ELSEVIER

Operations Research Letters 19 (1996) 87–94

operations research letters

An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming

Serpil Sayın

Bilkent University, Management Department, Faculty of Business Administration, 06533 Bilkent, Ankara, Turkey

Received 1 January 1995; revised 1 July 1995

Abstract

We propose a method for finding the efficient set of a multiple objective linear program based on the well-known facial decomposition of the efficient set. The method incorporates a simple linear programming test that identifies efficient faces while employing a top–down search strategy which avoids enumeration of efficient extreme points and locates the maximally efficient faces of the feasible region. We suggest that discrete representations of the efficient faces could be obtained and presented to the Decision Maker. Results of computational experiments are reported.

Keywords: Multiple objective linear programming; Vector maximization; Efficient faces; Efficient set

1. Introduction

Given a nonempty polyhedral set X defined as $X = \{x \in \mathbb{R}^n \mid Ax \leqq b,\ x \geqq 0\}$, where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$, and a $p \times n$ matrix C of objective function coefficients where $p \geqq 2$, we can define the Multiple Objective Linear Programming problem (MOLP) as follows:

(MOLP) Maximize Cx, subject to. $x \in X$.

Throughout the paper we employ the following notation: For two scalars a and b, $a \leqq b$ denotes $a < b$ or $a = b$. For two vectors $x, y \in \mathbb{R}^n$, $x \leqq y$ denotes $x_i \leqq y_i$ for $i = 1 \ldots n$, and $x \leqslant y$ denotes $x \leqq y$ but $x \neq y$.

Incorporation of p objective functions simultaneously into the linear programming framework necessitates the consideration of *efficient solutions* for problem (MOLP).

Definition 1. $x^o \in \mathbb{R}^n$ is an *efficient solution* for problem (MOLP) if $x^o \in X$ and there exists no $x \in X$ such that $Cx \geqslant Cx^o$.

Let X_E denote the set of all efficient solutions for problem (MOLP). The vector maximization approach to problem (MOLP) is based on the assumption that the Decision Maker (DM) prefers more to less in each objective. Therefore X_E contains all the relevant trade-off information to be conveyed to the DM. Thus vector maximization algorithms aspire to find all of X_E and present it to the DM. (For an overview of alternative approaches to problem (MOLP), the reader is referred to, for instance, [7, 10, 14] and references therein.)

It has been shown that the efficient set for problem (MOLP) can be represented as a union of efficient faces of X [15]. However, finding all of X_E is a computationally-demanding problem. Furthermore, presenting X_E to the DM is a difficult task since it is usually a nonconvex continuous set. Thus most

SSDI 0167-6377(95)00046-1

vector maximization methods have directed their effort to finding the set of all efficient extreme points of X (see, for instance, [9, 11]).

Typically, most of the vector maximization algorithms suggested for problem (MOLP) follow a "bottom–up search". That is, efficient faces are generated based on the information provided by efficient extreme points by incorporating certain tests (see, for instance, [8, 12]). Recently, Armand and Malivert [2] and Armand [1] presented such algorithms and reported computational results. A partial exception to the bottom–up search approach is the method of Yu and Zeleny [15] where a "top–down" search strategy is incorporated into the procedure without giving up the bottom-up search.

Since efficient extreme points are zero-dimensional faces, their number could be expected to be much more than the number of maximally efficient faces due to the combinatorial nature of the problem assuming that maximal efficient faces are of high dimensionality (see Remark 3 in Section 2). It has also been discussed that X may "collapse" under the mapping C, and the structure of the efficient set in outcome space may be much simpler than in decision space. Recently, Dauer and Gallagher [6] have shown that maximally efficient faces of X and maximally efficient faces in the outcome space are in one-to-one correspondence, supporting the hypothesis that the structure of X_E when considered as a union of maximally efficient faces directly could be relatively simple.

Developing a method for finding maximally efficient faces of X by employing a top–down search, that is, by avoiding the generation of lower dimensional efficient faces including efficient extreme points may be a viable idea. Discrete representations of the individual faces could then be obtained and presented to the DM [4].

In this paper, we propose an algorithm for locating X_E for problem (MOLP). We base our method mostly on the results presented in [15]. Our algorithm differs from the method presented in [15] mainly in its disregarding of the efficient extreme points of X, and in the test employed to detect efficient faces. We use the characterization of faces as a collection of indices that correspond to the constraints holding as equality at that particular face [15]. Thus the problem of searching for maximal efficient faces can be reduced to the problem of searching for collection of indices that correspond to maximal efficient faces. Therefore a search is performed over the space of collection of indices by employing a simple test that identifies efficient faces. Since the search procedure considers faces of possibly higher dimension prior to those of lower dimension, the obtained solution consists of maximally efficient faces of X.

The purpose of this paper is twofold. First, we want to formulate a method that is easy to understand and implement to find X_E. By means of the facial decomposition and the top–down search approach, we aspire to synchronize the vector maximization problem with problems of discrete nature that arise in various other contexts for which search techniques are utilized. Second, we want to gather empirical information about the structure of X_E and observe its sensitivity to certain factors such as the number of objectives, the number of variables and the number of constraints. The paper is organized as follows. In Section 2, we establish the theoretical background and state our results. Section 3 contains the algorithm. In Section 4, computational results are presented.

2. Problem definition and theoretical background

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 [13]. A face F is an *efficient face* if all the elements of F are efficient. A face F is a *maximally efficient face* if F is an efficient face and there exists no efficient face of X that contains F as a strict subset. Let

$$\bar{A} = \begin{bmatrix} A \\ -I_n \end{bmatrix} \quad \text{and} \quad \bar{b} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

where I_n is the $n \times n$ identity matrix and $0 \in \mathbb{R}^n$. Note that we can now rewrite $X = \{x \in \mathbb{R}^n \mid \bar{A}x \leqq \bar{b}\}$. Let $M = \{1, \ldots, m+n\}$ and $\mathcal{M} = \{I \mid I \subseteq M\}$. Define $\bar{A}^I$ as the matrix derived from $\bar{A}$ by deleting rows of $\bar{A}$ not in I. Define $\bar{b}^I$ as the vector derived from $\bar{b}$ by deleting elements of $\bar{b}$ not in I. For $I \in \mathcal{M}$, define $F(I) = \{x \in X \mid \bar{A}^I x = \bar{b}^I\}$. For $I \in \mathcal{M}$, $F(I)$ represents a face of X [15]. Note that, $F(\emptyset) = X$ and for $I \in \mathcal{M}$, $F(I) = \emptyset$ is possible. We will refer to $F(I)$ as a *proper face* of X if $F(I) \neq \emptyset$.

The following observations about the characterization, some of which are given in [15], will be useful in understanding the algorithm.

Remark 1. Given a face of X, its representation by $F(I)$ may not be unique. In other words, for I, $J \in \mathcal{M}$ such that $I \neq J$, it is possible to have $F(I) = F(J)$.

Remark 2. Let $|I|$ denote the number of elements of I. If $|I| \leqq |J|$, then $\dim(F(I)) \geqq \dim(F(J))$. Indeed, in the absence of degeneracy, for $|I| > n$, $F(I) = \emptyset$ and for $|I| \leqq n$, $\dim(F(I)) = n - |I|$.

Remark 3. The number of faces of X of dimension k is bounded from above by the quantity

$$\binom{n+m}{n-k} = \frac{(n+m)!}{(n-k)!(m+k)!}$$

Remark 4. If $I \subseteq J$, then $F(J) \subseteq F(I)$.

The following results will help us identify those $F(I)$ that are efficient. First, for $I \in \mathcal{M}$, define the problem (SP_I)

$$(SP_I)\quad v_I = \max : e^{\mathrm{T}}Cx - e^{\mathrm{T}}Cy, \tag{1}$$

$$\text{s.t. } \bar{A}x \leqq \bar{b}, \tag{2}$$

$$\bar{A}y \leqq \bar{b}, \tag{3}$$

$$\bar{A}^I y = \bar{b}^I, \tag{4}$$

$$-Cx + Cy \leqq 0, \tag{5}$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$.

Proposition 1. *Let $I \in \mathcal{M}$. $F(I)$ is a proper face of X if and only if problem (SP_I) has a feasible solution.*

Proof. (i) Assume that $F(I)$ is a proper face of X. Then there exists a point $y \in F(I)$. By definition of $F(I)$, y satisfies $\bar{A}y \leqq \bar{b}$ and $\bar{A}^I y = \bar{b}^I$. Let $x = y$. Then (x, y) constitutes a feasible solution for problem (SP_I).

(ii) Assume that problem (SP_I) has a feasible solution (x, y). Then, by (3) and (4), $y \in F(I)$. Thus $F(I)$ is a proper face. □

Theorem 1. *Let $I \in \mathcal{M}$. $F(I)$ is a proper efficient face of X if and only if the optimal objective function value v_I exists and is equal to 0.*

Proof. (i) Suppose that $F(I)$ is a proper efficient face of X. Assume, to the contrary, that the optimal objective value v_I for problem (SP_I) does not exist or is not equal to 0.

Case 1: v_I does not exist. Since $F(I)$ is a proper face, problem (SP_I) has a feasible solution by Proposition 1. Thus v_I does not exist implies that problem (SP_I) is unbounded. This implies that there exists $y \in F(I)$ (by (3) and (4)) and $x \in X$ (by (2)) such that $Cx \geqq Cy$ (by (5)) and $e^{\mathrm{T}}Cx - e^{\mathrm{T}}Cy > 0$ (by (1)). This implies that $Cx \geqslant Cy$. Thus $y \in F(I)$ is not efficient, which contradicts the supposition that $F(I)$ is an efficient face.

Case 2: $v_I \neq 0$. Observe that $v_I > 0$ should hold since for any $y \in F(I)$, (x, y) with $x = y$ is a feasible solution for problem (SP_I) and results in an objective value of 0. Then, by an argument similar to that in Case 1, there exists $y \in F(I)$ and $x \in X$ such that $Cx \geqslant Cy$, which contradicts the supposition that $F(I)$ is an efficient face.

(ii) Suppose that v_I exists and is equal to 0. Then there exists no $x, y \in \mathbb{R}^n$ that satisfies (2)–(5) with $e^{\mathrm{T}}Cx > e^{\mathrm{T}}Cy$. Then for all $x \in X$ and $y \in F(I)$ such that $Cx \geqq Cy$,

$$Cx = Cy \tag{6}$$

holds. Since $F(I) \neq \emptyset$, we can pick any $y \in F(I)$. Then by (6), there exists no $x \in X$ such that $Cx \geqslant Cy$. Hence $y \in X_{\mathrm{E}}$ follows. Since $y \in F(I)$ was chosen arbitrarily, it follows that $F(I)$ is an efficient face of X. □

Let $\mathcal{M}^k = \{I \in \mathcal{M} \mid |I| = k\}$ for $k = 0, 1, \ldots, m+n$. Since $\mathcal{M} = \bigcup_{k=0}^{m+n} \mathcal{M}^k$, it follows that (cf. Theorem 4.1 in [15])

$$X_{\mathrm{E}} = \bigcup_{k=0}^{m+n} \bigcup_{I \in \mathcal{M}^k} X_{\mathrm{E}} \cap F(I). \tag{7}$$

Since $X_{\mathrm{E}} \cap F(I) = F(J)$ for some $J \supseteq I$ ([15]), there exists $\mathcal{E}^0 \subseteq \mathcal{M}^0, \mathcal{E}^1 \subseteq \mathcal{M}^1, \ldots \mathcal{E}^{m+n} \subseteq \mathcal{M}^{m+n}$ such that

$$X_{\mathrm{E}} = \bigcup_{k=0}^{m+n} \bigcup_{I \in \mathcal{E}^k} F(I). \tag{8}$$

Note that, by Remark 1, $\mathcal{E}^k$, $k = 0, \ldots, m+n$, need not be unique. However, based on (8), it is possible

to formulate an algorithm that identifies X_E by identifying $\mathscr{E}^k, k = 0, \ldots, m+n$.

3. The algorithm

The proposed algorithm for finding X_E is based on checking elements of $\mathscr{M}^k$ starting with $k = 0$. For $k = 0$, the only element of $\mathscr{M}^k$ is $\emptyset$ and $F(\emptyset) = X$. Thus, by solving problem $(SP_\emptyset)$, the algorithm first checks if problem (MOLP) is completely efficient or not [3]. If problem (MOLP) is completely efficient, the algorithm terminates with the conclusion $X_E = X$. If not, then for each element I of $\mathscr{M}^1$, problem (SP_I) is solved. If problem (SP_I) is infeasible, I is dropped from further consideration since $F(I) = \emptyset$ (by Proposition 1), and is placed in a list that keeps index sets yielding infeasible combinations. If problem (SP_I) has an optimal objective value of 0, I is dropped from further consideration since $F(I)$ is efficient (by Theorem 1). In this case, I is placed in a list that keeps index sets yielding efficient faces. If problem (SP_I) is unbounded or has a positive optimal objective value, we can conclude that $F(I)$ has at least one element that is not efficient. Therefore, it is possible to have $F(J) \subset F(I)$ efficient and thus $J \supset I$ should be checked via solving problem (SP_J). Hence, immediate supersets of I, i.e., index sets that contain I and belong to $\mathscr{M}^{|I|+1}$, are placed in a list for later consideration. After all the elements of $\mathscr{M}^1$ are considered, the index sets that were placed in the list are considered in the order they were placed in the list. For an index set I, first it is checked if it is a superset of an index set previously placed in the list of infeasible index sets. If so, I is also an index set that yields an infeasible combination by Remark 4 and therefore is discarded. Next it is checked if I is a superset of an index set previously placed in the list of index sets that yield an efficient face. If so, I is also an index set yielding an efficient face that is a subset of a previously identified efficient face by Remark 4 and therefore is discarded. If these tests fail, then problem (SP_I) is solved and I is placed in an appropriate list based on the result obtained from the solution of the linear program as described above. The procedure stops when there are no more index sets in $\mathscr{M}$ to be processed. A formal description of the algorithm is given below.

3.1. The algorithm statement

Step 0 0.0 Set $I = \emptyset$. Solve problem (SP_I). If $v_I = 0$, then $X = X_E$. Set $\mathscr{E}\mathscr{L} = \{\emptyset\}$ and STOP. Else, go to step 0.1.

0.1 Set $k = 1, \mathscr{E}\mathscr{L} = \emptyset, \mathscr{I}\mathscr{L} = \emptyset, \mathscr{L} = \mathscr{M}^k$, and go to Step 1.

Step 1 1.0 If $\mathscr{L} \cap \mathscr{M}^k = \emptyset$, set $k = k+1$. If $k = m+n+1$ or $\mathscr{L} \cap \mathscr{M}^k = \emptyset$, STOP. Else, pick $I \in \mathscr{L} \cap \mathscr{M}^k$.

1.1 If $I \supset J$ for some $J \in \mathscr{I}\mathscr{L}$ or $I \supset J$ for some $J \in \mathscr{E}\mathscr{L}$, set $\mathscr{L} = \mathscr{L} \setminus \{I\}$ and go to 1.0. Else solve problem (SP_I).

1.2 If problem (SP_I) is infeasible, set $\mathscr{I}\mathscr{L} = \mathscr{I}\mathscr{L} \cup \{I\}$, set $\mathscr{L} = \mathscr{L} \setminus \{I\}$. Go to 1.0.

1.3 If $v_I = 0$, set $\mathscr{E}\mathscr{L} = \mathscr{E}\mathscr{L} \cup \{I\}$, set $\mathscr{L} = \mathscr{L} \setminus \{I\}$. Go to 1.0.

1.4 Set $\mathscr{L} = \mathscr{L} \setminus \{I\} \cup \mathscr{I}^1$, where $\mathscr{I}^1$ denotes supersets of I that belong to $\mathscr{M}^{k+1}$. Go to 1.0.

3.2. Validation

To validate the algorithm, we need to show that

$$X_E = \bigcup_{I \in \mathscr{E}\mathscr{L}} F(I).$$

By construction, $\mathscr{E}\mathscr{L}$ contains index sets I such that problem (SP_I) has an optimal objective value of 0. Thus $X_E \supseteq \bigcup_{I \in \mathscr{E}\mathscr{L}} F(I)$ is obvious by Theorem 1.

To see $X_E \subseteq \bigcup_{I \in \mathscr{E}\mathscr{L}} F(I)$, it is necessary to observe that all elements of $\mathscr{M}$ are inspected by the algorithm explicitly or implicitly. Indeed, a search procedure is conducted over $\mathscr{M}$ by considering elements I in the order of nondecreasing cardinality (Step 1.0) to identify those elements that correspond to efficient faces of X. Note that $\mathscr{L}$ is initialized to $\mathscr{M}^1$, and whenever an element I is removed from $\mathscr{L}$, its immediate supersets are added to $\mathscr{L}$ (Step 1.4) unless it is guaranteed that $F(I) = \emptyset$ (Step 1.2) or $F(I) \subseteq X_E$ (Step 1.3). Thus, by (7) and Step 1.3, it follows that $X_E \subseteq \bigcup_{I \in \mathscr{E}\mathscr{L}} F(I)$.

3.3. Implementation issues

A crucial part of the algorithm is keeping various lists of sets of discrete elements. To increase the efficiency of the list-keeping procedures, it is possible to consider $I \in \mathcal{M}$ as an $(m+n)$-tuple $\alpha^I = (\alpha_1^I, \ldots, \alpha_{(m+n)}^I)$ such that $\alpha_i^I = 1$ if $i \in I$, and 0 otherwise. Thus it becomes possible to avoid duplicate evaluations of index sets by following a lexicographic order of α^I's in the processing of the list $\mathcal{L}$ and in generation of $\mathcal{I}^1$ in Step 1.4.

In addition to list-keeping schemes, the only requirement to implement the algorithm is a tool to solve the linear programming problems. This requirement can be met, for instance, by utilizing the simplex method, which is simple and easy to implement. Furthermore, the following observations can be made regarding the use of simplex method in the algorithm.

(1) When solving problem (SP_I) for $I \in \mathcal{M}$, if the objective value is detected to be positive at any iteration, there is no need to solve the problem to optimality since it is guaranteed that $v_I > 0$ or problem (SP_I) is unbounded, and $F(I)$ is not an efficient face.

(2) For $J \in \mathcal{I}^1$, problem (SP_J) differs from problem (SP_I) in only one constraint. Therefore, the solution to problem (SP_I) can be used as a starting solution for problem (SP_J). This could possibly improve computational performance.

It can be noted that the proposed algorithm does not require an explicit treatment of degeneracy and unbounded feasible region since it does not concentrate on efficient extreme points or efficient bases. Thus by avoiding complex book-keeping schemes, it remains simple and easy to implement.

3.4. An example

Consider the problem (MOLP) with

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 12 \\ 8 \end{bmatrix}.$$

The efficient set is the union of the two maximally efficient faces that correspond to $I_1 = \{1\}$ and $I_2 = \{2\}$ (see Fig. 1). Here, $M = \{1, \ldots, 5\}$. The algorithm proceeds as follows.

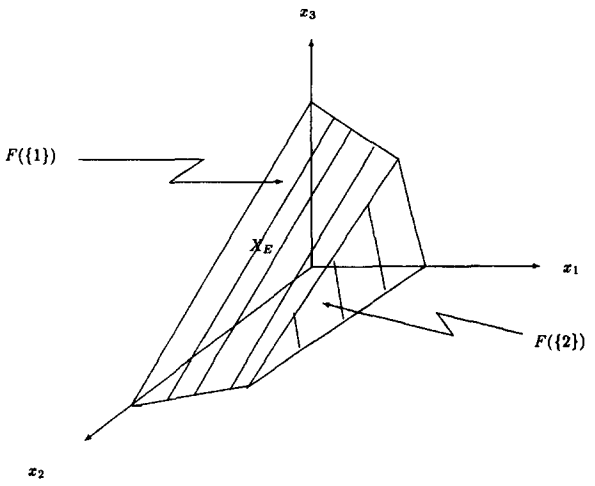

Fig. 1. Example: X and X_E.

0.0 Solve problem (SP_I) with $I = \emptyset$. $v_I > 0$.
0.1 $k = 1, \mathcal{EL} = \emptyset, \mathcal{IL} = \emptyset, \mathcal{L} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
1.1 Solve problem (SP_I) with $I = \{1\}$.
1.3 $v_I = 0$. $\mathcal{EL} = \{\{1\}\}, \mathcal{L} = \{\{2\}, \{3\}, \{4\}, \{5\}\}$.
1.1 Solve problem (SP_I) with $I = \{2\}$.
1.3 $v_I = 0$. $\mathcal{EL} = \{\{1\}, \{2\}\}, \mathcal{L} = \{\{3\}, \{4\}, \{5\}\}$.
1.1 Solve problem (SP_I) with $I = \{3\}$.
1.4 $\mathcal{L} = \{\{4\}, \{5\}, \{3,4\}, \{3,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{4\}$.
1.4 $\mathcal{L} = \{\{5\}, \{3,4\}, \{3,5\}, \{4,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{5\}$.
1.4 $\mathcal{L} = \{\{3,4\}, \{3,5\}, \{4,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{3,4\}$, ($k = 2$ in Step 1.0).
1.4 $\mathcal{L} = \{\{3,5\}, \{4,5\}, \{3,4,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{3,5\}$.
1.4 $\mathcal{L} = \{\{4,5\}, \{3,4,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{4,5\}$.
1.4 $\mathcal{L} = \{\{3,4,5\}\}$.
1.1 Solve problem (SP_I) with $I = \{3,4,5\}$, ($k = 3$ in Step 1.0).
1.4 $\mathcal{L} = \emptyset$.
1.0 $k = 4$, $\mathcal{L} \cap \mathcal{M}^k = \emptyset$. STOP.

4. Computational results

In this section we present our computational experimentation with the algorithm. The algorithm was coded in C, and the CPLEX Callable Library [5]

Table 1
Computational results:CPU and primary storage requirements

	CPU time (seconds)			Total LP pivots			Working list size			Infeasible list size		
$p \times m \times n$	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
$03 \times 10 \times 03$[a]	0.2	0.3	0.3	26	71.6	112	8	14.4	22	7	8.4	10
$03 \times 10 \times 03$	0.2	0.3	0.4	50	120.2	231	12	21.6	28	8	11.1	15
$03 \times 10 \times 10$	203.1	559	1247.7	37803	151825.3	452446	17200	38399.4	77968	30	95.5	203
$03 \times 15 \times 10$	237.4	1143.3	2105.9	34187	264859.8	500728	18650	61733.2	105580	64	197.9	399
$05 \times 10 \times 05$[a]	0.5	1.1	1.9	97	417.8	925	48	92.4	156	6	10.2	19
$05 \times 10 \times 05$	1.2	2.4	3.7	373	1150.6	2030	96	179.4	292	8	15.9	22
$05 \times 10 \times 10$	201.9	634	1500	35757	202883.8	661551	17112	37960.2	76762	29	95.4	203
$05 \times 15 \times 10$	284.5	1224.1	2252.9	65231	332334.8	598421	18268	59108.6	105434	63	191.1	399
$07 \times 10 \times 07$[a]	3	7.5	12.7	595	2244.7	4071	304	636.2	936	11	15.4	22
$07 \times 10 \times 07$	9.4	19.7	29.3	3216	7876.8	15182	604	1293.8	1772	12	25.7	38
$07 \times 10 \times 10$	212.2	700.6	1532.8	47454	245629.4	580017	14906	36277	76572	25	91.9	202
$07 \times 15 \times 10$	322.8	1092.3	2327.2	103954	328146.6	617840	17544	49952	105304	57	162.9	399

[a] Problems that are solved for admissible points.

was used to solve the linear programming problems (SP_I). The computational experiments were conducted on a multi-user SunSparc workstation. In our computational experiments, the class of a problem is determined by the number of objectives (p), number of constraints (m) and the number of variables (n). For each problem class, we created a set of 10 test problems. In the test problems, the elements of the constraint matrix A, the right-hand-side vector b, and the objective function coefficient matrix C were randomly generated integers belonging to the discrete uniform distribution in the intervals $(1,20)$, $(50,100)$ and $(-10,20)$, respectively. A 25% zero density was provided in the matrix A. Nine different classes were constructed according to the values of p, m and n. Three additional problem classes were created by solving the test problems in three categories with $p = n$ for admissible points (i.e., with $C = I_p$, where I_p is the $p \times p$ identity matrix).

The results of computational experiments are reported in two separate tables. The first table contains information regarding the computational performance of the algorithm. Along with CPU times, we report total number of LP-pivots as an indicator of computation time. As an indicator of storage requirements, the size of the working list ($\mathscr{L}$) and the size of the list of infeasible I's ($\mathscr{IL}$) are reported. In Table 1, it can be observed that computational requirements increase rapidly with problem size. It can also be observed that number of variables seems to have the most significant effect on computation time where the number of constraints also seems to be important. Based on these preliminary results, number of objectives does not seem to be a very important factor. Another observation is that solving for admissible points is likely to be easier than solving for efficient points, especially as the problems get larger.

Table 2 contains the information pertaining to the structure of the test problems solved. Basically, it would be interesting to know the number of distinct efficient faces and the dimensions of each for each problem. As one pseudo-measure, we report the efficient list size ($\mathscr{EL}$), which is an upper bound on the number of distinct efficient faces (see Remark 1). To have an idea about the dimension of the highest dimensional efficient face, we report $|I|_{\min} = \min_{I \in \mathscr{EL}} |I|$, since $n - |I|_{\min}$ is a lower bound on this quantity. Note that, in the absence of degeneracy, these are exact values rather than bounds. Finally, as an approximate measure in the variation in the dimensions of the efficient faces, we report $|I|_{\max} - |I|_{\min}$. The results in Table 2 indicate that the efficient set has a relatively simple structure. The efficient list size, though increasing with problem size, seems to be rather reasonable when the combinatorial nature of the total number of faces is considered (Remark 3). That seems to be more in effect for the set of admissible points. As far as the dimensions of the efficient faces are concerned, an

Table 2
Computational results: Structure of X_E.

	Efficient list size			$\|I\|_{min}$			$\|I\|_{max} - \|I\|_{min}$		
$p \times m \times n$	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
03 × 10 × 03[a]	1	2.2	4	1	1	1	0	0.2	1
03 × 10 × 03	1	1.8	3	1	1.6	3	0	0.3	1
03 × 10 × 10	4	11.4	27	8	8	8	0	0.7	1
03 × 15 × 10	4	15.5	45	8	8	8	0	0.6	1
05 × 10 × 05[a]	2	3.5	8	1	1	1	0	0.3	1
05 × 10 × 05	1	4.4	9	1	1.9	3	0	0.9	2
05 × 10 × 10	3	25.1	49	6	6.4	8	1	1.8	3
05 × 15 × 10	12	37.3	92	6	6.3	7	1	1.7	2
07 × 10 × 07[a]	3	6.4	10	1	1.1	2	0	1	2
07 × 10 × 07	4	12.5	21	2	2.6	4	2	2.2	3
07 × 10 × 10	16	43.6	57	4	4.8	6	2	2.8	3
07 × 15 × 10	16	48.1	93	4	4.7	6	2	3.1	4

[a] Problems that are solved for admissible points.

Table 3
CPU times (in seconds) for Armand–Malivert test problems

	Tube				Pyramid				Tent			
m	20	30	40	50	20	30	40	50	21	31	41	51
	0.6	0.9	1.5	2.0	0.6	0.9	1.4	1.8	0.9	1.2	1.8	2.6

important observation is that the variation across the efficient set is rather small. Furthermore, if we take $n - |I|_{min}$ as an indicator of the dimension of the highest dimensional efficient face, for a given value of p, this quantity shows little variation for different values of m and n. However, more experimentation would be necessary to draw strong conclusions.

In addition to the computational experiments conducted, the 12 problems presented by Armand and Malivert [2] were solved for comparison purposes since theirs is the only algorithm with reported computational results. The definitions of these problems, where $n = 3$, $p = 2$ or $p = 3$, which possess certain geometrical structures, can be found in [2]. The results are given in Table 3.

The results of computational experiments motivate heuristic modifications of the algorithm. For instance, one may aspire to find efficient faces that belong to a particular set $\mathcal{M}^k$, where k is determined by finding an appropriate face $F(I) \in \mathcal{M}^k$ in which an initial efficient point that has been obtained lies. To see if this type of an approach would generate good results and if it would bring computational benefits requires further study.

References

[1] P. Armand, "Finding all maximal efficient faces in multiobjective linear programming", *Math. Programming* **61**, 357–375 (1993).

[2] P. Armand and C. Malivert, "Determination of the efficient set in multiobjective linear programming", *J. Optimn. Theory Appl.* **70**, 467–489 (1991).

[3] H.P. Benson, "Complete efficiency and the initialization of the algorithms for multiple objective programming", *Oper. Res. Lett.* **10**, 481–487 (1991).

[4] H.P. Benson and S. Sayin, "Finding global representations of the efficient set in multiple objective mathematical programming", Discussion paper, Department of Decision and Information Sciences, Univerity of Florida, 1994.

[5] Cplex, *Using the Cplex Callable Library and Cplex Mixed Integer Library, Version 2.1*, Cplex Optimization, Inc, 1993.

[6] J.P. Dauer and R.J. Gallagher, "A combined constraint-space, objective-space approach for determining high-dimensional maximal efficient faces of multiple objective linear programs", *European J. Oper. Res.* **88**, 368–381 (1996).

[7] J.S. Dyer, P.C. Fishburn, R.E. Steuer, J. Wallenius, and S. Zionts, "Multiple criteria decision making, multiattribute utility theory: The next ten years", *Management Sci.* **38**(5), 645–654 (1992).

[8] J.G. Ecker, N.S. Hegner and I.A. Kouada, "Generating all maximal efficient faces for multiple objective linear programs", *J. Optim. Theory Appl.* **30**, 353–381 (1980).

[9] J.G. Ecker and I.A. Kouada, "Finding all efficient extreme points for multiple objective linear programs", *Math. Programming* **14**, 249–261 (1978).

[10] G.W. Evans, "An overview of techniques for solving multiobjective mathematical programs", *Management Sci.* **30**, 1268–1282 (1984).

[11] J.P. Evans and R.E. Steuer, "A revised simplex method for linear multiple objective problems", *Math. Programming* **5**, 54–72 (1973).

[12] T. Gal, "A general method for determining the set of all efficient solutions to a linear vector maximum problem", *European J. Oper. Res.* **1**, 307–322 (1977).

[13] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[14] R.E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley, New York, 1986.

[15] P.L. Yu and M. Zeleny, "The set of all nondominated solutions in linear cases and a multicriteria simplex method", *J. Math. Anal. Appl.* **49**, 430–468 (1975).