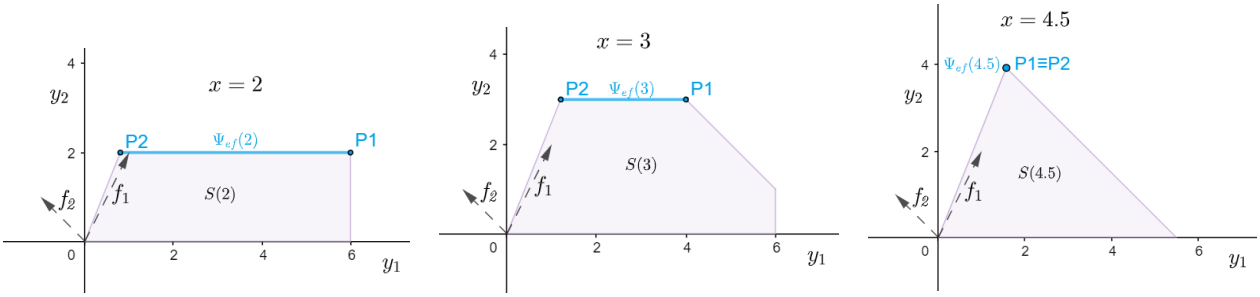

Figure 1 - MOLL(x) for =2, x =3 and x =4.5 of the Example 1.

Figure 2 shows the constraint region S in the (x, y_1, y_2) space and, in blue (darker) color, IR. Note that, in Figure 1, [P1, P2] for x =2 is the line segment [AB] in Figure 2, but P1 and P2 for x =3 and for x =4.5 do not correspond to vertices of S. As can be observed in Figure 2, the IR consists of the union of two faces of the constraint region S, but only one is maximal, [ABC], the other being the line segment [CD]. The vertices of IR have the following coordinates (x, y_1, y_2) and upper-level objective values: $A = (2, 6, 2)$, $F(A)$= 12; $B = (2,\ 0.8, 2), F(B)$=1.6; $C = (4.167,\ 1.667,\ 4.167)$, $F(C)$=3.333; $D = (5,\ 1.429,\ 3.571)$, $F(D)$=4.286. Thus, the optimal solution to the bilevel problem is the point *A* with $F(A)$= 12.

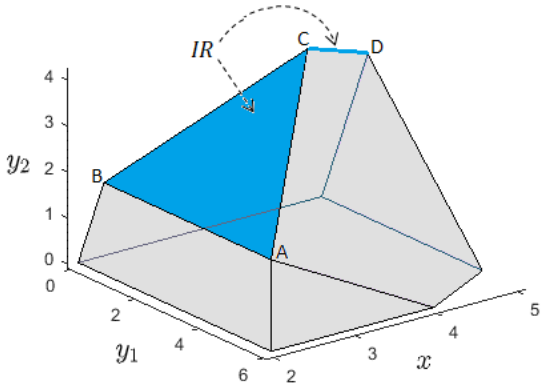

Figure 2 – The constraint region S (gray color) and the inducible region IR (blue color) of Example 1.

3 An exact algorithm based on the search for extreme points of *IR*

In this work, we propose an exact method (called *EEPSM*) to compute the optimal solution(s) to the LBPMOLL or, at least, a local optimum in the case of the heuristic procedure that will be presented in section 4. Both techniques use an associated MOLP problem with many objective functions, whose efficient solutions are feasible solutions to the bilevel problem (i.e., solutions belonging to *IR*). In the exact method, all extreme points (vertices) of the *IR* are computed, selecting from this set the one that presents the maximum $F(x, y)$. So, an algorithm for computing all efficient extreme points of a MOLP problem (following the general scheme of a vector-maximum algorithm described in (Steuer, 1986)) has been developed to apply to this MOLP problem.

This method will be compared with another exact method, the *k-th best* algorithm proposed by Calvete and Galé (2011), which goes through vertices by decreasing order of F until the first feasible vertex is found.

The *k-th best* algorithm (Calvete and Galé, 2011) for the LBPMOLL builds on the algorithm developed by Bialas and Karwan (1984) for the linear BP by searching the vertices of S and assessing whether they belong to IR. This test involves solving a linear programming problem using Benson's technique (Benson, 1978) to determine whether a solution (x', y') is efficient to the multiobjective problem MOLL(x'). The

algorithm starts by computing an optimal solution to the high-point relaxation problem, which consists of optimizing the upper-level objective function over S. If this solution belongs to the IR, then it is an optimal solution to the LBPMOLL. Otherwise, a set Q of its adjacent basic solutions in S is constructed. The algorithm proceeds by assessing the solution in Q with the best *F*-value. If this solution belongs to the IR, then the algorithm stops; else this solution is removed from Q and its adjacent basic solutions in S with worse *F*-value are added to Q. This set is progressively examined by selecting the best *F*-value solution until it belongs to IR. As the algorithm computes a set of non-improving *F*-value solutions, the first solution found belonging to IR is the optimal one, since a path between the vertices of S exists. This search strategy implies that the performance of the algorithm heavily depends on whether the first solution computed is close or far from the optimal solution to the LBPMOLL.

3.1 Fundamentals of the EEPSM

Consider the following MOLP problem with $(p + n_1 + 1)$ objective functions, which is associated with the feasible region of the LBPMOLL:

$$\begin{array}{lll} max & f_j(y) = d_j^2 y & j = 1, \cdots, p \\ max & x_i & i = 1, \cdots, n_1 \\ max & \sum_{i=1}^{n_1}(-x_i) & \\ \text{s.t.} & & \end{array} \qquad (3)$$

$$\left.\begin{array}{c} A^1 x \leq b^1 \\ A^2 x + B^2 y \leq b^2 \\ x, y \geq 0 \end{array}\right\} (x, y) \in S$$

The following proposition is an extension for the LBPMOLL of the Fülop's theorem (Fülöp, 1993) and the theorem presented in (Alves, Dempe and Júdice, 2012), both for the BP with only one objective function at the lower level.

Proposition 1: A solution (x', y') belongs to the IR of the LBPMOLL (1) if and only if (x', y') is an efficient solution to the MOLP problem (3).

Proof.

Let E denote the set of all efficient solutions to the MOLP problem (3).

(a) Let us first prove that $(x', y') \in IR \Rightarrow (x', y') \in E$.

By definition of IR, $(x', y') \in IR \Leftrightarrow (x', y') \in S \wedge y' \in \Psi_{ef}(x')$, which means that y' is efficient to the lower-level multiobjective problem (2) with $x = x'$, i.e., MOLL(x').

Let us suppose that $(x', y') \notin E$. Therefore, there is another $(x", y") \in E$ that dominates (x', y') in (3), i.e., the following conditions hold, with at least one strict inequality among the $p + n_1 + 1$ inequalities:

$$\begin{cases} d_j^2 y" \geq d_j^2 y' & j = 1, \cdots, p \qquad (a.1) \\ x"_i \geq x'_i & i = 1, \cdots, n_1 \qquad (a.2) \\ \sum_{i=1}^{n_1}(-x"_i) \geq \sum_{i=1}^{n_1}(-x'_i) & \qquad (a.3) \end{cases}$$

If conditions (a.2) are satisfied, then

$$\sum_{l=1,l\neq i}^{n_1} x"_l \geq \sum_{l=1,l\neq i}^{n_1} x'_l \quad , i = 1,\cdots,n_1 \qquad (a.4)$$

$(a.4)$ together with $(a.3)$ leads to

$$\sum_{l=1}^{n_1}(-x"_l) + \sum_{l=1,l\neq i}^{n_1} x"_l \geq \sum_{l=1}^{n_1}(-x'_l) + \sum_{l=1,l\neq i}^{n_1} x'_l \ , \quad i = 1,\cdots,n_1$$

$$\Leftrightarrow \ -x"_i + \sum_{l=1,l\neq i}^{n_1}(x"_l - x"_l) \geq -x'_i + \sum_{l=1,l\neq i}^{n_1}(x'_l - x'_l), \quad i = 1,\cdots,n_1$$

$$\Leftrightarrow \ -x"_i \geq -x'_i \ , \quad i = 1,\cdots,n_1$$

$$\Leftrightarrow \ x"_i \leq x'_i \ , \ i = 1,\cdots,n_1 \qquad (a.5)$$

By $(a.2)$ and $(a.5)$, $x"_i = x'_i$, $i = 1,\cdots,n_1$.

Therefore, the strict inequality among (a.1), (a.2) and (a.3) must be in (a.1), i.e.: $d_j^2 y" > d_j^2 y'$ for some $j \in \{1,\cdots,p\}$. Hence, since $y" \in S(x')$ because $x" = x'$, then $y"$ dominates y' in MOLL(x'), i.e. $y' \notin \Psi_{ef}(x')$, which contradicts the hypothesis that $(x', y') \in IR$. Therefore, $(x', y') \in E$.

(b) Let us now prove that $(x', y') \in E \ \Rightarrow (x', y') \in IR$.

Let us suppose that $(x', y') \notin IR$. Since $(x', y') \in S$, then $(x', y') \notin IR$ only if $y' \notin \Psi_{ef}(x')$, i.e., there is another $y" \in S(x')$ such that $d_j^2 y" \geq d_j^2 y'$ for all $j = 1,\cdots,p$ and $d_j^2 y" > d_j^2 y'$ for at least one j.

Hence, $(x', y")$ dominates (x', y') in the MOLP problem (3), i.e., $(x', y') \notin E$, which contradicts the hypothesis. Therefore, $(x', y') \in IR$. □

Based on Proposition 1, the following Corollary can be stated for the LBPMOLL.

Corollary 1 – An optimal solution to the LBPMOLL can always be found in an efficient extreme point of (3).

This corollary is valid because $F(x, y)$ is a linear function, thus an optimistic optimal solution can always be found in a vertex of IR (Calvete and Galé, 2011), i.e., a vertex of E.

Before describing the algorithm for the LBPMOLL, let us present some definitions and foundations of general MOLP problems (Steuer, 1986), (Ehrgott, 2005), (Antunes, Alves and Clímaco, 2016) that can be applied to (3). For the sake of simplicity, let us consider that $\boldsymbol{x}$ denotes the vector of all variables of the problem and $C\boldsymbol{x}$ are the objective functions of the MOLP problem, where C is the matrix of the objective function coefficients with ϱ rows, as many as the number of objective functions ($\varrho = p + n_1 + 1$ in (3)). Like in (3), S denotes the feasible region of the MOLP problem.

Let $\lambda \in \mathbb{R}^{\varrho}_{>0}$. The weighted-sum linear program (LP) associated with λ is: $\max\{\lambda C\boldsymbol{x} \mid \boldsymbol{x} \in S\}$.

Let $\mathcal{B}$ be a basis of S.

Let W denote the *reduced cost* matrix for the MOLP problem associated with $\mathcal{B}$: $W = \mathrm{C}_{\aleph} - \mathrm{C}_{\mathcal{B}}\overline{\mathcal{B}}^{-1}\overline{\aleph}$, where $\mathrm{C}_{\mathcal{B}}$ and $\mathrm{C}_{\aleph}$ are the basic and nonbasic columns of C, respectively, and $\overline{\mathcal{B}}$, $\overline{\aleph}$ are the corresponding columns of the constraint matrix of S.

- $\mathcal{B}$ is an *efficient* basis of the MOLP problem if and only if $\mathcal{B}$ is an optimal basis of the weighted-sum LP for some $\lambda \in \mathbb{R}^{\varrho}_{>0}$.
- An efficient basis $\mathcal{B}$ is always associated with an efficient extreme point of the MOLP problem. However, an efficient extreme point may be associated with several efficient bases (in the case of degeneracy).
- Two bases $\mathcal{B}$ and $\mathcal{B}'$ are called *adjacent* if one can be obtained from the other by a single pivot step.
- Let $\mathcal{B}$ be an efficient basis. Then, $\boldsymbol{x}_i$ is an *efficient nonbasic variable* with respect to (w.r.t.) $\mathcal{B}$ if there is a $\lambda \in \mathbb{R}^{\varrho}_{>0}$ such that $\lambda W \leq 0$ and $\lambda W_{\cdot i} = 0$, where $W_{\cdot i}$ is the column of W corresponding to $\boldsymbol{x}_i$.
- Let $\mathcal{B}$ be an efficient basis and $\boldsymbol{x}_i$ be an efficient nonbasic variable. Then, any feasible pivot operation from $\mathcal{B}$ (including any with negative pivot element whose associated basic variable is degenerate) is an efficient pivot operation leading to an adjacent efficient basis $\mathcal{B}'$.
- Let $\mathcal{B}$ be any efficient basis and $\overline{\boldsymbol{x}}$ be an efficient extreme point. Then, starting from $\mathcal{B}$, it is always possible to reach $\overline{\boldsymbol{x}}$ by performing only efficient pivots (Schechter and Steuer, 2005).

Therefore, if all efficient extreme points of (3) – vertices of E – are explored and the one with the best $F(x, y)$ is selected, then the optimal solution to the LBPMOLL is found. If there are more than one optimal solution, all of them can be known. The algorithm may start at any efficient basic solution of (3), as there is always a path of adjacent efficient bases between any two vertices of E.

3.2 The EEPSM algorithm

The *efficient extreme points search method – EEPSM –* to compute the optimal solution(s) to the LBPMOLL relies on Proposition 1. We have developed a vector-maximum algorithm to compute all efficient extreme points of the MOLP problem (3).

In order to assure that an optimal solution to the LBPMOLL is obtained, the algorithm must compute all efficient extreme points of (3). So, an algorithm intended to compute only a subset, as the parametric simplex algorithm of Rudloff, Ulus and Vanderbei (2017), does not suit this purpose. Moreover, searching for all nondominated extreme points in the objective space, as Benson's method (Benson, 1998), may not be sufficient to compute an optimal solution to the LBPMOLL. Each extreme point $\boldsymbol{x} = (x, y)$ can be mapped by the objective functions of the MOLP problem either into an extreme or a non-extreme nondominated point of the image set (Dauer and Liu, 1990). In addition, for a given extreme nondominated point there may exist several efficient solutions (in the decision variable space) and, although they are indifferent for the follower, they may have different values for the leader's objective function $F(x, y)$.

The algorithm we have developed to solve problem (3) is a variant of the Evans-Steuer algorithm, which searches for all efficient extreme points of bounded MOLP problems. The general scheme is similar to the one described in (Steuer, 1986) for a vector-maximum algorithm. All efficient bases are scanned including the degenerate ones. Having the main concern of being computationally efficient, the procedure employed to check the efficiency of nonbasic variables does not require solving an LP problem for each nonbasic variable. Although the description of the algorithm presented below refers to problem (3), this vector-maximum algorithm has a generic application and can be used for any bounded MOLP problem. In that case, the algorithm returns the set of all efficient extreme points rather than only the best solutions according to $F(x, y)$.

EEPSM Algorithm

Notation: $\mathcal{E}$ – set of efficient extreme points of (3); initially $\mathcal{E} = \emptyset$.
$\mathcal{L}_{\mathcal{B}}$ – list of efficient bases; initially $\mathcal{L}_{\mathcal{B}} = \emptyset$.
k – index of the basis in $\mathcal{L}_{\mathcal{B}}$ under analysis.
XY^* – set of incumbent solutions (x^*, y^*) to the LBPMOLL, optimal at the end of the algorithm.
F^*– value of the upper-level objective function in any solution $(x^*, y^*) \in XY^*$.

Step 1. Solve a weighted-sum LP of (3) with $\lambda \in \mathbb{R}^{\varrho}_{>0}$ to obtain a first basic efficient solution of (3).
Let $\mathcal{B}^1$ be the basis obtained and (x^1, y^1) be the efficient solution of (3) associated with $\mathcal{B}^1$.
Insert (x^1, y^1) into $\mathcal{E}$ and insert $\mathcal{B}^1$ into $\mathcal{L}_{\mathcal{B}}$ (by storing an encoding of the indices of the basic variables).
Initialize the set of incumbent solutions: $XY^* = \{(x^1, y^1)\}$; $F^* = F(x^1, y^1)$.
$k \leftarrow 1$

Step 2. For all nonbasic variable $\boldsymbol{x}_i$ (which can be either an x or y variable) w.r.t. $\mathcal{B}^k$, check whether $\boldsymbol{x}_i$ is an efficient nonbasic variable. If so, determine all feasible pivots (more than one only if the basis is degenerate) and each corresponding basis $\mathcal{B}$ (characterized by its basic variables) resulting from entering $\boldsymbol{x}_i$ into the basis. If $\mathcal{B} \notin \mathcal{L}_{\mathcal{B}}$, then $\mathcal{L}_{\mathcal{B}} \leftarrow \mathcal{L}_{\mathcal{B}} \cup \{\mathcal{B}\}$.

Step 3. $k \leftarrow k + 1$
If $|\mathcal{L}_{\mathcal{B}}| < k$, then all efficient bases have been analyzed – go to Step 4.
Otherwise, choose the k^{th} basis in $\mathcal{L}_{\mathcal{B}}$: $\mathcal{B}^k$.
Compute (x^k, y^k) associated with the basis $\mathcal{B}^k$. If $(x^k, y^k) \notin \mathcal{E}$, then $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x^k, y^k)\}$.
Update the set of incumbent solutions to the LBPMOLL:

If $F(x^k, y^k) \geq F^*$ then
  If $F(x^k, y^k) > F^*$ then
    $XY^* = \{(x^k, y^k)\}$; $F^* = F(x^k, y^k)$.
  Else
    $XY^* = XY^* \cup \{(x^k, y^k)\}$.

Go to Step 2.

Step 4. Return XY^* with all the optimal solutions found for the LBPMOLL. F^* is the optimal value of the upper-level objective function.

In Step 1, any $\lambda \in \mathbb{R}^{\varrho}_{>0}$ can be used. In our implementation, we have considered a central vector with equal weights for all the $\varrho = p + n_1 + 1$ objective functions, i.e., $\lambda = (\frac{1}{\varrho}, \frac{1}{\varrho}, \cdots, \frac{1}{\varrho})$.

In Step 2, we need a method to check whether a nonbasic variable $\boldsymbol{x}_i$ is efficient w.r.t. the basis under analysis $\mathcal{B}^k$. There are several tests for this purpose, almost all requiring to solve an LP subproblem for each $\boldsymbol{x}_i$. The Zionts-Wallenius routine (Zionts and Wallenius, 1980) enables to determine the efficiency status of all nonbasic variables in one extended run instead of solving one individual subproblem for each $\boldsymbol{x}_i$. However, according to our practical experience, this routine has difficulties in determining that status for all variables in some problems, particularly problems with many objective functions. Therefore, we have considered the following procedure:

Let W be the reduced cost matrix (with ϱ rows) associated with $\mathcal{B}^k$.

(i) If, for a given $\boldsymbol{x}_i$, all elements of the column $W_{\cdot i}$ are equal to 0, then $\boldsymbol{x}_i$ is efficient (in this case, the nonbasic variable will lead to an efficient solution corresponding to the same nondominated point as the current one).

(ii) Exclude from W the columns of $\boldsymbol{x}_i$ that received the status of *efficient* in *(i)* and apply the Zionts-Wallenius routine (as described in (Steuer, 1986)), which employs a pivoting iterative scheme to classify the variables as *efficient* or *non-efficient* from an initial status of *unknown*. If the routine does not change the status of some variable(s) during a predefined number of iterations, then we force it to stop.

(iii) In general, all nonbasic variables are classified in *(ii)* but, if there is still any nonbasic $\boldsymbol{x}_i$ whose status remain *unknown*, then the Isermann's test (Isermann, 1977) is applied to $\boldsymbol{x}_i$ to determine its *efficient*/*non-efficient* status.

The algorithm only explores different efficient bases. However, the verification whether $(x^k, y^k) \notin \mathcal{E}$ is necessary in Step 3 because (x^k, y^k) may already exist in $\mathcal{E}$ if $\mathcal{B}^k$ is a degenerate basis, and only different efficient solutions are saved in $\mathcal{E}$.

Step 3 guarantees that all alternative best solutions are retained, so that the algorithm returns, in Step 4, all alternative extreme optimal solutions to the LBPMOLL.

The algorithm is guaranteed to finish because the number of bases S is finite and, in Step 2, only different bases ($\mathcal{B}$) enter into the list of efficient bases for further analysis (If $\mathcal{B} \notin \mathcal{L}_{\mathcal{B}}$, then $\mathcal{L}_{\mathcal{B}} \leftarrow \mathcal{L}_{\mathcal{B}} \cup \{\mathcal{B}\}$).

In the computational implementation of this algorithm, the bounded-variable simplex method was used. So, the basis encoding includes not only the indexes of the basic variables but also a record of the nonbasic

variables at the upper bound. Also, the Zionts-Wallenius routine in *(ii)* and the additional test in *(iii*) to check the efficiency status of nonbasic variables were adapted to deal with both variables at the lower bound (zero) and at the respective upper bounds. The algorithm was implemented in Delphi® (by Embarcadero), using the free solver *lpsolve* to obtain the first efficient solution in Step 1. It also takes advantage of the *lpsolve* subroutines to obtain the necessary data for determining the feasible pivots and the W matrix in Step 2, and to jump directly to a given basis in Step 3, calculating the respective solution.

To have a first evaluation of the vector-maximum algorithm's performance, we have tested it to calculate all efficient extreme solutions in general MOLP problems. The results of these experiments are presented in the Appendix, revealing that the number of efficient extreme points grows very fast with the number of objective functions. The resolution of (3) is, therefore, challenging.

All computational times reported throughout this paper refer to a computer Intel Core i7-7700 CPU 3.6 GHz, 64 GB RAM.

3.3 Numerical Example

Let us illustrate the use of the *EEPSM* in small numerical LBPMOLL example. In order to compare the search path with another exact method, we will start by showing the application of the *k-th best* algorithm (Calvete and Galé, 2011) in the same problem. As referred to above, this method searches for extreme solutions of S by decreasing order of $F(x,y)$ and, for each one, determines whether that extreme solution is efficient to the lower-level problem; the first efficient solution found is optimal to the bilevel problem. The Benson's test is employed to check the efficiency of a given solution (x',y'). Considering maximizing objective functions (as stated above in the definition of the LBPMOLL), this test consists of maximizing $\sum_{j=1}^{p} z_j$ subject to $\{y \in S(x'): d_j^2 y - z_j = d_j^2 y', j = 1,\cdots,p; \forall z_j \geq 0\}$; if the optimal objective value is equal to zero, then the solution is efficient.

Let us consider the problem of Example 1. This problem has the particularity of having several degenerate bases. The set S is defined by the following constraints (the slack/surplus of each constraint is in brackets):

$$
\begin{array}{ll}
y_1 \leq 6 & (s_1)\\
x + y_1 + y_2 \leq 10 & (s_2)\\
-x+y_2 \leq 0 & (s_3)\\
5y_1 - 2y_2 \geq 0 & (s_4)\\
x \geq 2 & (s_5)\\
x \leq 5 & (s_6)\\
y_1, y_2 \geq 0 &
\end{array}
$$

Let $\boldsymbol{x}$ denote (x, y_1, y_2), $f = (f_1, f_2)$ and nbv is the set of nonbasic variables for a given basis.

- ***k-th best algorithm***
  - ① Optimize $F(\boldsymbol{x}) = x + 2y_1 - y_2$ over S, which leads to point **E** (see Figure 3):

    $\boldsymbol{x} = (4, 6, 0)$, $f = (6, -6)$, $F = 16$.

Check the efficiency of **E** w.r.t. the lower-level problem MOLL($x = 4$) by solving the Benson's test problem → **E** is non-efficient.

Generate all bases adjacent to **E** and respective solutions (arrows 1 in Figure 3):

- point **G** with $\boldsymbol{x} = (5, 5, 0), f = (5, -5),\ F = 15$
- point **H** with $\boldsymbol{x} = (2, 6, 0), f = (6, -6), F = 14$
- point **A** (1st basis) with $\boldsymbol{x} = (2, 6, 2), f = (10, -4), F = 12,\ \ nbv = \{s_1, s_2, s_5\}$
- point **A** (2nd basis), $nbv = \{s_1, s_2, s_3\}$

② Choose the solution with best $F(\boldsymbol{x})$ among {**G**, **H**, **A**} → point **G** with $F = 15$.

Check the efficiency of **G** w.r.t. the lower-level problem MOLL($x = 5$) → **G** is non-efficient.

Generate all bases adjacent to **G** and respective solutions with $F(\boldsymbol{x}) \leq 15$ (arrows 2. In Figure 3):

- point **J** (1st basis) with $\boldsymbol{x} = (5, 0, 0), f = (0, 0), F = 5,\ \ nbv = \{y_1, y_2, s_6\}$
- point **J** (2nd basis), $nbv = \{y_2, s_4, s_6\}$
- point **D** with $\boldsymbol{x} = (5, 1.429, 3.571), f = (8.571, 2.143), F = 4.286$

③ Choose the solution with best $F(\boldsymbol{x})$ among {**H**, **A, J, D**} → point **H** with $F = 14$.

Check the efficiency of **H** w.r.t the lower-level problem MOLL($x = 2$) → **H** is non-efficient.

Generate all bases adjacent to **H** and respective solutions with $F(\boldsymbol{x}) \leq 14$ which are different from the already known bases (arrows 3. in Figure 3):

- point **I** (1st basis) with $\boldsymbol{x} = (2, 0, 0), f = (0, 0), F = 2,\ \ nbv = \{y_1, y_2, s_5\}$
- point **I** (2nd basis), $nbv = \{y_2, s_4, s_5\}$
- point **A** (3rd basis), $nbv = \{s_1, s_3, s_5\}$

④ Choose the solution with best $F(\boldsymbol{x})$ among {**A, J, D, I**} → point **A** with $F = 12$.

Check the efficiency of **A** w.r.t. the lower-level problem MOLL($x = 2$) → **A** is efficient.

Thus, **A** is an optimal solution to the LBPMOLL.

The *k-th best* algorithm searched for a total of 11 bases. This process is illustrated in Figure 3.

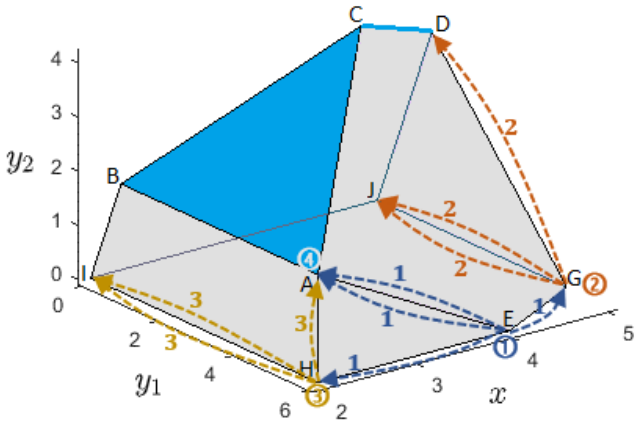

Figure 3 – Search path of the *k-th best* algorithm for the problem of Example 1.

- ***EEPSM***

  The MOLP problem (3) for this example has $\varrho = p + n_1 + 1 = 4$ objective functions to be optimized over S. All the efficient extreme solutions to this MOLP problem are computed. Although we consider equal weights for all the objective functions to begin the algorithm, the starting point is indifferent, as well as the order the algorithm follows to search for efficient extreme points, since all efficient extreme points must be visited.

  - Solve a weighted-sum LP of (3) with $\lambda = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, which leads to point **C**: $\boldsymbol{x} = (4.1667, 1.667, 4.1667)$, F =3.333. The set of efficient extreme points is $\mathcal{E}$ ={**C**}
  - Determine the efficient bases of (3) resulting from entering into the basis each of the efficient nonbasic variables w.r.t. the basis of **C**, say $\mathcal{B}^0 \rightarrow$ 4 bases are generated, $\{\mathcal{B}^1, \mathcal{B}^2, \mathcal{B}^3, \mathcal{B}^4\}$ (see Figure 4).
  - Compute the solution corresponding to $\mathcal{B}^1$:
    - point **B** with $\boldsymbol{x} = (2,\ 0.8, 2)$, $f = (4.8, 1.2)$, F =1.6
      $\mathcal{E}$ ={**C, B**}
      Determine new efficient bases adjacent to $\mathcal{B}^1 \rightarrow$ 1 new basis is generated, $\mathcal{B}^5$, which is added to the list of bases to be analyzed.
  - Compute the solution corresponding to $\mathcal{B}^2$:
    - point **D** with $\boldsymbol{x} = (5, 1.429, 3.571)$, $f = (8.571, 2.143)$, F =4.286
      $\mathcal{E}$ ={**C, B, D**}
      There are no new efficient bases adjacent to $\mathcal{B}^2$
  - Compute the solution corresponding to $\mathcal{B}^3$:
    - point **A** (1st basis found) with $\boldsymbol{x} = (2, 6, 2)$, $f = (10, -4)$, F =12, $nbv = \{s_1, s_2, s_3\}$
      $\mathcal{E}$ ={**C, B, D, A**}
      There are no new efficient bases adjacent to $\mathcal{B}^3$
  - Compute the solution corresponding to $\mathcal{B}^4$:
    - point **A** (2nd basis) , $nbv = \{s_2, s_3, s_5\}$
      There are no new efficient bases adjacent to $\mathcal{B}^4$
  - Compute the solution corresponding to $\mathcal{B}^5$:
    - point **A** (3rd basis), $nbv = \{s_1, s_3, s_5\}$
      There are no new efficient bases adjacent to $\mathcal{B}^5$
  - There are no more bases to analyze, so the algorithm finishes with $\mathcal{E}$ ={**C, B, D, A**} the set of all feasible extreme points of the LBPMOLL. The optimal solution is the point in $\mathcal{E}$ with best *F*, i.e., point **A** with F =12.

  The *EEPSM* searched for a total of 6 bases. This process is illustrated in Figure 4.

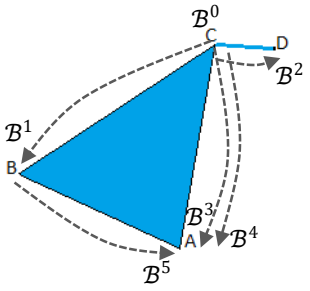

Figure 4 – Search path of the *EEPSM* for the problem of Example 1.

3.4 Main characteristics of the methods - similarities and differences

A first critical assessment of the methods – the *EEPSM* and the *k-th best* algorithm – can be made based on the main characteristics of these methods:

- If the LBPMOLL is *trivial*, then the *k-th best* algorithm finds its optimal solution in the first iteration. A bilevel problem is trivial if its optimal solution is given by solving the so-called high-point relaxation problem (i.e., the optimization of the upper-level objective function over the set of all constraints of the problem, the upper and the lower-level constraints). Therefore, the *k-th best* algorithm has advantages over the *EEPSM* if the optimal solution of the LBPMOLL is close to the solution that optimizes F in S.
- The *k-th best* algorithm can be used for problems with upper-level constraints involving lower-level variables (i.e., coupling constraints). The *EEPSM* requires that upper-level constraints include only upper-level variables in order to ensure the validity of Proposition 1.
- An advantage of the *EEPSM* is that it only searches for feasible solutions of the LBPMOLL. Therefore, if some computational budget is imposed and the method cannot finish within that time limit, the optimal solution cannot be assured but the method finishes with a feasible solution of the LBPMOLL; this does not happen with the *k-th best* algorithm, which does not yield a feasible solution until it ends.
- The main disadvantage of the *EEPSM* is that it will only be able to deal with bilevel problems with a few upper-level variables. The number of objective functions of the MOLP problem (3) depends on the number of upper-level variables and the computational burden of computing all efficient extreme solutions of (3) highly increases with the number of objective functions (as noted for general MOLP problems in Table A.1 of the Appendix).
- If the LBPMOLL has alternative optimal solutions, the *EEPSM* finds them all.

4 Computational results of the exact algorithms

The *EEPSM* was implemented in Delphi and the experiment was performed on a PC Intel Core i7-7700 CPU 3.6 GHz, 64 GB RAM under Windows 10. We also implemented the *k-th best* algorithm in Delphi and tested it using the same computer.

The problems were randomly generated using the rules in (Calvete and Galé, 2011) with two objective functions at the lower level, and the following numbers n_1- n_2 - m of upper-level decision variables (n_1), lower-level decision variables (n_2) and constraints (m): 5-10-10, 5-15-15, 5-10-20, 5-20-10, 5-20-20, 5-30-30, 5-40-40, 5-50-50, 10-20-20, 10-50-50. We consider problems with a few upper-level variables due to the nature of the *EEPSM*. All the constraints are of the type '$\leq$' and placed at the lower level ($m = m_2$); thus, the total number of variables of the problem (including slack variables) is $n_1 + n_2 + m$. For every problem size, a number of instances was generated in order to obtain five non-trivial problems in each category (i.e., problems whose optimal solution is given by the relaxed problem were discarded). Only in the categories 5-10-10 and 5-10-20 trivial problems were generated: in the first category, 11 problems were generated to obtain 5 non-trivial; in the category 5-10-20, only one trivial problem was generated. A total number of 50 problems was considered in this experiment. The data of this test set can be found at https://data.mendeley.com/datasets/6prkd8w9sm/1, an open-source online data repository hosted at Mendeley Data.

It should be noticed that we have also considered other instances presented in the literature that have been used to test/illustrate other algorithms for LBPMOLL, namely all the problems in (Ankhili and Mansouri, 2009), (Zheng and Wan, 2011), (Zheng, Chen and Cao, 2014), (Lv and Wan, 2014) and (Ren and Wang, 2016). However, these problems are very small and have been mainly used for illustrative purposes, thus not posing real challenges to the algorithms. Most of them are trivial, in which the optimization of the upper-level objective function over S leads to the optimal solution to LBPMOLL; these LP relaxations have at most 15 variables and 16 constraints. Only 3 non-trivial problems were found in this set – 1) the first example in (Ankhili and Mansouri, 2009), also presented in the other papers: both *EEPSM* and the *k-th best* algorithm searched for 5 basic solutions to reach the optimal solution; 2) the second example in (Zheng, Chen and Cao, 2014), also presented as illustrative example in (Calvete and Galé, 2011), which has two alternative optimal solutions: both *EEPSM* and the *k-th best* algorithm searched for 6 basic solutions; the *EEPSM* yielded the two optimal solutions; 3) the second example in (Lv and Wan, 2014): *EEPSM* searched for 4 basic solutions and the *k-th best* algorithm searched for 24 basic solutions to reach the optimal solution.

In the experiment using our test set of 50 problems, we imposed a computational time limit for each problem, which is equal in both algorithms. Since the computational effort increases with the size of the problem, in particular with n_1, the following limits were considered: 300 seconds for problems from 5-10-10 to 5-20-20, 600 s for problems 5-30-30 to 5-50-50, 900 s for 10-20-20, and 1800 s for 10-50-50. In the problems up to 5-20-20, the optimal solution was reached within the 300 s time limit in all cases except in one problem by the *k-th best* algorithm (problem 5-20-20-*b* in Table 1). The algorithm still does not finish even if the time limit is increased to 600 seconds (i.e., the same limit as the one given to the categories of higher dimensions).

Table 1 shows the computational results for both algorithms. For each problem, the optimal value of *F* is presented if any of the algorithms or both were able to achieve the optimal solution within the pre-established time limit. For each algorithm, the number of bases explored (#bases) and the computational time (in seconds) are presented; if the algorithm has finished, thus yielding the optimal solution, the mention 'Opt' is shown in the *best F* column; otherwise, the best value of *F* for feasible solutions obtained during the search is presented. This is only possible for the *EEPSM*, since the *k-th best* algorithm does not yield any feasible solution to the bilevel problem until the end of the algorithm. In this case, the best upper bound for *F* found by the *k-th best* algorithm (i.e., the *F* value in the last solution explored by the method) is presented in the UB column. The results of the algorithm with best performance in each problem are highlighted in bold.

Table 1 – Results of the *EEPSM* and the *k-th best* algorithm

		EEPSM			***k-th best* algorithm**			
Problem (n_1-n_2-m)	optimal *F*	#bases	time (s)	best *F* (feasible LB)	#bases	time (s)	best *F*	UB
5-10-10								
a	616.76	**282**	**0.26**	Opt	340	0.36	Opt	
b	1016.69	335	0.33	Opt	**29**	**0.02**	Opt	
c	366.19	**143**	**0.12**	Opt	269	0.23	Opt	
d	488.91	399	**0.39**	Opt	**389**	0.40	Opt	
e	2412.76	644	0.66	Opt	**16**	**0.01**	Opt	
5-15-15								
a	1737.74	2666	3.88	Opt	**2250**	**2.07**	Opt	
b	2205.02	3304	4.71	Opt	**2470**	**2.28**	Opt	
c	1042.43	**1901**	**2.51**	Opt	4387	5.29	Opt	
d	2791.62	818	0.93	Opt	**126**	**0.10**	Opt	
e	2130.60	528	0.66	Opt	**477**	**0.44**	Opt	
5-10-20								
a	664.18	**1499**	**1.83**	Opt	11764	19.28	Opt	
b	2440.45	1678	1.82	Opt	**67**	**0.05**	Opt	
c	1310.88	1649	1.80	Opt	**32**	**0.03**	Opt	
d	818.44	**2984**	3.80	Opt	3095	**3.41**	Opt	
e	64.28	**1473**	**1.68**	Opt	5313	6.96	Opt	
5-20-10								
a	827.62	**212**	**0.23**	Opt	4219	4.42	Opt	
b	1684.84	**212**	**0.24**	Opt	6241	8.46	Opt	
c	2692.50	**1212**	1.74	Opt	1759	**1.62**	Opt	
d	4140.67	**1303**	2.04	Opt	1491	**1.23**	Opt	
e	1577.06	**889**	**1.19**	Opt	20713	40.96	Opt	
5-20-20								
a	2578.20	9855	22.77	Opt	**6271**	**6.35**	Opt	
b	1962.73	**5945**	**11.60**	**Opt**	70073	300 (limit)	---	2123.09
c	7311.26	2651	3.88	Opt	**2149**	**3.08**	Opt	
d	4152.51	**3044**	**4.23**	Opt	10191	15.93	Opt	
e	1976.77	**1128**	**1.35**	Opt	41425	133.57	Opt	
5-30-30								
a	3525.16	50489	302.36	**Opt**	98340	600 (limit)	---	4367.30
b		77845	600 (limit)	**3042.64**	104416	600 (limit)	---	4053.62
c	1477.95	**11894**	**28.32**	**Opt**	102238	600 (limit)	---	3186.46
d	7988.01	**14839**	**40.31**	Opt	41153	157.48	Opt	
e		76331	600 (limit)	**5580.16**	111776	600 (limit)	---	6052.31
5-40-40								
a		52824	600 (limit)	**1718.89**	112624	600 (limit)	---	5437.49
b		57689	600 (limit)	**6077.09**	114789	600 (limit)	---	6497.47

c		79760	600 (limit)	**7620.52**	116402	600 (limit)	---	8475.72
d	4841.82	76215	573.79	**Opt**	109404	600 (limit)	---	6461.62
e		56270	600 (limit)	**5996.256**	113935	600 (limit)	---	6982.90
5-50-50								
a		51939	600 (limit)	**3614.24**	119051	600 (limit)	---	6318.37
b		50261	600 (limit)	**5949.92**	97272	600 (limit)	---	8744.37
c		51730	600 (limit)	**8500.70**	112299	600 (limit)	---	11858.44
d		52779	600 (limit)	**2342.99**	110513	600 (limit)	---	6400.02
e		54411	600 (limit)	**4397.52**	108675	600 (limit)	---	7032.82
10-20-20								
a	2598.90	58212	900 (limit)	2518.44	**97826**	**541.87**	**Opt**	
b		55053	900 (limit)	**1730.09**	126554	900 (limit)	---	2037.00
c		82231	900 (limit)	**2932.42**	126489	900 (limit)	---	2992.37
d	3692.22	77408	900 (limit)	3692.22 (Opt)	**29186**	**65.45**	**Opt**	
e	2240.98	60715	900 (limit)	1866.31	**83629**	**454.29**	**Opt**	
10-50-50								
a		50964	1800 (limit)	**1864.58**	205349	1800 (limit)	---	6842.56
b		55186	1800 (limit)	**6780.05**	205469	1800 (limit)	---	9957.25
c		66950	1800 (limit)	**7862.87**	193021	1800 (limit)	---	9981.91
d		59528	1800 (limit)	**7281.39**	206918	1800 (limit)	---	10253.44
e		58750	1800 (limit)	**5354.05**	173524	1800 (limit)	---	9001.67

--- No feasible solution found

The analysis of these results enables to conclude that:

- Both methods are computationally demanding, so they are not able to solve large problems in a reasonable time, as already pointed out by the authors for the *k-th best* algorithm (Calvete and Galé, 2011).
- In the first 4 categories (5-10-10, 5-15-15, 5-10-20, 5-20-10) with smaller problems, both methods were able to find the optimal solution to all problems and it is not possible to conclude for the superiority of one method in relation to the other. The *EEPSM* was faster in 9 problems of this sub-set with 20 problems, and the *k-th best* algorithm was faster in 11 problems. In terms of the number of bases searched, the reverse situation occurred: the *EEPSM* searched less bases in 11 problems and the *k-th best* algorithm in 9 problems. The *EEPSM* was more regular in the computational time spent in each problem, requiring times between 0.12 and 4.71 seconds, with an average of 1.54 sec., while the *k-th best* algorithm required times between 0.01 and 40.96 seconds, with an average of 4.88 sec.
- In the category 5-20-20, the *k-th best* algorithm was not able to find the optimal solution to one problem. As the problem size increases, both methods could finish the process within the pre-established time limit in a few cases only. In problems with higher numbers of lower-level variables and constraints, the *EEPSM* is always better because it returns a feasible solution to the problem while the *k-th best* algorithm does not. When the number of upper-level variables is increased from 5 to 10, and considering a small number of constraints and lower-level variables (10-20-20), the *k-th best* algorithm performs better than the *EEPSM*. This behavior was expected because, as mentioned above, the *EEPSM* is only adequate for problems with few upper-level variables. However, when the other dimensions of the problems are increased (10-50-50), the *k-th best* algorithm also cannot finish, even increasing the time limit to 1800 seconds.

- The main advantage of the *EEPSM* is that it can be used partially to find an approximation of the optimal solution, because it always returns a feasible solution (that is, a solution of *IR)* even if halted during the process. The *k-th best* algorithm cannot be stopped before reaching the end, because only the final solution is feasible, which is also optimal. From the *k-th best* algorithm we can obtain an upper bound for the optimal value of *F*.

The fact that the *EEPSM* can be used in a partial way has been the main motivation for developing a heuristic procedure.

5 A local search heuristic based on the *EEPSM*

The local search heuristic is based on the *EEPSM* and computes at least a local optimal solution – *LOH* (Local Optimum Heuristic). Given a feasible basic solution of the LBPMOLL, the procedure explores all efficient extreme solutions to the problem (3) adjacent to it with higher $F(x, y)$. The process is repeated to all these solutions, by looking for efficient extreme solutions adjacent to each one with an F higher than its own. The algorithm finishes when there are no more adjacent solutions that improve *F*.

The aim of the heuristic is to reduce the computational effort with respect to the complete *EEPSM* algorithm, generating a solution that is at least a local optimum of the LBPMOLL.

In order to detect trivial problems, in which the optimal solution is given by optimizing $F(x, y)$ in S, a step 0 has been included (both in the heuristic and in the *EEPSM*) similar to the first step of the *k-th best* algorithm.

***LOH* algorithm:**

Step 0 – Solve the high-point relaxation problem of the LBPMOLL: $\max F(x, y)$ s.t. $(x, y) \in S$. Check if the solution obtained is efficient to the lower-level problem. If it is efficient, then the algorithm finishes because this solution is optimal to the LBPMOLL; otherwise, proceed to Step 1.

Step 1 – Choose an initial weight vector for the MOLP problem (3): $\lambda \in \mathbb{R}^{\varrho}_{>0}$

Step 2 – Compute a first basic efficient solution of (3) by solving a weighted-sum with the λ defined in Step 1.

Step 3 – Determine the nonbasic efficient variables w.r.t. the current basis (as in step 2 of the *EEPSM*), but considering only the variables that lead to solutions with an *F* value higher than (or equal to) the current one. Determine the corresponding bases and keep the different ones in the $\mathcal{L}_{\mathcal{B}}$ list.

Explore all bases in $\mathcal{L}_{\mathcal{B}}$ by repeating Step 3 for each one, and adding to $\mathcal{L}_{\mathcal{B}}$ the efficient bases adjacent to the current one that do not decrease *F*.

The solution with best *F* found during the process is returned.

This heuristic is very sensitive to the starting point, which is determined by the weight vector λ chosen in Step 1. Different initial weight vectors can be defined, e.g. the vector with equal weights, $\lambda =$

$(\frac{1}{\varrho}, \frac{1}{\varrho}, \cdots, \frac{1}{\varrho})$, and other weight vectors dispersed within the weight space. Thus, several runs of the heuristic may be performed with different starting points leading to different results.

This procedure can be generalized to extend the search by considering in Step 3 not just the adjacent extreme points which improve *F* (or, at least, keep it equal) but also allowing for a certain percentage δ of degradation of *F* with respect to the extreme point for which all neighbor extreme points are being inspected. By enabling worsening the objective function value, the heuristic may potentially explore a more diverse, yet controlled, selection of extreme points. The *LOH* becomes a heuristic parameterized in δ, i.e., *LOH*(δ). Therefore, *LOH*(0) is the *LOH* and *LOH*(∞) is the *EEPSM*.

6 Computational results of the heuristic

6.1 Comparison of the *LOH* with the *EEPSM*

For this comparison, we have considered the sub-set of problems described above for which at least one of the methods (*EEPSM* or the *k-th best* algorithm) did not achieve the optimal solution. Therefore, the 30 problems in the following categories were used: 5-20-20, 5-30-30, 5-40-40, 5-50-50, 10-20-20, 10-50-50.

We have performed *N* runs of the *LOH* for each instance, considering a pre-defined set of *N* weight vectors for starting the search. This set has been defined as follows: ϱ extreme weight vectors from $\lambda^{(1)} =$ (1,0,0…,0) to $\lambda^{(\varrho)} =$ (0,0,0,….,1), a vector in which the total weight is distributed by the *p*=2 lower-level objective functions, $\lambda^{(\varrho+1)} = (1/2,\ 1/2,\ 0,\ 0,\ldots,0)$, and a vector with equal weights, $\lambda^{(\varrho+2)} = (\frac{1}{\varrho}, \frac{1}{\varrho}, \cdots, \frac{1}{\varrho})$. The 0's in the weight vectors were replaced by a small positive value (ε=0.0001) in order to ensure that the solutions obtained are efficient rather than weakly efficient only. Therefore, $N = \varrho + 2$ runs were performed for each instance, i.e., 10 runs for the problems with 5 upper-level variables (5-20-20, 5-30-30, 5-40-40, 5-50-50) and 15 runs for the problems with 10 upper-level variables (10-20-20, 10-50-50). A time limit was imposed to each run such that the sum of the times of all runs do not exceed the time limit given to the *EEPSM* for the same problem. Therefore, the following time limits were considered per run: 30 s (=300/10) for problems 5-20-20, 60 s (=600/10) for categories 5-30-30 to 5-50-50, 60 s (=900/15) for 10-20-20, and 120 s (=1800/15) for 10-50-50. The time limit of 30 seconds for problems 5-20-20 was never achieved.

A summary of the results obtained and a comparison with the results of *EEPSM* is presented in Table 2. For the sake of clarity and ease of comparison, we repeat the *EEPSM* values already shown in Table 1. Regarding the heuristic, the following information is shown: the sum of the running times (in seconds) spent in the *N* runs; the value of the *F* in the best solution obtained; the number of times the best solution was yielded over the *N* runs; an indication of whether the heuristic was able to finish within the time limit per run, displaying the number of runs it happened over the *N* runs (column "Finish?"). For each problem, the best *F* value is highlighted in bold face. The '*' denotes that the optimal value was obtained.

Table 2 – Comparison of the *LOH* with the *EEPSM*

		EEPSM		***LOH***			
Problem (n_1-n_2-m)	optimal F	time (seconds)	best F	Σ times (seconds)	best F	# best solution found / N runs	Finish?
5-20-20				N = 10 runs; time limit = 30 s/run			
a	2578.20	22.77	**2578.20 ***	90.00	**2578.20 ***	10/10	10/10
b	1962.73	11.60	**1962.73 ***	2.70	**1962.73 ***	6/10	10/10
c	7311.26	3.88	**7311.26 ***	9.42	**7311.26 ***	7/10	10/10
d	4152.51	4.23	**4152.51 ***	3.70	**4152.51 ***	9/10	10/10
e	1976.77	1.35	**1976.77 ***	2.81	**1976.77 ***	10/10	10/10
5-30-30				N = 10 runs; time limit = 60 s/run			
a	3525.16	302.36	**3525.16 ***	312.15	**3525.16 ***	7/10	8/10
b		600 (limit)	**3042.64**	380.42	**3042.64**	6/10	6/10
c	1477.95	28.32	**1477.95 ***	44.14	**1477.95 ***	9/10	10/10
d	7988.01	40.31	**7988.01 ***	53.25	**7988.01 ***	8/10	10/10
e		600 (limit)	**5580.16**	134.42	**5580.16**	7/10	8/10
5-40-40				N = 10 runs; time limit = 60 s/run			
a		600 (limit)	1718.89	361.32	**2149.08**	2/10	4/10
b		600 (limit)	6077.09	446.75	**6180.45**	6/10	3/10
c		600 (limit)	7620.52	230.51	**7865.92**	3/10	7/10
d	4841.82	600 (limit)	**4841.82 ***	161.03	**4841.82 ***	8/10	8/10
e		600 (limit)	**5996.26**	135.86	**5996.26**	7/10	8/10
5-50-50				N = 10 runs; time limit = 60 s/run			
a		600 (limit)	**3614.24**	232.41	**3614.24**	7/10	7/10
b		600 (limit)	5949.92	487.95	**7747.93**	1/10	2/10
c		600 (limit)	8500.70	347.19	**9043.38**	4/10	5/10
d		600 (limit)	2342.99	252.14	**3364.72**	2/10	7/10
e		600 (limit)	4397.52	600 (limit)	**5294.50**	1/10	0/10
10-20-20				N = 15 runs; time limit = 60 s/run			
a	2598.90	900 (limit)	2518.44	707.81	**2598.90 ***	7/15	4/15
b		900 (limit)	1730.09	603.67	**1954.59**	5/15	6/15
c		900 (limit)	**2932.42**	267.97	**2932.42**	14/15	13/15
d	3692.22	900 (limit)	**3692.22 ***	282.54	**3692.22 ***	14/15	11/15
e	2240.98	900 (limit)	1866.31	692.88	**2240.98 ***	2/15	4/15
10-50-50				N = 15 runs; time limit = 120 s/run			
a		1800 (limit)	1864.58	1800 (limit)	**5376.78**	1/15	0/15
b		1800 (limit)	6780.05	1800 (limit)	**8675.49**	1/15	0/15
c		1800 (limit)	7862.87	1800 (limit)	**8841.17**	1/15	0/15
d		1800 (limit)	7281.39	1800 (limit)	**8308.79**	1/15	0/15
e		1800 (limit)	5354.05	1800 (limit)	**6169.35**	1/15	0/15

* Optimal solution

The following conclusions can be drawn from this experiment:

- In all problems for which the optimal solution was reached by the *EEPSM*, the *LOH* also reached the optimal solution in at least half of the runs.
- The use of the heuristic may not be justified in the smallest problems of this experiment (5-20-20) because, although it always reaches the optimal solution, the total time (sum of the times of the 10 runs) is, in general, higher that running the *EEPSM* until the end; in addition, the *EEPSM* has the advantage of ensuring the optimality of the solution.
- In the problems for which we have no guarantee that the optimal solution is known (a total of 18 over 30 problems), the *LOH* improved the solution obtained by the *EEPSM* execution in 13 problems and

attained the same solution as the *EEPSM*'s in 5 problems; this means that the outcome of *N* runs of the *LOH* was never worse than the one of the EEPSM, being substantially better in several problems.

- This comparison experiment was calibrated in terms of the maximum computational time given to each run of the *LOH* so that the limit of the sum of the times of all runs ('Σ times') was equal to the maximum time given to the *EEPSM*. However, the *LOH* may not use its time budget in some runs because it may finish before. Only in the last category of problems (10-50-50) the total times are equal in the two algorithms; but, even consuming, in general, less time than the *EEPSM*, the *LOH* presented better results.

Thus, the *LOH* seems interesting for problems in which the number of efficient bases is significantly high, yielding better solutions than the *EEPSM* for a similar or smaller computational effort.

6.2 Comparison of the heuristic with different parameterizations

We now compare the results of the *LOH* with the generalized *LOH* for the tolerance values of δ =10% and δ=20%. For a given extreme point with $F = F'$, that tolerance means that all adjacent extreme points with $F \geq (1 - \delta/100)F'$ are accepted. This comparative study was made considering the 3 most challenging categories of problems. The results are displayed in Table 3.

Table 3 shows, for each problem, the best and the average of *F* obtained in *N* runs. We omit the computational times because the times for *LOH* were presented in Table 2, and *LOH*(10%) and *LOH*(20%) always reached the time limit, even in the problems in which *LOH* did not; this happens because the scope of the search is extended when δ increases. Thus, the total times consumed by *LOH*(10%) and *LOH*(20%) were: 600 s, 900 s and 1800 s, respectively in problems 5-50-50, 10-20-20 and 10-50-50. In these 3 categories of problems, the results of the *LOH* are in general better, and never worse, than the ones obtained with *LOH*(10%) and *LOH*(20%) regarding the best and the average values of *F*. The results worsened with the increase of the tolerance δ. The highest value in each row of Table 3 is highlighted in bold.

Table 3 – Comparison of the heuristic with different tolerance values

Problem (n_1-n_2-m)	***LOH***		***LOH*(10%)**		***LOH*(20%)**	
	best *F*	Average *F*	best *F*	Average *F*	best *F*	Average *F*
5-50-50						
a	**3614.24**	3333.50	**3614.24**	2812.97	**3614.24**	2607.69
b	**7747.93**	4963.64	5820.97	4331.12	5657.50	4204.01
c	**9043.38**	7787.95	8100.62	6786.54	8325.65	6622.04
d	**3364.72**	1621.21	**3364.72**	1684.39	**3364.72**	1318.20
e	**5294.50**	4341.25	4991.92	3779.78	4991.92	3682.07
10-20-20						
a	**2598.90 ***	2340.48	**2598.90 ***	2169.92	**2598.90 ***	2136.64
b	**1954.59**	1888.43	**1954.59**	1848.34	**1954.59**	1811.76
c	**2932.42**	2898.39	**2932.42**	2874.16	**2932.42**	2810.06
d	**3692.22 ***	3691.40	**3692.22 ***	3654.40	**3692.22 ***	3516.28
e	**2240.98 ***	1856.43	**2240.98 ***	1836.69	**2240.98 ***	1778.86
10-50-50						
a	**5376.78**	1771.63	5368.86	1458.94	5368.86	1273.96
b	**8675.49**	6860.93	7827.14	5701.45	7827.14	5557.92

c	**8841.17**	6108.54	8657.40	5259.99	8657.40	5083.00
d	**8308.79**	5219.50	7487.16	4354.24	7289.48	4038.28
e	**6169.35**	4246.14	5709.24	3931.18	5709.24	3843.87

Due to the increased capability of *LOH*(10%) and *LOH*(20%) to promote exploration, thus enabling to escape from local optima, they could perform better than *LOH*; however, the time limit is clearly an issue due to the number of bases to be inspected.

We further studied the evolution of the value of *F* when the time limit is extended. This experiment was carried out for problems 5-50-50, with a time limit of 3600 s for the *N*=10 runs, i.e., 360 s for each run corresponding to a different vector of weights to compute the initial solution. The *Average F* value improves in all *LOH* versions, with the improvement rate (360 s vs. 60 s runtime) in *LOH*(20%) higher than in *LOH*(10%) which, in turn, is higher than in *LOH*. For illustration purposes, the evolution of the values of *Best F* and *Average F* are displayed in Fig. 5 for problems *b*, *c*, *d* and *e* (problem *a* is similar to *d* in the behavior of the *Best F* and it is similar to problems *b* and *c* in the behavior of the *Average F*). The *Best F* value improves in *LOH*(10%) and *LOH*(20%) for the problems *b*-*c*-*e*, and it improves in *LOH* just in problem *e*. In problems *a* and *d* the *Best F* coincides for all parameterizations and it does not change even increasing the time limit.

Therefore, it can be expected that for higher time limits, which may depend on the application, the *Best F* computed by *LOH*($\delta > 0$) can approach, or even exceed, the results obtained for *LOH*.

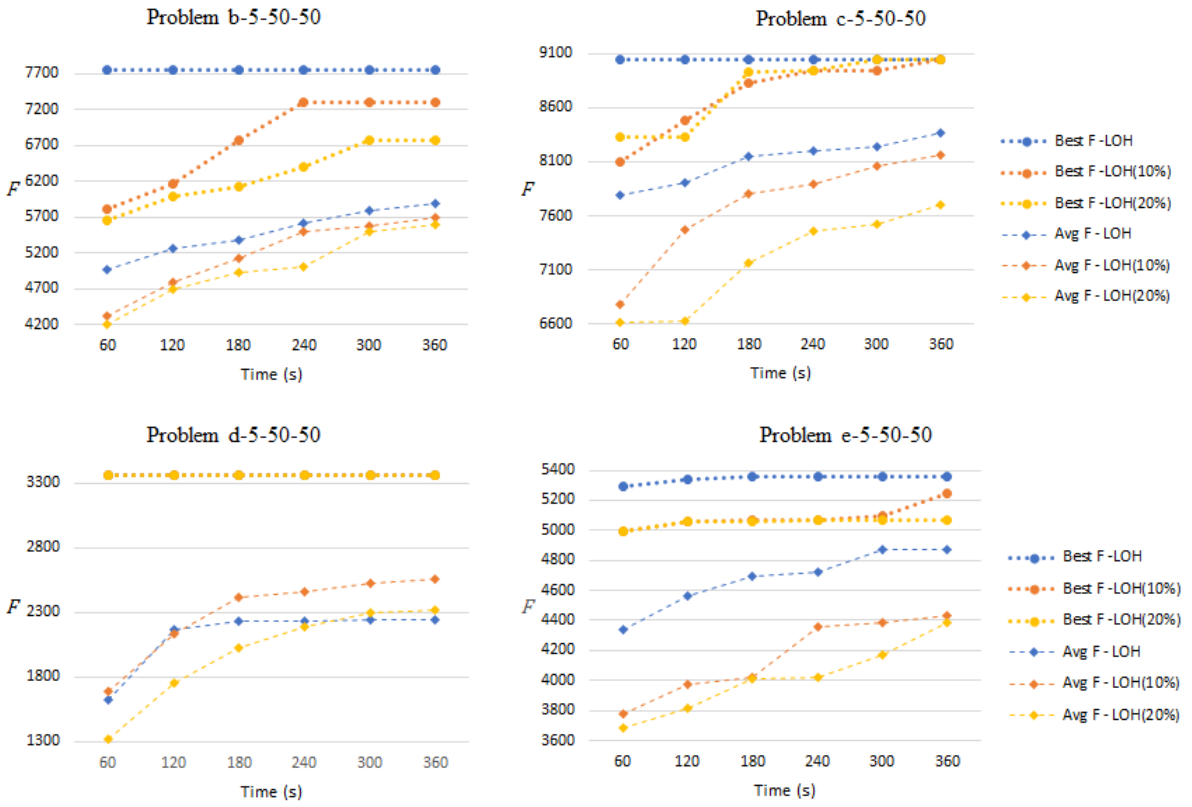

Figure 5 – Behavior of the *Best F* and the *Average F* for *LOH*, *LOH*(10%) and *LOH*(20%) with runtime 360 s.

6.3 Comparison of the *LOH* with a genetic algorithm

In order to cope with the complexity to obtain the optimal solution to the LBPMOLL and the poor performance of the exact algorithms, even in medium-sized problems, Calvete and Galé (2011) proposed a genetic algorithm (GA). This approach is specially tailored for the characteristics of the problem by using specific solution encoding (vertex solutions represented by a string of integers, the components of which are the indices of the basic variables) as well as crossover and mutation operators. The GA proceeds by making a population of basic feasible solutions to evolve using genetic operators. From two parent solutions, the (one point) crossover operator generates two extreme points which are basic feasible solutions of S. The mutation operator leads to an adjacent vertex of a given solution. As in the *k-th best* algorithm, the Benson's technique is used to check whether $(x', y') \in S$ is a point of IR. The fitness is defined lexicographically, first privileging the solutions of IR and then ranking the solutions according to the *F*-value. A kind of elitist strategy is used, keeping the best solutions (among parents and offspring) in the population.

The *LOH* was compared with this GA considering the same instances as in the experiment reported in Table 2. The main purpose is the comparison of our heuristic with another non-exact method which, according to Calvete and Galé (2011), «provided estimable results in terms of both quality of the solution an time invested». For a fair comparison, we have also implemented the GA in Delphi and tested both algorithms using the same computer.

The main common and distinct features of these two non-exact approaches are: both algorithms search for vertices only; while the heuristic proposed herein moves from one vertex to another vertex of IR, basically carrying out a local search of feasible vertices strongly dependent on the initial solution, the GA explores vertices of the entire search space S, i.e., feasible and infeasible solutions of the LBPMOLL. On one hand, the GA does a broader search, potentially being able to reach better solutions; on the other hand, it may spend a considerable amount of time searching for infeasible solutions.

For each problem, *N* independent runs of the GA were performed, the same number of runs performed with the heuristic (10 or 15, depending on the problem category – see Table 2). The *LOH* is deterministic (the different runs result from choosing a different weight vector for its beginning), while the GA is stochastic. To allow for replication of results and a better comparison of different GA parameterizations, the same random seeds were used in runs with the same index for all problems (e.g., the random seed in run 1 of the GA for problem *a* is the same as in run 1 for problem *b*).

The following GA parameter values for the GA were considered: crossover probability (p_c)=0.5 and mutation probability (p_m)=0.9, since the best results in (Calvete and Galé, 2011) were obtained with these values. For the population size (*Pop*), considering the values proposed in the original paper, which are *Pop* = n or $2n$, with $n = n_1 + n_2 + m$, we have adopted *Pop* = n in problems with $n_1 = 5$ and *Pop* = $2n$ in problems with $n_1 = 10$ (i.e., categories 10-20-20 and 10-50-50). No limit is imposed on the number of

iterations, since this is defined by the computational time given to each run, which is equal to the maximum time assigned to the *LOH*.

Although the time limit given to the *LOH* and the GA is the same, the *LOH* does not always use the whole time. The *LOH* may spend different times in the same problem depending on the initial solution (i.e., the weight vector used for each run), whereas the GA always uses the entire computational budget.

Table 4 makes a comparison between the *LOH* and the GA for the problems reported in Table 2. For each problem, it is shown the *best F* (maximum *F*), the minimum *F*, and the median *F* obtained by each algorithm in *N* runs. $F=-\infty$ means that the GA finished with no feasible solution. The existence of some $F=-\infty$ justifies why we have adopted the median as a centrality measure for the *F* value, instead of the average. Table 4 also presents: the average number of bases explored (in both algorithms), the average number of iterations of the GA and the average time of one run (in seconds) for the *LOH*. The time given to each GA run is fully used, so the average GA time per run is equal to the time limit with a possible very small increase needed to complete the generation. The highest *F*-value known for each problem is highlighted in bold. If this value is obtained in more than 50% of the runs (so that the median *F* is equal to the highest *F*-value) or in all runs (the minimum *F* is equal to the highest *F*-value), then it appears in bold also in the columns of Min *F* and Median *F*.

Table 4 – Summary of the results comparing the heuristic with the GA

		LOH					**GA**					
	Problem (n_1-n_2-m)	Best *F*	Min *F*	Median *F*	Avg bases explored	Avg time	Best *F*	Min *F*	Median *F*	Avg bases explored	Avg time	Avg iter
10 runs	5-20-20	time limit = 30 s/run					*Pop*=45, time = 30 s/run					
	a	**2578.20 ***	**2578.20**	**2578.20**	3633	9.0	**2578.20 ***	- ∞	**2578.20**	15273	30.0	362
	b	**1962.73 ***	1232.77	**1962.73**	175	0.3	**1962.73 ***	- ∞	190.04	15533	30.0	373
	c	**7311.26 ***	5995.22	**7311.26**	625	0.9	**7311.26 ***	- ∞	**7311.26**	15485	30.1	373
	d	**4152.51 ***	4050.41	**4152.51**	264	0.4	**4152.51 ***	4050.41	**4152.51**	15432	30.0	365
	e	**1976.77 ***	**1976.77**	**1976.77**	213	0.3	**1976.77 ***	**1976.77**	**1976.77**	16112	30.0	388
	5-30-30	time limit = 60 s/run					*Pop*=65, time = 60 s/run					
	a	**3525.16 ***	607.21	**3525.15**	8437	31.2	**3525.16**	- ∞	1622.70	25485	60.0	420
	b	**3042.64**	984.93	3042.64	9193	38.0	-378.62	- ∞	- ∞	25633	60.1	431
	c	**1477.95 ***	1013.06	**1477.95**	2133	4.4	**1477.95 ***	- ∞	- ∞	25627	60.1	430
	d	**7988.01 ***	1762.80	**7988.01**	2193	5.3	**7988.01 ***	- ∞	- ∞	24858	60.1	416
	e	**5580.16**	3764.80	**5580.16**	3973	13.4	- ∞	- ∞	- ∞	25194	60.1	423
	5-40-40	time limit = 60 s/run					*Pop*=85, time = 60 s/run					
	a	**2149.08**	1125.29	1579.16	6554	36.1	- ∞	- ∞	- ∞	20045	60.1	254
	b	**6180.45**	5738.78	**6180.45**	9991	44.7	- ∞	- ∞	- ∞	20867	60.1	265
	c	**7865.92**	6559.27	7591.49	5992	23.1	7620.52	- ∞	- ∞	21333	60.1	263
	d	**4841.82 ***	3841.04	**4841.82**	4442	16.1	**4841.82**	- ∞	- ∞	20284	60.1	255
	e	**5996.26**	3798.83	**5996.26**	3134	13.6	- ∞	- ∞	- ∞	19993	60.1	254
	5-50-50	time limit = 60 s/run					*Pop*=105, time = 60 s/run					
	a	**3614.24**	2307.36	**3614.24**	4495	23.2	- ∞	- ∞	- ∞	19859	60.2	200
	b	**7747.93**	2279.36	5489.83	8211	48.8	- ∞	- ∞	- ∞	18660	60.2	189
	c	**9043.38**	5340.29	8986.55	5547	34.7	- ∞	- ∞	- ∞	15248	60.2	153
	d	**3364.72**	-4142.64	2547.55	3838	25.2	-4060.71	- ∞	- ∞	15799	60.2	158
	e	**5294.50**	1882.91	4289.60	9423	60.0	- ∞	- ∞	- ∞	15893	60.1	159

15 runs												
	10-20-20	time limit = 60 s/run					*Pop*=100, time = 60 s/run					
	a	**2598.90 ***	1562.75	2586.67	8229	47.2	**2598.90 ***	2586.67	**2598.90**	31511	60.1	335
	b	**1954.59**	1735.96	1898.71	7798	40.2	**1954.59**	1848.10	**1954.59**	31579	60.1	334
	c	**2932.42**	2421.99	**2932.42**	4811	17.9	**2932.42**	- ∞	**2932.42**	30934	60.1	332
	d	**3692.22 ***	3679.95	**3692.22**	4828	18.8	**3692.22 ***	**3692.22**	**3692.22**	31504	60.1	339
	e	**2240.98 ***	751.76	1863.94	8209	46.2	**2240.98**	1899.88	**2240.98**	31211	60.1	331
	10-50-50	time limit = 120 s/run					*Pop*=220, time = 120 s/run					
	a	5376.78	-3818.03	1864.58	7662	120.0	**5390.37**	- ∞	- ∞	33334	120.4	155
	b	8675.49	2179.95	7659.48	7588	120.0	**9604.63**	- ∞	9186.27	33662	120.3	147
	c	8841.17	-2821.93	6896.19	10012	120.0	**8843.39**	- ∞	-7448.36	33332	120.3	150
	d	**8308.79**	-1203.25	4873.94	10295	120.0	**8308.79**	- ∞	- ∞	32937	120.3	148
	e	**6169.35**	-1986.48	4916.09	9173	112.4	5814.81	- ∞	- ∞	33614	120.4	153

Analysis of the results:

- The *LOH* is clearly better than the GA on the 20 problems with $n_1 = 5$ (categories from 5-20-20 to 5-50-50) considering similar computational times. The Best F obtained with the heuristic is always higher or equal to the Best F of the GA, being strictly better in 11 problems. Likewise, the Min F and Median F are higher or equal to the corresponding values of the GA in all problems; the Min F is strictly better in 18 problems and the Median F is strictly better in 16 problems (in the other problems, the values are equal).
- In the 10 problems with $n_1 = 10$ (categories 10-20-20 and 10-50-50), the superiority of one algorithm over the other cannot be concluded for all the problems. In these categories, the Best F values obtained by the two algorithms were equal in 6 problems, the *LOH* was better in 1 problem and worse in 3 problems than GA. The Median F values were equal in 2 problems, the *LOH* was better in 4 problems and worse in 4. Regarding the worst solution (Min F), the *LOH* was inferior to the GA in 4 out of the 5 problems in the category 10-20-20, being superior to the GA in all the 5 problems 10-50-50 for which there were always some GA runs returning an infeasible solution. In category 10-50-50, the GA finished with infeasible solutions in 3 (minimum) to 11 (maximum) runs over the N=15 runs performed for each problem *a* - *e*. The three problems in which the GA finished without finding any feasible solution for at least 8 runs (i.e., problems *a*, *d*, *e*) are those where the Median F is $-\infty$.

These results reinforce our conviction that the proposed algorithmic approach is suitable for problems with a reduced number of variables in the upper level. However, this limitation is not so strict in the other dimensions of the problem (number of lower-level variables and constraints), showing that the *LOH* is competitive and provides interesting results in medium-sized problems.

7 Conclusions

In this paper, we proposed an exact method to solve the optimistic formulation of the linear bilevel programming problem with multiple objective functions at the lower level (semivectorial bilevel problems). This method is based on a proposition stating that an optimistic optimal solution to the problem is an efficient extreme point of an associated MOLP problem with as many objective functions as the

number of lower-level objective functions plus the number of upper-level decision variables plus 1. Since the number of objective functions of this associated problem increases with the number of upper-level decision variables, and the number of efficient extreme points of a MOLP problem grows very quickly with the increase of the number of objective functions, this method is mainly adequate to bilevel problems with a small number of upper-level decision variables. This number is the dimension with the major impact on the computational effort required by the method.

In order to obtain better quality solutions for problems where the exact method does not reach the optimal solution within a reasonable computational time, we have also developed a local search heuristic based on the same proposition. Despite being deterministic, the heuristic can lead to distinct final solutions if different starting points are considered. Thus, we can perform several quick runs of the heuristic starting at different points, which altogether can yield a very good solution to the problem in a short computational time. The heuristic showed to be quite effective in problems where the global optimum is difficult to achieve and it surpassed the exact method in these problems by computing better solutions in a similar time. Furthermore, the heuristic can be parameterized to extend the search to a larger neighborhood, including not just the adjacent extreme points that improve the leader's objective function with respect to the current solution, but also allowing for a certain degradation of its value.

The exact method *EEPSM* and the heuristic *LOH* were compared with another exact method and a genetic algorithm, which are state-of-the-art algorithms for the problem addressed in this paper. The algorithms we propose have shown good quality results, outperforming the other algorithms under comparison in the problems with few upper-level decision variables. The main advantage of the strategy employed by our algorithms (the exact and the heuristic one) is that they can be interrupted at any moment always yielding a feasible solution to the bilevel problem. This does not occur in an exact algorithm based on the *k-th best* search (which only reaches the inducible region when the optimum is found) or in a metaheuristic based on the search of extreme points of the entire constraint region. This advantage is particularly relevant in larger problems for which the algorithms that work with feasible and infeasible solutions often finish with an infeasible solution.

In addition to the exact method and the heuristic to semivectorial bilevel problems, another contribution of this work is the development and implementation of an effective vector maximum algorithm (multiobjective simplex method) that allows the computation of all efficient extreme points of general MOLP problems. This algorithm has been extensively experimented, not only within the scope of the approaches proposed herein, but also in general MOLP problems with fewer objective functions (3 to 6) but larger numbers of constraints and decisions variables. These experiments led to several enhancements that strengthened the algorithm to be numerically robust. We strongly believe that this contribution can also be very useful for practitioners and researchers in the field of multiobjective optimization.

Acknowledgment. This work has been funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., Projects UIDB/05037/2020, UIBD/00308/2020, MAnAGER (POCI-01-0145-FEDER-028040) and RTCARE (POCI-01-0145-FEDER-028030).

8 References

Alves, M. J. and Antunes, C. H. (2018) 'A semivectorial bilevel programming approach to optimize electricity dynamic time-of-use retail pricing', *Computers and Operations Research*, 92, pp. 130–144. doi: 10.1016/j.cor.2017.12.014.

Alves, M. J., Antunes, C. H. and Carrasqueira, P. (2015) 'A PSO Approach to Semivectorial Bilevel Programming: Pessimistic, Optimistic and Deceiving Solutions', in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015)*, pp. 599–606. doi: 10.1145/2739480.2754644.

Alves, M. J., Antunes, C. H. and Costa, J. P. (2019) 'Multiobjective Bilevel Programming: Concepts and Perspectives of Development', in Doumpos, M. et al. (eds) *New Perspectives in Multiple Criteria Decision Making: Innovative Applications and Case Studies*. Cham: Springer International Publishing, pp. 267–293. doi: 10.1007/978-3-030-11482-4_10.

Alves, M. J., Antunes, C. H. and Costa, J. P. (2021) 'New concepts and an algorithm for multiobjective bilevel programming: optimistic, pessimistic and moderate solutions', *Operational Research*, 21(4), pp. 2593–2626. doi: 10.1007/s12351-019-00534-9.

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

Alves, M. J., Dempe, S. and Júdice, J. J. (2012) 'Computing the Pareto frontier of a bi-objective bi-level linear problem using a multiobjective mixed-integer programming algorithm', *Optimization*. Taylor & Francis, 61(3), pp. 335–358.

Ankhili, Z. and Mansouri, A. (2009) 'An exact penalty on bilevel programs with linear vector optimization lower level', *European Journal of Operational Research*, 197(1), pp. 36–41. doi: 10.1016/j.ejor.2008.06.026.

Antunes, C. H., Alves, M. J. and Clímaco, J. (2016) *Multiobjective Linear and Integer Programming*. Springer International Publishing. doi: 10.1007/978-3-319-28746-1.

Benson, H. P. (1978) 'Existence of efficient solutions for vector maximization problems', *Journal of Optimization Theory and Applications*, 26(4), pp. 569–580. doi: 10.1007/BF00933152.

Benson, H. P. (1998) 'An Outer Approximation Algorithm for Generating All Efficient Extreme Points in the Outcome Set of a Multiple Objective Linear Programming Problem', *Journal of Global Optimization*, 13(1), pp. 1–24. doi: 10.1023/A:1008215702611.

Bialas, W. F. and Karwan, M. H. (1984) 'Two-level linear programming', *Management Science*, 30(8), pp. 1004–1020.

Bonnel, H. (2006) 'Optimality conditions for the semivectorial bilevel optimization problem', *Pacific Journal of Optimization*, 2(3), pp. 447–468.

Bonnel, H. and Morgan, J. (2006) 'Semivectorial bilevel optimization problem: penalty approach', *Journal of Optimization Theory and Applications*, 131(3), pp. 365–382. doi: 10.1007/s10957-006-9150-4.

Calvete, H. and Galé, C. (2011) 'On linear bilevel problems with multiple objectives at the lower level', *Omega*, 39(1), pp. 33–40. doi: 10.1016/j.omega.2010.02.002.

Dauer, J. P. and Liu, Y.-H. (1990) 'Solving multiple objective linear programs in objective space', *European Journal of Operational Research*, 46(3), pp. 350–357. doi: 10.1016/0377-2217(90)90010-9.

Dempe, S. (2002) *Foundations of bilevel programming*. Springer US. doi: 10.1007/b101970.

Dempe, S. and Mehlitz, P. (2020) 'Semivectorial bilevel programming versus scalar bilevel programming', *Optimization*. Taylor & Francis, 69(4), pp. 657–679. doi: 10.1080/02331934.2019.1625900.

Ehrgott, M. (2005) *Multicriteria Optimization*. 2nd edn. Springer-Verlag Berlin Heidelberg. doi: 10.1007/3-540-27659-9.

Evans, J. P. and Steuer, R. E. (1973) 'A revised simplex method for linear multiple objective programs',

Mathematical Programming, 5(1), pp. 54–72. doi: 10.1007/BF01580111.

Fülöp, J. (1993) *On the equivalence between a linear bilevel programming problem and linear optimization over the efficient set*. Working paper no. WPO 93-1, Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences.

Isermann, H. (1977) 'The Enumeration of the Set of All Efficient Solutions for a Linear Multiple Objective Program', *Operational Research Quarterly*. Palgrave Macmillan Journals, 28(3), pp. 711–725.

Löhne, A. and Weißing, B. (2017) 'The vector linear program solver Bensolve – notes on theoretical background', *European Journal of Operational Research*, 260(3), pp. 807–813. doi: https://doi.org/10.1016/j.ejor.2016.02.039.

Lv, Y. and Wan, Z. (2014) 'A solution method for the optimistic linear semivectorial bilevel optimization problem', *Journal of Inequalities and Applications*, 2014(1), p. 164. doi: 10.1186/1029-242X-2014-164.

Ren, A. and Wang, Y. (2016) 'A novel penalty function method for semivectorial bilevel programming problem', *Applied Mathematical Modelling*, 40(1), pp. 135–149. doi: 10.1016/j.apm.2015.04.041.

Rudloff, B., Ulus, F. and Vanderbei, R. (2017) 'A parametric simplex algorithm for linear vector optimization problems', *Mathematical Programming*, 163(1), pp. 213–242. doi: 10.1007/s10107-016-1061-z.

Schechter, M. and Steuer, R. (2005) 'A correction to the connectedness of the Evans-Steuer algorithm of multiple objective linear programming', *Foundations of Computing and Decision Sciences*, pp. 351–359.

Sinha, A., Malo, P. and Deb, K. (2018) 'A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications', *IEEE Transactions on Evolutionary Computation*, 22(2), pp. 276–295. doi: 10.1109/TEVC.2017.2712906.

Steuer, R. E. (1986) *Multiple Criteria Optimization: Theory, Computation, and Application*. Wiley (Series in Probability and Statistics).

Zheng, Y., Chen, J. and Cao, X. (2014) 'A Global Solution Method for Semivectorial Bilevel Programming Problem', *Filomat*, 28(8), pp. 1619–1627. doi: DOI 10.2298/FIL1408619Z.

Zheng, Y. and Wan, Z. (2011) 'A solution method for semivectorial bilevel programming problem via penalty method', *Journal of Applied Mathematics and Computing*, 37(1–2), pp. 207–219. doi: 10.1007/s12190-010-0430-7.

Zionts, S. and Wallenius, J. (1980) 'Identifying Efficient Vectors: Some Theory and Computational Results', *Operations Research*, 28(3-part-ii), pp. 785–793. doi: 10.1287/opre.28.3.785.

Appendix

The algorithm to compute all efficient extreme points of a MOLP problem was applied to 20 instances with 100 variables, 50 constraints and 3 to 6 objective functions used in (Alves and Costa, 2009), which are available at the internet (http://www4.fe.uc.pt/mjalves). Since these instances were randomly generated with no particular structure, they are not degenerate. Thus, the number of efficient extreme points is equal to the number of bases explored. Table A.1 summarizes the results obtained, illustrating the performance of the algorithm. This table presents the minimum, maximum and average number of efficient extreme solutions (min $|\mathcal{E}|$, max $|\mathcal{E}|$, avg $|\mathcal{E}|$) in each group of five instances with the same number of objective functions. The corresponding computational times (in seconds) are also shown.

Table A.1- Experiments of the vector-maximum algorithm in general MOLP problems

No. of instances	No. of objectives	No. of variables	No. of constraints	min $\|\mathcal{E}\|$	max $\|\mathcal{E}\|$	avg $\|\mathcal{E}\|$	Min time	Max time	Avg time
5	3	100	50	326	444	393.4	0.44	0.59	0.53
5	4	100	50	1 929	5 513	3 387.6	3.05	9.88	5.97
5	5	100	50	6 736	25 264	14 457.4	23.03	103.95	58.90
5	6	100	50	34 734	75 624	53 606.8	282.03	814.44	601.30