Journal of Global Optimization (2005) **31:** 1–16

A Bilinear Algorithm for Optimizing a Linear Function over the Efficient Set of a Multiple Objective Linear Programming Problem

JESÚS M. JORGE
Department of Estadística, Investigación Operativa y Computación, University of La Laguna, 38271, La Laguna, Tenerife, Spain (e-mail: jjorge@ull.es)

(Received 16 September 2003; accepted 12 October 2003)

Abstract. The problem Q of optimizing a linear function over the efficient set of a multiple objective linear program serves several useful purposes in multiple criteria decision making. However, Q is in itself a difficult global optimization problem, whose local optima, frequently large in number, need not be globally optimal. Indeed, this is due to the fact that the feasible region of Q is, in general, a nonconvex set. In this paper we present a monotonically increasing algorithm that finds an exact, globally-optimal solution for Q. Our approach does not require any hypothesis on the boundedness of neither the efficient set E^P nor the optimal objective value. The proposed algorithm relies on a simplified disjoint bilinear program that can be solved through the use of well-known specifically designed methods within nonconvex optimization. The algorithm has been implemented in *C* and preliminary numerical results are reported.

Key words. bilinear programming, global optimization, multiple objective linear programming, optimization over efficient sets.

1. Introduction

Let X be a pointed polyhedron of R^n and let C be a $k \times n$ real matrix with $k \geqslant 2$. Then the multiple objective linear programming problem or, as usually abbreviated MOLP problem, may be written as:

$$\text{VMAX}\{Cx / x \in X\}. \tag{1}$$

We will suppose, without loss of generality, that:

$$X = \{x \in R^n / Ax = b,\ x \geqq 0\} \tag{2}$$

where $A \in R^{m \times n}$ and $b \in R^m$ are fixed real matrices.

Let P be a MOLP problem such as the one given in (1). Since typically the objective functions of P are conflicting, the concept of an efficient solution becomes especially useful in the analysis and resolution of P.

DEFINITION 1.1. A point $\bar{x} \in X$ is said to be an efficient solution of P if, and only if, there exists no $x \in X$ such that $Cx \geqq C\bar{x}$ and $Cx \neq C\bar{x}$.

Throughout this paper we will use the following notation for vectors: For $x, y \in R^n$, $x \geqslant y$ means $x \geqq y$ and $x \neq y$.

We shall denote by E^P the efficient set, that is, the set of all efficient solutions, for problem P.

The problem of main concern in this paper is the optimization of a linear function over the efficient set E^P of MOLP P. This problem, denoted henceforth as Q, may be written as

$$\max\{v^t x / x \in E^P\} \tag{3}$$

where $v \in R^n$.

Since the importance of problem Q is beyond all doubt, having in fact been discussed extensively in the literature (see, for example, Benson and Lee (1996)), it will not be treated here.

From a mathematical point of view, problem Q is a difficult global optimization problem because its feasible region, E^P, is in general, a nonconvex set. Thus, it can have many local optima that are not global.

Through the years, several procedures for solving Q have been suggested, including of course the brute force method that makes use of the complete enumeration of the efficient extreme points. To our knowledge, the first algorithm that attempted to solve Q in a smart fashion was schematically described by Philip (1972). This method was based on cutting planes. Later, Isermann and Steuer (1987) outlined a similar procedure for the special case of minimizing an objective function of problem P over E^P. However, neither of these authors dealt with implementation aspects of the algorithms. Ecker and Song (1994) developed detailed methods for solving Q, based on Philip's approach. But perhaps one of the authors that has spent more effort on studying problem Q is Benson. Besides theoretical studies, such as Benson (1984), where he analyzes the mathematical structure of the problem, Benson has proposed several implementable methods for Q which use different approaches than the previously employed. Among others, in Benson (1991a) and Benson (1992) two relaxation algorithms are suggested. More recently, Sayin (2000) has proposed a branch and bound algorithm that carries out a top-down search of the efficient faces of the underlying MOLP problem. Due to the difficulties found in the general solving of Q, a few special-case procedures have been developed (see, for instance, Benson and Sayin (1994) and Benson (1993)) and some heuristic methods for approximating an optimal solution have also been proposed (see, e.g., Benson and Sayin (1993) and Sayin (2000)).

In this paper we present a new proposal for finding an exact, globally-optimal solution for problem Q. This algorithm needs to solve, in each iteration, two subproblems: one of them is a simplified bilinear program and the other is an ordinary scalar linear programming problem.

The algorithm presents a number of advantages, such as those of being implementable, monotonically increasing and independent of any hypothesis on the boundedness of the efficient set E^P or of the optimal objective value.

The organization of this article is as follows: Section 2 summarizes some relevant results concerning faces of a polyhedron and efficiency for the MOLP problem. In the section that follows we give the theoretical foundations of our procedure. In particular, we will see how we can choose an efficient face of the efficient region that improves a certain objective level. Section 4 provides a detailed statement of the algorithm and proves its convergence after a finite number of iterations. Additionally, a small example problem is solved for illustration purposes. Section 5 contains our computational results on randomly generated test problems. Some concluding remarks are given in the last section.

2. Definitions and Preliminaries

2.1. IDENTIFYING FACES OF A POLYHEDRON

For the purposes of our algorithm we need a convenient manner of identifying an arbitrary face of a polyhedron.

First, let us start with a known definition.

DEFINITION 2.1 (Rockafellar (1970)). Let F be a subset of X. F is a face of X if every line segment in X with a relative interior point in F has both end-points in F.

There are many different forms for characterizing faces of a polyhedron. Among others (see, for example, Yu and Zeleny (1975) or Murty (1985)), we have at our disposal the following proposal developed in Jorge (2003). Let $J' \subseteq J = \{1, \ldots, n\}$.

DEFINITION 2.2 (Jorge (2003), p. 125). We will say that a subset $J' \subseteq J$ is a descriptor set for a face F of X if, and only if, $F = F(J')$ where $F(J')$ is given by $\{x \in X / x_j = 0, \forall j \in J'\}$.

It can be proven that every face F of X can be described (not necessarily in a unique form) through a descriptor set (see, Jorge (2003)).

For notational convenience, we define R^n_+ and R^k_{++} as $\{x \in R^n / x \geqq 0\}$ and $\{x \in R^k / x > 0\}$, respectively. Then, having in mind the linear constraint representation of X given in (2), it is obvious that:

$$F(J') = \{x \in R^n_+ / A^{J-J'} x_{J-J'} = b, x_{J'} = 0\}$$

It is easy to check that, in general, the larger the descriptor set is, the smaller the face it describes.

An arbitrary face can have several different descriptor sets associated, which in turn constitutes a disadvantage for characterizing purposes. This motivates the definition of a new concept that overrides this difficulty.

DEFINITION 2.3 (Jorge (2003), Definition 4). We will say that a subset $J' \subseteq J$ is a maximal descriptor set if, and only if, there exists no subset $J'' \subseteq J$ that contains J' as a strict subset and verifies $F(J'') = F(J')$.

The importance of the above concept lies in the one-to-one correspondence that can be established between the faces of a polyhedron and their maximal descriptor sets, as the following result states:

THEOREM 2.1 (Jorge (2003), Theorem 3). *Every nonempty face F of X has a unique maximal descriptor set associated.*

2.2. SOME RESULTS ON EFFICIENCY FOR THE MOLP

DEFINITION 2.4. A face F of X is said to be efficient for problem P if all the elements of F are efficient, that is, $F \subseteq E^P$. E_f^P will denote the set of all efficient faces of problem P.

Recall that, as is common knowledge, E^P can be described as the union of all efficient faces of P (see, for example, Steuer (1986)).

Now taking $\lambda \in R^k$, it is possible to define a collection of parametric scalar programs associated to problem (1) as,

$$P_\lambda \equiv \max\{\lambda^t Cx / x \in X\}.$$

If we denote by S_{P_λ} the set of all optimal solutions of P_λ, the following result is well-known:

THEOREM 2.2 (Evans and Steuer (1973), Corollary 1.4). *$\bar{x} \in E^P$ if, and only if, $\exists \lambda \in R^k_{++}$ such that $\bar{x} \in S_{P_\lambda}$.*

Denoting by e a vector whose entries are each equal to one and applying linear duality to the above result we can obtain:

COROLLARY 2.1. *Let $X \neq \varnothing$. Then, $E^P \neq \varnothing$ if, and only if, the system $u^t A - \lambda^t C \geqq 0^t, \lambda \geqq e$ has a solution.*

There are a great variety of procedures that conclude the efficiency of a face. Particularly, the next result provides us with a simple test to check the efficiency of a face when it comes described through its maximal descriptor set.

THEOREM 2.3 (Jorge (2003), Corollary 5). *Let J' a maximal descriptor set. Then, $F(J') \in E_f^P$ if, and only if $\exists \lambda \in R^k_{++}, \exists s \in R^n_+, \exists u \in R^m$, verifying $\lambda^t C + u^t A + s^t = 0$ and $s_{J-J'} = 0$.*

Proof. '$\Rightarrow$' Let $\bar{x}$ be a point belonging to the relative interior of $F(J')$. Since J' is a maximal descriptor set we have that $\bar{x}_{J-J'} > 0$. Now since $\bar{x} \in E^P$, by Theorem 2.2, $\exists \lambda \in R^k_{++}$ such that $\bar{x} \in S_{P_\lambda}$. Applying the duality theory of linear programming and in particular the complementary slackness property (see, for instance, Murty (1983)) we have that $\exists \lambda \in R^k_{++}$, $\exists s \in R^n_+$, $\exists u \in R^m$, verifying $\lambda^t C + u^t A + s^t = 0$ and $s_{J-J'} = 0$.

'$\Leftarrow$' Let $\bar{\lambda} \in R^k_{++}$, $\bar{s} \in R^n_+$ and $\bar{u} \in R^m$, verifying $\bar{\lambda}^t C + \bar{u}^t A + \bar{s}^t = 0$ and $\bar{s}_{J-J'} = 0$. So, $\forall x \in F(J')$ we have that $x^t \bar{s} = 0$. Now, by the duality theory of linear programming we have that $F(J') \subseteq S_{P_{\bar{\lambda}}}$ and therefore, by Theorem 2.2, $F(J') \in E^P_f$. □

Notice that the 'if portion' of Theorem 2.3 holds independently of the maximality of the descriptor set. This property will be taken into account by the algorithm proposed in Section 4.

3. Theoretical Justification

In this section we are going to show how we can choose an efficient face of a MOLP P that improves a certain level $\alpha \in R$ through an objective function $v^t x$.

Let $J' \subseteq J = \{1, \ldots, n\}$.

First we will give an auxiliary property:

PROPOSITION 3.1. *The system*

$$\left.\begin{array}{l} Ax - by = 0 \\ v^t x - \alpha y \geqslant 1 \\ x \geqq 0,\ x_{J'} = 0,\ y \geqslant 1 \end{array}\right\} \tag{4}$$

has a solution if, and only if, $\exists \hat{x} \in F(J')$ *such that* $v^t \hat{x} > \alpha$.

Proof. '$\Rightarrow$' Let $(\bar{x}, \bar{y})$ be a solution for system (4). Set $\hat{x} = (1/\bar{y})\bar{x}$. Then $A\hat{x} = b$, $\hat{x} \geqq 0$, $\hat{x}_{J'} = 0$ and $v^t \hat{x} \geqslant \alpha + (1/\bar{y}) > \alpha$. Thus $\hat{x} \in F(J')$ and $v^t \hat{x} > \alpha$.

'$\Leftarrow$' Let $\hat{x} \in F(J')$ such that $v^t \hat{x} > \alpha$. Then we can write: $A\hat{x} = b$, $\hat{x} \geqq 0$, $\hat{x}_{J'} = 0$, $v^t \hat{x} > \alpha$. Let $\delta \in (0,1)$ such that $v^t \hat{x} - \delta \geqslant \alpha$. Taking $\bar{x} = (1/\delta)\hat{x}$ and $\bar{y} = 1/\delta \geqslant 1$ it is clear that $(\bar{x}, \bar{y})$ is a solution for system (4). □

Now, let us consider the following problem:

$$\left.\begin{array}{l} \min\ s^t x \\ \text{s.t.} \\ \lambda^t C + u^t A + s^t = 0 \\ Ax - by = 0 \\ v^t x - \alpha y \geqslant 1 \\ x, s \geqq 0,\ \lambda \geqq e,\ y \geqslant 1 \end{array}\right\} \tag{5}$$

Notice that problem (5) is a simplified disjoint bilinear program (Horst and Tuy (1996)), that is lower bounded by 0.

The importance of problem (5) is due to the following result, which is key for our purposes:

THEOREM 3.1. *Program* (5) *is bounded, with optimal objective value* 0 *if, and only if,* $\exists F \in E_f^P$ *and* $\exists \hat{x} \in F$ *such that* $v^t \hat{x} > \alpha$.

Proof.

'$\Leftarrow$' By hypothesis $\exists F \in E_f^P$. We know, by Theorem 2.1, that every nonempty face of a polyhedron has associated a unique maximal descriptor set. So, $\exists J' \subseteq J = \{1, \ldots, n\}$ maximal descriptor set such that $F = F(J')$. Since $F(J') \in E_f^P$, applying Theorem 2.3, the following system $\lambda^t C + u^t A + s^t = 0$, $s \geqq 0$, $s_{J-J'} = 0$, $\lambda \geqq e$, has a solution $(\bar{\lambda}, \bar{u}, \bar{s})$.

On the other hand, $\hat{x} \in F(J')$ verifies that $v^t \hat{x} > \alpha$. By Proposition 3.1 it is equivalent to system (4) having a solution $(\bar{x}, \bar{y})$. Thus, it is clear that $(\bar{\lambda}, \bar{u}, \bar{s}, \bar{x}, \bar{y})$ is a feasible solution for program (5) that satisfies $\bar{s}^t \bar{x} = \bar{s}_{J'}^t \bar{x}_{J'} + \bar{s}_{J-J'}^t \bar{x}_{J-J'} = 0$. Having in mind that problem (5) is lower bounded by 0, we get that program (5) is bounded, with optimal objective value 0.

'$\Rightarrow$' By hypothesis, program (5) is bounded, with optimal objective value 0. Then we can consider an optimal solution $(\bar{\lambda}, \bar{u}, \bar{s}, \bar{x}, \bar{y})$ of this problem. Let $J' = \{j \in J / \bar{s}_j > 0\}$ (notice that it is not necessarily a maximal descriptor set for $F(J')$). By the remark made on Theorem 2.3 we get $F(J') \in E_f^P$. Since $(\bar{x}, \bar{y})$ is a solution of system (4), Proposition 3.1 guarantees that $\exists \hat{x} \in F(J')$ verifying $v^t \hat{x} > \alpha$ and therefore, the proof is complete. □

Notice that the proof given for Theorem 3.1 not only proves the existence of an efficient face with the desired property of yielding an objective value greater than a fixed level, but it also provides the procedure to obtain such face ($J' = \{j \in J / \bar{s}_j > 0\}$). The algorithm proposed in the next section will take advantage of this fact.

THEOREM 3.2. *If program* (5) *is bounded, with optimal objective value strictly positive then* $v^t x \leqslant \alpha$ *for all* $x \in E^P$.

Proof. Suppose on the contrary that $\exists \bar{x} \in E^P$ verifying $v^t \bar{x} > \alpha$. Since, we can find $F \in E_f^P$ such that $\bar{x} \in F$, applying Theorem 3.1 we obtain that program (5) is bounded, with optimal objective value 0. But, this is a contradiction with the initial assumptions.

THEOREM 3.3. *Assume that* $E^P \neq \varnothing$. *If program* (5) *is infeasible then there exists no* $x \in X$ *such that* $v^t x > \alpha$.

Proof. Since $E^P \neq \varnothing$, applying Corollary 2.1 it is equivalent to system $\lambda^t C + u^t A + s^t = 0$, $s \geqq 0$, $\lambda \geqq e$, being feasible. But by hypothesis, the bilinear program

(5) is infeasible, so the system $Ax-by=0$, $v^t x-\alpha y \geqslant 1$, $x \geqq 0$, $y \geqslant 1$, must be infeasible. Thus, by Proposition 3.1, we conclude that $\nexists x \in X$ verifying $v^t x > \alpha$. □

4. The Algorithm

Since E^P can be described as the union of all efficient faces of P and having in mind that each efficient face is, from a mathematical point of view, only a polyhedron over which we can optimize a linear function very efficiently, the results of the above section suggest the following procedure for solving problem Q.

Without loss of generality, assume that $E^P \neq \varnothing$. Start with an arbitrary efficient face. Set α to the optimal objective value of $v^t x$ over the considered face and find, if possible, a new efficient face that allows a strictly improvement of level α. Now repeat the procedure over the new face found until none efficient face with the required condition can be obtained. The optimal objective value of Q is α.

According to this idea, we are going to specify in a detailed fashion an algorithm (named *EFC*) for solving problem Q.

4.1. STATEMENT OF THE ALGORITHM

Step 0. (Initialization)

If $E^P = \varnothing$, STOP. Q has not feasible solutions.

Otherwise, let $i=0$ and $\alpha^0 = v^t \hat{x}^0$, being $\hat{x}^0$ an initial efficient solution.

Step 1. (Exploration)

Solve the following bilinear programming problem T_i:

$$\left.\begin{array}{l} \min\ s^t x \\ \text{s.t.} \\ \lambda^t C + u^t A + s^t = 0 \\ Ax - by = 0 \\ v^t x - \alpha^i y \geqslant 1 \\ x, s \geqq 0, \lambda \geqq e, y \geqslant 1 \end{array}\right\}$$

Step 2. (Stopping Rule)

If program T_i is infeasible or bounded with an optimal objective value strictly positive, STOP. The optimal objective value of Q is α^i.

Otherwise (program T_i is bounded, with optimal objective value 0), continue.

Step 3. (Progression)

Let $(\bar{\lambda}^i, \bar{u}^i, \bar{s}^i, \bar{x}^i, \bar{y}^i)$ be an optimal solution for T_i. Set $J^i = \{j \in J / \bar{s}^i_j > 0\}$.

Solve the problem $Q^i \equiv \max\{v^t x / x \in F(J^i)\}$.

If Q^i is unbounded, STOP. Problem Q is unbounded.

Otherwise, let $\hat{x}^{i+1}$ an optimal extreme point of Q^i and set $\alpha^{i+1} = v^t \hat{x}^{i+1}$.

Set $i = i+1$ and go to Step 1.

Before analyzing the properties of the algorithm just described, a few words are necessary to clarify some details of it.

To begin with, in order to compute an initial efficient solution in Step 0 we can use, among others, the method described in Ecker and Kouada (1975), which can be readily accomplished by using the well-known simplex method (see, for instance, Murty (1983)). However, the use of a good heuristic for computing an initial efficient solution close to the optimal solution of Q can save a lot of computational effort. In this sense, we propose the following heuristic procedure, which begins by solving the linear relaxation of problem Q, given by:

$$\max\{v^t x / x \in X\}$$

and then employs the obtained solution as the entry point in the Ecker–Kouada method. Although there is not guarantee on the quality of the solution generated by this approach, at least it allows dealing with the complete efficiency possibility of the underlying MOLP problem (see, for instance, Benson (1991b)) in a convenient way.

On the other hand, Step 1 assumes the responsibility of finding an efficient face that strictly improves the objective function value for problem Q obtained in the previous iteration. Clearly, this is the most computationally demanding task of the algorithm since it needs to solve the simplified disjoint bilinear program (BLP) T_i.

Problem BLP has been extensively studied in the literature for more than thirty years. See, for instance, Horst and Tuy (1996) for a survey. Although the resolution of T_i is far from being considered straightforward, it is possible to find several BLP solvers in some recent contributions (see, for instance, Audet et al. (1999) and Alarie et al. (2001)).

Finally, the implementation of both Step 2 and Step 3 has not difficulties. Particularly, note that Step 3 only requires the solution of a standard LP.

4.2. CONVERGENCE AND OTHER PROPERTIES OF THE ALGORITHM

The following results will show that the feasible extreme points $\hat{x}^i$ generated by the algorithm for problem Q will yield monotonically increasing objective function values. Furthermore, we will see that the algorithm only needs a finite number of iterations to find a global optimal solution or to conclude that Q is unbounded.

One of the more important features of our algorithm is that an improved feasible solution for problem Q is found in each iteration. Indeed:

PROPOSITION 4.1. *The algorithm yields monotonically increasing objective function values for problem Q.*

Proof. Let α^i the level imposed to the objective function $v^t x$ in the algorithm iteration i. By Theorem 3.1, if program T_i is bounded, with optimal objective

value 0, then $F(J^i) \in E_f^P$ and $\exists \hat{x} \in F(J^i)$ such that $v^t \hat{x} > \alpha$. Since in step 3 we compute α^{i+1} as the optimal objective value of problem $Q^i \equiv \max\{v^t x / x \in F(J^i)\}$, this implies $\alpha^{i+1} > \alpha^i$. □

The proposed algorithm only considers, for each encountered face, one descriptor set (not necessarily maximal) and discards from further consideration all those descriptor sets corresponding to faces contained in some other already obtained. Before showing this statement we need a preliminary result:

LEMMA 4.1. *Let* $J', J'' \subseteq J$ *such that* $F(J'') \subseteq F(J')$ *and assume that the program* $\max\{v^t x / x \in F(J')\}$ *is bounded, with optimal objective value* α. *Let* $\beta \geqslant \alpha$ *an arbitrary constant. Then, the system*:

$$\left.\begin{array}{l} Ax - by = 0 \\ v^t x - \beta y \geqslant 1 \\ x \geqq 0, x_{J''} = 0, y \geqslant 1 \end{array}\right\} \tag{6}$$

has no solution.

Proof. By hypothesis $F(J'') \subseteq F(J')$. Suppose on the contrary that the system (6) is feasible. Applying Proposition 3.1 we have $\exists \hat{x} \in F(J'') \subseteq F(J')$ such that $v^t \hat{x} > \beta \geqslant \alpha$, but this is untenable. □

Now, we are able to conclude the aforementioned result.

THEOREM 4.1. *If* J^i *is the descriptor set yielded by the algorithm in the iteration* i, *then none* $J' \subseteq J$, *such that* $F(J^i) \supseteq F(J')$ *will be considered in a later iteration.*

Proof. Let $J' \subseteq J$ such that $F(J^i) \supseteq F(J')$. Let α^i the objective level used by the bilinear problem T_i in the algorithm iteration i. By applying Lemma 4.1 we get that $\forall \beta \geqslant \alpha^i$, the system $\left.\begin{array}{l} Ax - by = 0 \\ v^t x - \beta y \geqslant 1 \\ x \geqq 0,\ x_{J'} = 0,\ y \geqslant 1 \end{array}\right\}$ has no solution. Since the previous system constitutes a convenient particularization of part of the restriction set of T_i, Propositions 3.1 and 4.1 assure that J' can not be generated in a later iteration j of the algorithm by solving program T_j. □

As a direct consequence, combining that P has a finite number of efficient faces with Theorem 4.1, it is straightforward to prove the convergence of the proposed algorithm:

COROLLARY 4.1. *The algorithm executes only a finite number of iterations.*

If we denote by $|E_{xp}^P|$ the total number of efficient extreme points of problem P, it is clear that, in the worst case, the number of iterations executed by the algorithm is $|E_{xp}^P| + 1$.

Finally, having in mind the previous properties, we can prove that the algorithm is valid.

COROLLARY 4.2. *If $E^P \neq \varnothing$, the algorithm finds an exact, globally optimal solution for problem Q or concludes that Q is unbounded.*

Proof. Straightforward since P has a finite number of efficient faces and by construction, in each iteration, the algorithm finds an efficient face that strictly improves the objective function value for problem Q obtained in the previous iteration or concludes that none exists with the required property (Theorem 3.1). □

4.3. AN EXAMPLE

Consider the following multiple objective linear programming problem P:

$$\left.\begin{array}{l}\text{VMAX } x\\ \text{s.t. } x \in X\end{array}\right\}$$

where X is the polyhedron given by $\left\{ x \in R^3_+ \Big/ \begin{pmatrix}1 & 1 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} x \leqq \begin{pmatrix}5\\ 3\\ 3\\ 1\end{pmatrix} \right\}$.

The feasible set X is shown in Figure 1. It is clear that E^P is the edge between $x^2 = (3,\ 2,\ 1)^t$ and $x^3 = (2,\ 3,\ 1)^t$. Furthermore, the set of all efficient extreme points, E^P_{xp}, is $\{x^2,\ x^3\}$.

As an illustration of our algorithm, we want to minimize the first objective function of problem P over E^P. This is equivalent to solve the problem Q given by $\max\{(-1,\ 0,\ 0)x / x \in E^P\}$. Notice that the optimal objective value of our original problem is the same that the obtained solving Q, but with reversed sign.

Graphically it is easy to see that, the maximum value of $-x_1$ over X equals 0. However, the optimal objective value of Q is -2, which is achieved at the vertex x^3.

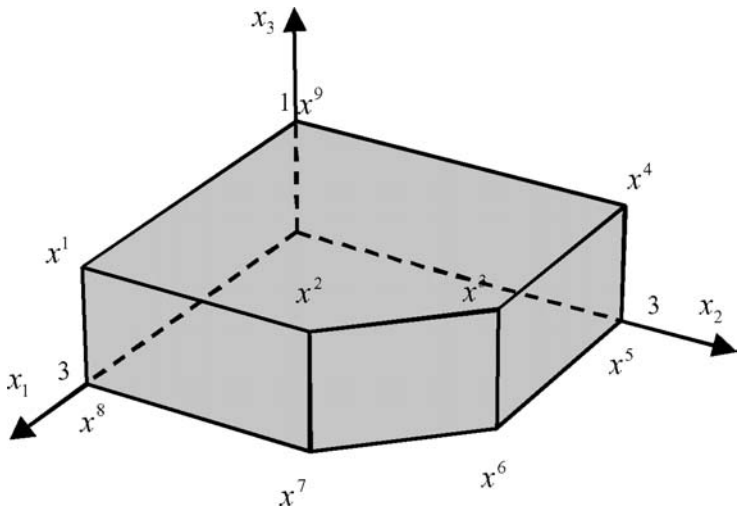

Figure 1. The feasible set X.

For algorithmic requirements we need a description of the feasible region X in standard form. This is accomplished by adding the slack variables x_4 to x_7. Thus, the coefficient matrices to be considered are:

$$A=\begin{pmatrix}1&1&0&1&0&0&0\\1&0&0&0&1&0&0\\0&1&0&0&0&1&0\\0&0&1&0&0&0&1\end{pmatrix},\ b=\begin{pmatrix}5\\3\\3\\1\end{pmatrix}\ \text{and}\ C=\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\end{pmatrix}.$$

One thing more before executing the algorithm. For doing the procedure geometrically more intuitive, all the solutions of problems Q^i will be given with their slack variable values omitted.

Step 0.

By using the Ecker–Kouada method we compute an initial efficient solution $\hat{x}^0$ given by $(3,\ 2,\ 1)^t$. Note that this points is the vertex x^2.

We set $i=0$ and $\alpha^0=-3$.

Iteration 0.

Step 1.

Solving the bilinear programming problem T_0, we obtain that it is bounded, with (globally) optimal objective value 0, being one of its optimal solutions the one given by $(\bar{\lambda}^0,\bar{u}^0,\bar{s}^0,\bar{x}^0,\bar{y}^0)$, where: $\bar{\lambda}^0=(1,\ 1,\ 1)^t, \bar{u}^0=(-1,\ 0,\ 0,\ -1)^t, \bar{s}^0=(0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1)^t, \bar{x}^0=(2,\ 3,\ 1,\ 0,\ 0,\ 1,\ 0)^t$ and $\bar{y}^0=1$.

Step 3.

Thus, $J^0=\{4,7\}$, which is the maximal descriptor set associated to the edge joining the vertices x^2 and x^3.

Now we have to solve problem $Q^0\equiv\max\{v^t x/x\in F(J^0)\}$.

We find that an optimal solution for Q^0 is $x^3=(2,\ 3,\ 1)^t$, with optimal objective value $\alpha^1=-2$.

We set $i=1$ and proceed to iteration 1.

Iteration 1.

Step 1.

We have to solve the bilinear program T_1. In this case, we obtain that T_1 is bounded but with (globally) optimal objective value equal to 1. In particular, one of its optimal solutions is the given by $(\bar{\lambda}^1,\bar{u}^1,\bar{s}^1,\bar{x}^1,\bar{y}^1)$, where: $\bar{\lambda}^1=(1,\ 1,\ 1)^t, \bar{u}^1=(-1,\ 0,\ 0,\ -1)^t, \bar{s}^1=(0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1)^t, \bar{x}^1=(1,\ 3,\ 1,\ 1,\ 2,\ 0,\ 0)^t$ and $\bar{y}^1=1$.

Step 2.

Since the optimal objective value of T_1 is strictly positive, we STOP. The extreme point $x^3=(2,\ 3,\ 1)^t$ is an optimal solution for problem Q, whose optimal objective value equals $\alpha^1=-2$.

5. Numerical Results

The algorithm described in Section 4 was coded in *C* (compiler gcc) and run on a PC Pentium at 450 Mhz with 256 Mb of RAM under Linux operating system.

Alarie et al. (2001)'s implementation of the algorithm *CBA* has been used to solve the disjoint bilinear programming problem. Additionally, the Cplex 6.0 library has been used to solve the linear programs.

In order to test the performance of our code, a total of 1080 problems, with a 25% nonzero density, were randomly generated according to the next scheme, similar to that followed by Sayin (2000) in his experimental tests. Elements of the matrix A, the vector b and the matrix C were randomly drawn from discrete uniform distributions in the range $[1, 30]$, $[50, 300]$ and $[-10, 10]$, respectively.

To avoid the infeasibility, all the constraints of each problem are of the '$\leqslant$' type. Additionally, to prevent unboundedness, the constraints of each problem were generated with all left hand side coefficients positive.

The vector v was generated in four different ways, describing four cases of problems. In Case 1, v is a dense vector randomly generated according to the same distribution as the objective coefficients of the underlying MOLP problem. In Case 2, v is a weighted combination of the rows of the objective coefficient matrix C, where the weights are randomly generated using a uniform distribution in the interval $[-1, 1]$ and at least one of them is negative. In Case 3, v is the first row of C reversed in sign. Case 4 is like Case 2, taking the weights from the interval $[-1, 0]$.

For each of these cases, 9 categories of problems were selected, according to the number of variables and constraints. The number of objectives k was taken as 3, 5 or 7. Finally, in each defined class we solved 10 test problems.

Computational results are summarized in Tables 5.1–5.3, which contain the average number of iterations and the average computing time (in seconds) for solving the instances. Also, in all the tables we report the minimum and the maximum of each measure in brackets.

An overall look at the results shown in Table 5.1 indicates that the algorithm presented compares favorably with the results obtained by Sayin (2000). However, it is worth noting that the hardware used and some characteristics of the test problems are different in both experiments, which only allows for cautious comparisons.

Obviously, problem difficulty is highly related to the size of the problem. In this way computational requirements of the algorithm increase significantly with the number of variables and restrictions, as can be inferred from Tables 5.2 and 5.3. Particularly, the results listed in Table 5.2 show that an increase in the number of variables of the problem requires more computational effort than increasing the number of restrictions, a difference that is rather common sense since the number of restrictions is in general inversely related to the size of the feasible region. As far as problem dimensions is concerned, Table 5.3 reports results about the performance of the algorithm when we increase simultaneously the number of variables and constraints. Note that in some categories the time needed for an exact solution is even smaller than the demanded by the heuristic proposed in Sayin (2000).

Table 5.1. CPU Times (seconds) and number of iterations for the initial test problems

	$C1$		$C2$		$C3$		$C4$	
$m \times n \times k$	Time	Niter	Time	Niter	Time	Niter	Time	Niter
$10 \times 10 \times 3$	0.368	1.6	0.261	1.3	0.17	1.1	0.456	1.6
	[0.09, 0.65]	[1, 3]	[0, 0.68]	[1, 2]	[0, 0.41]	[1, 2]	[0.25, 0.84]	[1, 3]
5	0.424	2.5	0.229	1.6	0.176	1.5	0.346	2.1
	[0, 0.74]	[1, 4]	[0, 0.58]	[1, 3]	[0.01, 0.5]	[1, 3]	[0.01, 0.67]	[1, 3]
7	0.21	1.4	0.139	1.4	0.167	2.2	0.361	2.5
	[0, 0.54]	[1, 2]	[0, 0.4]	[1, 3]	[0, 0.38]	[1, 5]	[0.01, 0.6]	[1, 4]
$15 \times 10 \times 3$	0.342	1.3	0.307	1.3	0.255	1.3	0.43	1.6
	[0, 0.97]	[1, 3]	[0, 2.27]	[1, 4]	[0, 0.74]	[1, 3]	[0.01, 0.95]	[1, 3]
5	0.575	1.9	0.273	1.1	0.267	1.4	0.932	2.9
	[0.01, 1.63]	[1, 3]	[0.01, 0.47]	[1, 2]	[0.01, 0.64]	[1, 2]	[0.3, 1.82]	[1, 5]
7	0.343	1.7	0.287	1.4	0.126	1.1	0.629	2.7
	[0, 0.87]	[1, 3]	[0, 1.11]	[1, 4]	[0, 0.41]	[1, 2]	[0.01, 1.19]	[1, 4]
$15 \times 12 \times 3$	0.766	1.8	0.536	1.7	0.495	1.5	1.029	2.2
	[0.01, 1.7]	[1, 3]	[0, 1.57]	[1, 3]	[0.13, 1.33]	[1, 3]	[0.41, 1.97]	[2, 3]
5	0.771	2.4	0.347	1.3	0.626	1.5	1.526	3
	[0.01, 1.69]	[1, 4]	[0, 0.74]	[1, 2]	[0, 2.8]	[1, 3]	[0.44, 5.53]	[1, 6]
7	0.771	2.4	0.59	1.6	0.152	1.4	1.177	4
	[0, 1.46]	[1, 4]	[0, 1.55]	[1, 3]	[0, 0.48]	[1, 3]	[0.3, 2.07]	[1, 7]

A remarkable point in our experiments is the fact that the average number of iterations always remains very low (under 8). This seems to be an intrinsic characteristic of the algorithm, rather than an accident; however, this point requires further study.

Table 5.2. The effect of increasing the number of constraints (CPU seconds and number of iterations)

	$C1$		$C2$		$C3$		$C4$	
$m \times n \times k$	Time	Niter	Time	Niter	Time	Niter	Time	Niter
$10 \times 15 \times 3$	0.548	2	0.153	1.4	0.326	1.8	0.586	2.6
	[0.21, 1.58]	[1, 4]	[0, 0.35]	[1, 2]	[0.11, 0.81]	[1, 5]	[0.32, 0.93]	[2, 5]
5	0.763	2.2	0.568	2.5	0.314	2	0.913	3.2
	[0.01, 2.61]	[1, 6]	[0, 1.48]	[1, 6]	[0, 0.92]	[1, 4]	[0.24, 1.96]	[1, 6]
7	0.325	1.7	0.214	1.4	0.283	1.8	0.815	2.8
	[0, 0.85]	[1, 3]	[0, 1.02]	[1, 4]	[0.01, 0.4]	[1, 4]	[0, 1.3]	[1, 4]
$12 \times 15 \times 3$	1.078	2.2	0.463	1.4	0.271	1.5	0.768	2.7
	[0.39, 2.6]	[1, 4]	[0, 1.24]	[1, 2]	[0, 0.85]	[1, 3]	[0, 1.35]	[1, 5]
5	0.964	2.7	0.479	2.4	0.234	1.2	0.731	2.9
	[0.25, 3.41]	[1, 7]	[0, 1.72]	[1, 6]	[0, 0.69]	[1, 2]	[0, 1.15]	[1, 6]
7	0.788	2.4	0.435	2	0.307	1.9	1.022	3.7
	[0, 1.64]	[1, 5]	[0, 1.54]	[1, 5]	[0, 1]	[1, 5]	[0.01, 2.7]	[1, 6]
$15 \times 15 \times 3$	1.076	1.7	0.771	2.1	1.294	2.1	1.822	2.9
	[0, 2.48]	[1, 3]	[0, 1.95]	[1, 4]	[0.33, 5.01]	[1, 5]	[0.48, 4.15]	[2, 6]
5	0.907	2.2	0.585	1.9	0.297	2.2	1.016	3
	[0, 1.9]	[1, 4]	[0, 1.66]	[1, 3]	[0, 0.87]	[1, 6]	[0.38, 2.57]	[1, 4]
7	1.155	2.1	0.212	1.3	0.263	1.4	1.315	3.1
	[0, 2.82]	[1, 3]	[0, 0.62]	[1, 2]	[0, 1.9]	[1, 4]	[0, 3.18]	[1, 5]

Table 5.3. The effect of increasing the number of variables and constraints (CPU seconds and number of iterations)

	$C1$		$C2$		$C3$		$C4$	
$m \times n \times k$	Time	Niter	Time	Niter	Time	Niter	Time	Niter
$20 \times 20 \times 3$	4.916	2.2	2.512	2.2	1.045	2.3	5.285	3.1
	[2.19, 8.67]	[1, 4]	[0, 8.56]	[1, 5]	[0.32, 2.11]	[1, 4]	[2.07, 10.33]	[2, 5]
5	5.397	2.5	0.789	1.8	0.809	2.7	8.733	4.5
	[1.14, 22.63]	[1, 5]	[0, 2.82]	[1, 5]	[0.23, 2.1]	[1, 5]	[1.6, 35.46]	[3, 9]
7	4.606	2.4	0.774	1.8	0.2133	3	4.213	5.9
	[0, 20.31]	[1, 8]	[0, 2.1]	[1, 4]	[0, 10.06]	[1, 8]	[0.98, 8.32]	[3, 8]
$25 \times 25 \times 3$	14.295	2.7	25.86	2.6	19.561	3.5	16.807	3.7
	[2.25, 47.73]	[1, 5]	[0.01, 216.88]	[1, 9]	[0.63, 124.51]	[1, 7]	[1.2, 39.41]	[1, 7]
5	13.547	4.5	4.064	2.8	19.32	3.3	17.367	7.1
	[2.27, 54.05]	[2, 8]	[0, 19.55]	[1, 8]	[0.41, 86.4]	[1, 7]	[1.4, 53.74]	[4, 10]
7	8.189	4.6	1.314	2.4	4.349	2.2	9.347	6.6
	[2.25, 13.83]	[1, 8]	[0, 3.51]	[1, 6]	[0.4, 13.56]	[1, 4]	[2.7, 23.1]	[1, 14]
$30 \times 30 \times 3$	63.288	3.6	14.725	2.8	39.908	3.3	49.179	4.5
	[6.66, 216.43]	[1, 7]	[0, 64.38]	[1, 7]	[1.78, 350.92]	[1, 6]	[6.08, 234.26]	[2, 9]
5	23.992	3	2.976	1.9	5.234	2.5	30.432	5.2
	[0.01, 148.15]	[1, 5]	[0, 7.58]	[1, 5]	[0.01, 16.42]	[1, 7]	[2.36, 89.82]	[3, 8]
7	50.046	6.2	9.21	2.7	16.646	4	60.03	7.7
	[4.92, 171.27]	[2, 13]	[0, 57.55]	[1, 8]	[0, 49.62]	[1, 8]	[5.73, 259.39]	[3, 15]

Many refinements of the basic algorithm here presented are possible to improve the performance of our implementation. These include, among others:

(i) Using better heuristics for computing an initial efficient solution in Step 0 of the algorithm (thus, our algorithm can be combined in a nice way with the heuristic proposed in Sayin (2000)).

(ii) Improving at least three important aspects concerning the implementation of the *CBA* algorithm in order to completely fit this particular context: (a) Extending the entry format for managing different types of variables (not only nonnegatives) and the constraints (not only of the '$\leqslant$' sort). Although it is easy to adapt an arbitrary formulation to any entry format, sometimes the number of variables and restrictions increase considerably during the translation. (b) Allowing a hot restarting from known feasible solutions for the bilinear problem. Note that our algorithm solves in each iteration a problem very similar to the problem solved in the last iteration (only a single coefficient of one constraint of the BLP problem changes) and it is easy, e.g. by the *mountain climbing* procedure of Konno (see, for instance, Horst and Tuy (1996)), to find a good feasible solution for the BLP from the previous one. (c) Since all the bilinear problems solved, except the last one have an optimal objective value of 0, the idea of establishing a cut that exploits this characteristic probably would save a great deal of computational effort.

6. Conclusions

The problem Q of optimizing a linear function over the efficient set of a MOLP has an inherently difficult nature due to the characteristics that describe it as a global optimization problem.

In this paper we have presented an algorithm that finds an exact globally optimal solution for problem Q, after a finite number of iterations and without making any assumptions on the boundedness of the efficient set E^P.

The algorithm generates a sequence of distinct efficient faces over which the objective function of problem Q is optimized. From a mathematical point of view, our procedure needs to solve, in each iteration, a bilinear program (step 1) and a scalar linear programming problem (step 3).

Mainly, the computational power of the algorithm comes from the following two features: first, the efficient faces found need not be adjacent or form a connected set and second, an improved feasible solution for problem Q is obtained in each iteration.

Although the worst case is exponential, our preliminary numerical results are very encouraging. In its present state, the algorithm performs fairly well (very low number of iterations and reasonable computing times) with problems up to 30 variables and 30 constraints.

Moreover, further improvements of performance can be yet achieved. In fact, the area of bilinear programming is constantly evolving, and new advances in this field will yield important computational savings in our algorithm since Step 1 appears as its main bottleneck. Additionally, the use of good heuristics for computing an initial efficient solution such as that of Sayin (2000) in Step 0 can significantly reduce the number of iterations required for our algorithm when it is involved with larger problems.

For all these reasons the procedure suggested throughout this work is expected to represent a useful and practical tool in the field of multiple objective linear programming.

Acknowledgements

The author is indebted to Dr. Charles Audet for kindly providing the source code of the *CBA* algorithm and to Dr. Marianela Carrillo for her ongoing support.

References

Alarie, S., Audet, C., Jaumard, B. and Savard, G. (2001), Concavity cuts for disjoint bilinear programming, *Mathematical Programming*, 90, 373–398.

Audet, C., Hansen, P., Jaumard, B. and Savard, G. (1999), A symmetrical linear maxmin approach to disjoint bilinear programming, *Mathematical Programming*, 85, 573–592.

Benson, H. (1984), Optimization over the efficient set, *Journal of Mathematical Analysis and Applications*, 98, 562–580.

Benson, H. (1991a), An all-linear programming relaxation algorithm for optimizing over the efficient set, *Journal of Global Optimization*, 1, 83–104.
Benson, H. (1991b), Complete efficiency and the initialization of algorithms for multiple objective programming, *Operations Research Letters*, 10, 481–487.
Benson, H. (1992), A finite, nonadjacent extreme-point search algorithm for optimization over the efficient set, *Journal of Optimization Theory and Applications*, 73(1), 47–64.
Benson, H. (1993), A bisection-extreme point search algorithm for optimizing over the efficient set in the linear dependence case, *Journal of Global Optimization*, 3, 95–111.
Benson, H. and Lee, D. (1996), Outcome-based algorithm for optimizing over the efficient set of a bicriteria linear programming problem, *Journal of Optimization Theory and Applications*, 88(1), 77–105.
Benson, H. and Sayin, S. (1993), A face search heuristic algorithm for optimizing over the efficient set, *Naval Research Logistics*, 40, 103–116.
Benson, H. and Sayin, S. (1994), Optimization over the efficient set: four special cases, *Journal of Optimization Theory and Applications*, 80(1), 3–18.
Ecker, J. and Kouada, I. (1975), Finding efficient points for linear multiple objective programs, *Mathematical Programming*, 8, 375–377.
Ecker, J. and Song, J. (1994), Optimizing a linear function over an efficient set, *Journal of Optimization Theory and Applications*, 83(3), 541–563.
Evans, J. and Steuer, R. (1973), A revised simplex method for linear multiple objective programs, *Mathematical Programming*, 5, 54–72.
Horst, R. and Tuy, H. (1996), *Global Optimization (Determinsistic Approaches)*, 3rd Edition, Springer.
Isermann, H. and Steuer, R. (1987), Computational experience concerning payoff tables and minimun criterion values over the efficient set, *European Journal of Operational Research*, 33, 91–97.
Jorge, J. (2003), Maximal descriptor set characterizations of efficient faces in multiple objective linear programming, *Operations Research Letters*, 31, 124–128.
Mangasarian, O. (1969), *Nonlinear Programming*, McGraw-Hill.
Murty, K. (1983), *Linear Programming*, John Wiley & Sons.
Murty, K. (1985), Faces of a polyhedron, *Mathematical Programming Study*, 24, 30–42.
Philip, J. (1972), Algorithms for the vector maximization problem, *Mathematical Programming*, 2, 207–229.
Rockafellar, R. (1970), *Convex Analysis*, Princeton University Press.
Sayin, S. (2000), Optimizing over the efficient set using a top-down search of faces, *Operations Research*, 48, 65–72.
Steuer, R. (1986), *Multiple Criteria Optimization: Theory, Computation and Application*, John Wiley & Sons.
Yu, P. and Zeleny, M. (1975), The set of all nondominated solutions in linear cases and a multicriteria simplex method, *Journal of Mathematical Analysis and Applications*, 49, 430–468.