RAIRO-Oper. Res. 42 (2008) 371–387
DOI: 10.1051/ro:2008018

RAIRO Operations Research
www.rairo-ro.org

ENUMERATING THE SET OF NON-DOMINATED VECTORS IN MULTIPLE OBJECTIVE INTEGER LINEAR PROGRAMMING

John Sylva[1] and Alejandro Crema[2]

Abstract. An algorithm for enumerating all nondominated vectors of multiple objective integer linear programs is presented. The method tests different regions where candidates can be found using an auxiliary binary problem for tracking the regions already explored. An experimental comparision with our previous efforts shows the method has relatively good time performance.

Keywords. Integer Programming, Multiple Objective Programming, Parametric Programming.

Mathematics Subject Classification. 90C10, 90C11, 90C29.

1. Introduction

Generating non-dominated vectors and efficient solutions in multiple objective mathematical programming has been a field of interest both as a theoretical problem and as a part of Multiple Criteria Decision Making procedures. Multiple Objective Integer Linear Programming (MOILP) arises in Multiple Criteria Decision Making (MCDM) problems involving discrete decisions [14]. Several procedures have been developed for generating the non-dominated set for MOILP problems [3,7,12,14] as well as interactive procedures for these problems [1,5,9].

The authors have contributed in this field with two different approaches for generating the set of non-dominated vectors in MOILP problems; in our first

Received September 15, 2006. Accepted January 17, 2008.

[1] Departamento de Matemáticas Aplicadas, Facultad de Ingeniería, Universidad Central de Venezuela, Caracas, Venezuela; jsylva@cantv.net

[2] Escuela de Computación, Facultad de Ciencias,Universidad Central de Venezuela, Apartado 47002, Caracas 1041-A, Venezuela; acrema@kuaimare.ciens.ucv.ve

Article published by EDP Sciences

proposal, the results were generated in decreasing order of some fixed weighted objective function [10], while in our second proposal the output was ordered to obtain a well dispersed subset of vectors [11].

In this paper, we present a different approach for solving the problem where no specific order is imposed to the output. At each stage, the algorithm tries to find a new nondominated vector by selecting for each known nondominated vector an objective to surpass, testing different combinations until a new efficient solution is found. The control of the combinations of objectives tested is made using an auxiliary binary ILP problem. Although this simple approach cannot control the order of apparition of the solutions, it offers a much better time performance than our previous methods.

The division of the search space for solving MOILP problems has also been used in algorithms recently proposed by Laumanns *et al.* [6] and Tenfelde-Podehl [13]. The main difference with these algorithms is that they split the objective space using a grid whose nodes are determined by the values of all previously found non dominated vectors, imposing upper and lower bounds to the values of the objective functions. This leads to ILP problems whose solutions cannot be guaranteed to correspond to non dominated solutions. The method presented here, on the contrary, does not impose explicit upper bounds to the objective functions when defining partial search regions and therefore it will always find a new non dominated solution, if it exists, or will lead to an infeasible problem if such solution does not exists in that region. The procedure can be adapted to find subsets of nondominated vectors

2. Theoretical basis and algorithm

The MOILP problem can be stated as:

$$(P)\text{: ``max''}\ \{Cx\text{: } Ax \leq b,\ x \geq 0,\ x \in Z^n\}$$

where $C \in Z^{p\times n}$, $A \in R^{m\times n}$ and $b \in R^m$. Cx represents p objective functions, $Ax \leq b$ represents m linear constraints and x represents n integer decision variables. The feasible set of problem (P) will be denoted $F(P)$. In this work, we assume that $F(P)$ is bounded and non-empty.

Because of conflicting objectives, there is not usually a maximum solution but a set of non-dominated vectors.

Definition 1. *A feasible solution x^* to problem (P) is an* efficient solution *iff there is not another feasible x such that $Cx \geq Cx^*$ with at least one strict inequality. The resulting criterion vector Cx^* is said to be* non-dominated.

A well-known result connecting Multiple Objective Programming and Parametric Programming is the following [9]:

Theorem 1. *If x^* is an optimal solution to the (single objective) problem:*

$$\max\left\{\lambda^t Cx\text{: } x \in S\right\}$$

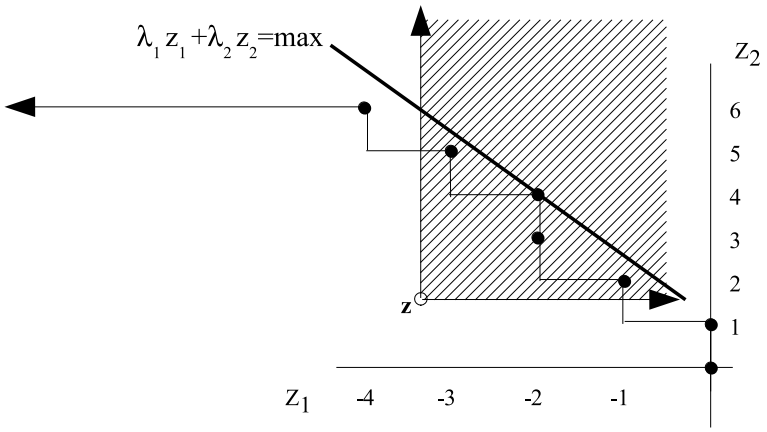

FIGURE 1. Non-dominated vectors greater than a fixed z are non dominated to (P).

for some $\lambda \in \mathbb{R}^p$, $\lambda > 0$, *then* x^* *is an efficient solution to problem:*

$$\text{“max”} \ \{Cx\colon\ x \in S\}\,.$$

Efficient solutions that are optimal to the parametric problem in Theorem 1 are said to be *supported efficient solutions*. Unlike Multiple Objective Linear Programming, the converse of this theorem does not hold when some variables are integer [2] as some efficient solutions (known as *unsupported efficient solutions*) may not be optimal for any $\lambda > 0$.

The result of Theorem 1 still holds true if we constrain the search region:

Proposition 1. *Let* $z \in \mathbb{R}^p$ *be a fixed objective vector,*$\lambda \in \mathbb{R}^p$ *a fixed positive weight vector and* x^* *an optimal solution to the problem:*

$$\max\left\{\lambda^t Cx\colon\ Cx \geq z,\ x \in F\,(P)\right\}.$$

Then x^* *is an efficient solution to problem* (P).

Proof. Let us suppose that x^* is not efficient. Then there exist $\widetilde{x} \in S$ such that $C\widetilde{x} \geq Cx^*$ with at least one strict inequality. As $C\widetilde{x} \geq Cx^* \geq z$ and $\lambda^t C\widetilde{x} > \lambda^t Cx^*$, x^* is not optimal. This contradicts the hypothesis that x^* is an optimal solution (see Fig. 1). □

Taking z as a parameter[1] we can find the whole set of non dominated vectors [8]:

[1]Theoretically, this holds true if z takes all possible values in $\mathbb{R}^p$; in practice, the values of z are chosen according to the values of known non-dominated vectors.

Proposition 2. *The set of efficient solutions to problem* (P) *is the set of optimal solutions to the parametric problem:*

$$
\begin{array}{lll}
(P_{\lambda,z}): & & \\
\quad max & \lambda^t Cx & \\
\quad s.t. & Ax \leq b & \text{for an arbitrarily fixed } \lambda > 0 \text{ and all } z \in R^n. \\
& Cx \geq z & \\
& x \geq 0,\ x \in Z^n &
\end{array}
$$

Proof. From Proposition 1 we have that any optimal solution x^* to $(P_{\lambda,s})$ is efficient to (P). Conversely, if x^* is efficient to (P) it is an optimal solution to $(P_{\lambda,z})$ for $z = Cx^*$. □

2.1. Algorithm

The basic idea of the algorithm is to find an initial non dominated vector using Theorem 1 and then sequentially find new vectors applying Proposition 1 with z such that for each non dominated vector previously found at least one component of z is strictly greater. The decision of increasing or not the value of the k-th objective function on the s-th solution is controlled by a binary variable y_k^s ; a value of 1 means we must improve the value of the function while a value of 0 indicates we will not necessarily do so. Then for each non dominated vector z^s we have a constraint $\sum_{k=1}^{p} y_k^s \geq 1$ to guarantee that at least one objective function is surpassed. For a given $\underline{y} = \left(\underline{y_k^s}\right)_{k=1,\ldots,p;\ s=1,\ldots l}$ we will have a constraint $(Cx)_k > (Cx^s)_k$ when $\underline{y_k^s} = 1$ or, more tersely, for each objective function k we have the constraint:

$$(Cx)_k > \max_{\substack{s=1\ldots l \\ \underline{y_k^s}=1}} \{(Cx^s)_k\}$$

where l is the number of previously found non dominated vectors. By convention, $\max_{\substack{s=1\ldots l \\ \underline{y_k^s}=1}} \{(Cx^s)_k\} = -\infty$ when $\underline{y_k^s} = 0$ for all $s = 1, ..., l$.

In a MOILP problem with an integer cost matrix C we can restate this constraint as $(Cx)_k \geq \max_{\substack{s=1\ldots l \\ \underline{y_k^s}=1}} \{(Cx^s)_k\} + 1$. For large problems, the enumeration of all non-dominated vectors may not be practical, and a representative subset of the non-dominated vectors can be generated changing this constraint to $(Cx)_k \geq \max_{\substack{s=1\ldots l \\ \underline{y_k^s}=1}} \{(Cx^s)_k\} + f_k$; where f_k is a fixed value representing a minimal increment on objective k for a new solution to be noteworthy.

The management of the selected combination $\underline{y}$ is basically an enumerative scheme. In this work we did this enumeration using a linear integer system (L) implemented as an ILP with an arbitrary objective function. Other enumeration schemes can be used, however, this choice reuses ILP routines already being used for solving weighted objective problems.

Some $\underline{y_k^s}$ combinations will lead to an empty search region. In order to avoid such a failed combination $\underline{y_k^s}$ to reappear it is necessary that at least one variable

y_k^s that was equal to 1 be changed to 0 in future attempts (that is, at least one of the objectives that were requested to rise should not be increased). Defining $K = \left\{(k,s) \mid \underline{y}_k^s = 1;\ k = 1...p;\ s = 1...l\right\}$, this is equivalent to ask that the sum of all indices corresponding to those in K must be strictly less than the cardinality of this set:

$$\sum_{(k,s)\in K} y_k^s \leq \|K\| - 1.$$

Notice that this constraint not only will rule out $\underline{y}_k^s$ but also any other one that includes the increases requested by the failed combination. This suggests trying first the combinations with the least requested increases; that is, implementing the linear system (L) as an ILP with the objective of minimizing the sum of the y_k^s variables.

2.1.1. *Preliminary Algorithm*

Accordingly to these ideas we have the following procedure:

(1) (Initialization step) Choose $\lambda > 0$ and steps $f_k > 0, k = 1, ...p$. Solve $P_\lambda : \max\{\lambda^t Cx\colon\ x \in F(P)\}$. If there is no solution then *stop*; otherwise, let x^1 be an optimal solution; $l = 1$.
Solve the problem:

$$L : \min\left\{\sum_{s=1}^{l}\sum_{k=1}^{p} y_k^s\colon\ \sum_{k=1}^{p} y_k^1 \geq 1,\ y_k^1 \in \{0,1\},\ k = 1...p\right\}.$$

Let $\underline{y}^1$ be an optimal solution (one solution is $\underline{y}_1^1 = 1$; $\underline{y}_k^1 = 0$, $k = 2, ..., p$).

(2) Let $\underline{z}$ be such that $\underline{z}_k = \max_{\substack{s=1...l \\ \underline{y}_k^s = 1}}\{(Cx^s)_k + f_k\}$; $\underline{z}_k = -\infty$ if $\underline{y}_k^s = 0$ for all $s = 1...l$. Let P_λ^l be the problem obtained adding the constraints $Cx \geq \underline{z}$ to P_λ. Solve P_λ^l. If there is no solution, go to step 3; otherwise, go to step 4.

(3) Add the constraint

$$\sum_{\substack{k=1...p \\ s=1...l \\ \underline{y}_k^s = 1}} y_k^s \leq -1 + \sum_{\substack{k=1...p \\ s=1...l \\ \underline{y}_k^s = 1}} 1$$

to problem (L). Go to step 5.

(4) Let x^{l+1} be an optimal solution to P_λ^l; let $l = l+1$ (this implies modifying the objective function in L) and add the constraints

$$\sum_{k=1}^{p} y_k^l \geq 1,\ y_k^l \in \{0,1\},\ k = 1...p$$

to problem (L).

(5) Solve L. If there is no solution *stop*; otherwise, let $\underline{y} = (\underline{y}_k^s)_{\ k=1...p;\ s=1...l}$ be an optimal solution. Go to step 2.

At the end of the process, we have a list of efficient solutions $\{x^s\}_{s=1...l}$ and a set of non dominated vectors $\{Cx^s\}_{s=1...l}$.

Proposition 3. *The algorithm ends in a finite number of steps when applied to a Multiple Objective Mixed Integer Linear Programming problem with a bounded feasible region for any* $f_k > 0,\ k = 1, ...p.$

Proof. First we will prove that the algorithm yields a new solution or ends in a finite number of steps (that is, step 2 eventually leads to step 4 or to the stop condition in step 5). In effect, at the beginning of step 2, Problem L has a finite number of feasible solutions and each time step 3 is executed at least one solution $\underline{y}$ is deleted from the feasible set; therefore, we will ultimately get a feasible problem P_λ^l or an infeasible problem L .

Now we show that the number l of solutions generated by the algorithm is finite. Defining $\underline{\underline{z}}$ such that $\underline{\underline{z}}_k = \min_{x\in F(P)} \{(Cx)_k\} - f_k$, $\Delta^s = \{z \mid Cx^s - f < z \leq Cx^s\}$ where $f = (f_{1,...,}f_p)^T$, $\overline{W} = \left\{z \mid \underline{\underline{z}} \leq z \leq Cx, \text{ for some } x \in F(P)\right\}$ and $W^s = \left\{z \mid \underline{\underline{z}} \leq z \leq Cx^s\right\}$ for $s = 1, ..., l$; we can verify that $\Delta^s \subset W^s \subset \overline{W}$ but for any $s' < s$ there is a k such that $(Cx^s)_k \geq (Cx^{s\prime})_k + f_k$ so $\Delta^s \cap W^{s\prime} = \emptyset$ and therefore $\Delta^s \cap \Delta^{s\prime} = \emptyset$ for $s' < s$.

If μ represents the usual measure[2] in R^p, then the measure of Δ^s is constantly equal to $\mu(\Delta^s) = \prod_{k=1}^{p} f_k$, and we have that $\mu\left(\bigcup_{s=1}^{l} \Delta^s\right) = l \prod_{k=1}^{p} f_k$. But $\bigcup_{s=1}^{l} \Delta^s \subset \overline{W}$ so $l \prod_{k=1}^{p} f_k \leq \mu(\overline{W})$ and $l \leq \mu(\overline{W}) / \prod_{k=1}^{p} f_k$. (See Fig. 2) □

Proposition 4. *When applied to MOILP with a bounded feasible region and an integer cost matrix* C *the algorithm generates all non dominated vectors when* $f_k = 1,\ k = 1, ...p.$

Proof. Let $\{Cx^s\}_{s=1...l}$ be the set of non dominated vector generated at the end of the algorithm and let $\widetilde{x} \in F(P)$ efficient such that $C\widetilde{x} \neq Cx^s$ for all $s = 1...l$. Let $\widetilde{y} = \left(\widetilde{y_k^s}\right)_{k=1...p;\ s=1...l}$ such that $\widetilde{y_k^s} = 1$ if $(C\widetilde{x})_k > (Cx^s)_k$, $\widetilde{y_k^s} = 0$ otherwise. At the end of the algorithm the last version of problem L is infeasible and thus $\widetilde{y}$ must violate some constraint in L.

As $C\widetilde{x}$ is non dominated, for each $s = 1...l$ there is k such that $(C\widetilde{x})_k \geq (Cx^s)_k + 1$ and therefore all constraints $\sum_{k=1}^{p} y_k^s \geq 1$ must hold. Then we are

[2]That is, the measure of a hyperbox with opposite vertices at the origin and $(f_1, ..., f_p)$ is $\prod_{k=1}^{p} f_k$.

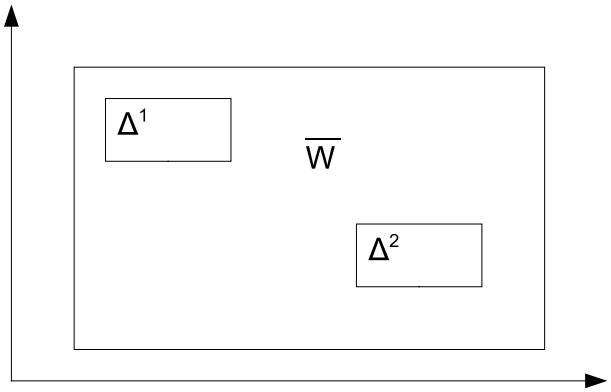

FIGURE 2. The Δ^s are non-intersecting equally-sized subsets of $\overline{W}$ so there can only be a finite number of them.

violating some constraint of the form

$$\sum_{\substack{k=1\ldots p\\ s=1\ldots l\\ \underline{y}_k^s=1}} y_k^s \leq -1 + \sum_{\substack{k=1\ldots p\\ s=1\ldots l\\ \underline{y}_k^s=1}} 1$$

for some $\underline{y}$ that led to an infeasible P_λ^l. As $\widetilde{y}$ violates this constraint, $\underline{y}_k^s = 1 \Rightarrow \widetilde{y}_k^s = 1$ so in the construction of this problem P_λ^l :

$$\underline{z}_k = \max_{\substack{s=1\ldots l\\ \underline{y}_k^s=1}} \{(Cx^s)_k + 1\} \leq \max_{\substack{s=1\ldots l\\ \widetilde{y}_k^s=1}} \{(Cx^s)_k + 1\} \leq (C\widetilde{x})_k$$

and $\widetilde{x}$ is feasible to P_λ^l contradicting that this problem is infeasible. □

2.1.2. *Order Constraint between Artificial Variables*

As stated now, the algorithm relies on the enumeration of possible values of the artificial variables y_k^s. As the number of combinations grows exponentially with the number of solutions and objective functions, we need to incorporate ideas to lower the number of combinations we must try. Recalling that $y_k^l = 1$ means trying to find a better value for the k-th objective function than on the l-th solution, we find that some y_k^s combinations are logically inconsistent.

The algorithm can be improved noticing that if $y_k^s = 1$ then for each solution $x^{s\prime}$ such that $(Cx^{s\prime})_k \leq (Cx^s)_k$ we can only consider the possibility of $y_k^{s\prime} = 1$ as increasing $(Cx^s)_k$ implies raising $Cx^{s\prime}$, then we can add the constraint $y_k^{s\prime} \geq y_k^s$ whenever $(Cx^{s\prime})_k \leq (Cx^s)_k$. The algorithm is easily modified by inserting between steps 4 and 5 the following instruction:

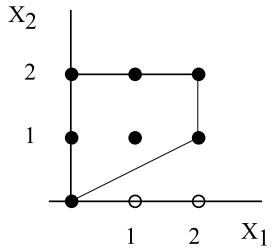

FIGURE 3. Feasible set.

For each $k = 1, ..., p$ find a permutation σ_k on the index set $\{1, ..., s\}$ such that $(Cx^s)_{\sigma_k(1)} \geq (Cx^s)_{\sigma_k(2)} \geq ... \geq (Cx^s)_{\sigma_k(s)}$ and add to problem L the constraints $y_k^{\sigma_k(1)} \leq y_k^{\sigma_k(2)} \leq ... \leq y_k^{\sigma_k(s)}$.

As this modification only adds constraints to problem L, the proof of Proposition 3 remains valid. We now prove that the conclusion of Proposition 4 still holds:

Proposition 5. *When applied to MOILP with a bounded feasible region an integer cost matrix C the modified algorithm generates all non dominated vectors when $f_k = 1,\ k = 1, ...p$.*

Proof. Defining $\widetilde{x}$ and $\widetilde{y}$ as in the proof of Proposition 4 , we have already proved that $\widetilde{y}$ does not violate the constraints of the preliminary algorithm then it must violate some constraint of the form $y_k^s \leq y_k^{s\prime}$ for a pair of solutions such that $(Cx^s)_k \geq (Cx^{s\prime})_k$. Then we must have $\widetilde{y}_k^s = 1$ and $\widetilde{y}_k^{s\prime} = 0$ that accordingly to the construction of $\widetilde{y}$ corresponds to $(C\widetilde{x})_k > (Cx^s)_k$ and $(C\widetilde{x})_k \leq (Cx^{s\prime})_k$ contradicting that $(Cx^s)_k \geq (Cx^{s\prime})_k$. □

However, when the hypothesis of the proposition are not met ($f_k \neq 1$) the algorithms can yield different subsets of non dominated vectors.

3. Numerical example

Let us consider the MOILP problem:

$$
\begin{array}{lll}
(P): & \text{“max”} & x_1 - 2x_2 \\
 & & -x_1 + 3x_2 \\
 & \text{s.t.} & x_1 - 2x_2 \leq 0 \\
 & & x_1, x_2 \in \{0, 1, 2\}.
\end{array}
$$

The feasible region is shown in Figure 3 while the objective space image is shown in Figure 4. For this example we choose $\lambda = (4, 3)^T$. As we want to enumerate all non dominated vectors we take $f_1 = f_2 = 1$. An initial solution is found solving:

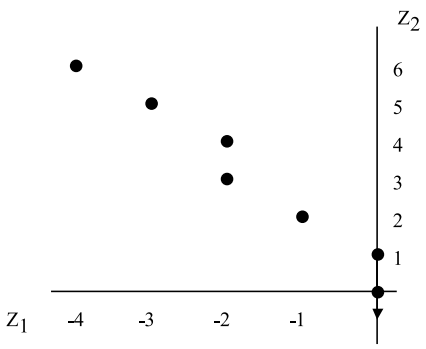

FIGURE 4. Feasible objective vectors.

$$
\begin{array}{lll}
(P^0_\lambda) & & \\
\quad \max & x_1 + x_2 & \\
\quad \text{s.t.} & x_1 - 2x_2 \leq 0 & \\
& x_1, x_2 \in \{0, 1, 2\}\,. &
\end{array}
$$

An optimal solution to this ILP problem is $x^1 = (2, 2)^T$ with an optimal objective function value $v(P^0_\lambda) = 4$. Then $x^1 = (2, 2)^T$ is an efficient solution to problem (P) with a (non dominated) objective function value vector equal to $(-2, 4)^T$.

Now we select an objective to raise by solving the following linear integer problem:

$$
\begin{array}{lll}
(L) \quad \min & y^1_1 + y^1_2 & \\
\quad \text{s.t.} & y^1_1 + y^1_2 \geq 1 & \\
& y^1_1, y^1_2 \in \{0, 1\}\,. &
\end{array}
$$

There are two solutions, $(1, 0)^T$ and $(0, 1)^T$. For this example we will always choose the solution where the value 1 is assigned to the variable with the smallest possible index,so we will use $y^1 = (1, 0)^T$. This indicates we will add restrictions to raise the value of the first objective function (see Figs. 5 and 6):

$$
\begin{array}{lll}
(P^1_\lambda) \quad \max & x_1 + x_2 & \\
\quad \text{s.t.} & x_1 - 2x_2 \leq 0 & \\
& x_1 - 2x_2 \geq -2 + 1 = -1 \ ([3]) & \\
& x_1, x_2 \in \{0, 1, 2\}\,. &
\end{array}
$$

The optimal solution is $x^2 = (2, 1)^T$ with an objective value $v(P^1_\lambda) = 3$. We have found a new efficient solution $x^2 = (2, 1)^T$ and a non dominated vector $(0, 1)^T$.

[3]We will look for solutions with a better value of the first objective function.

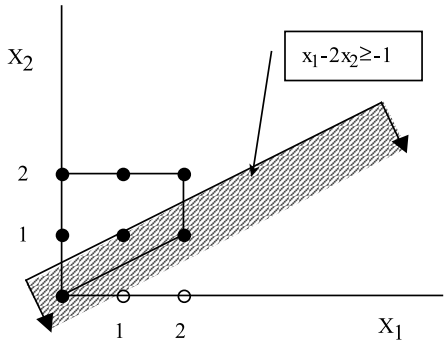

FIGURE 5. Search region for P_λ^1 (decision variable space). Feasible region is the shaded area.

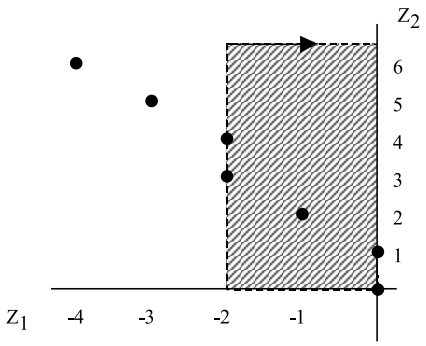

FIGURE 6. Search region for P_λ^1 (objective function space). Feasible region is the shaded area.

Now we solve:

$$
\begin{array}{lll}
(L) & \min & y_1^1 + y_2^1 + y_1^2 + y_2^2 \\
 & \text{s.t.} & y_1^1 + y_2^1 \geq 1 \\
 & & y_1^2 + y_2^2 \geq 1 \\
 & & y_1^1 \geq y_1^2 \ (^4) \\
 & & y_2^2 \geq y_2^1 \\
 & & y_1^1, y_2^1, y_1^2, y_2^2 \in \{0,1\}.
\end{array}
$$

We consider first the solution $y^1 = y^2 = (1,0)^T$, getting the problem:

$$
\begin{array}{lll}
(P_\lambda^2) & \max & x_1 + x_2 \\
 & \text{s.t.} & x_1 - 2x_2 \leq 0 \\
 & & x_1 - 2x_2 \geq 1 \\
 & & x_1, x_2 \in \{0,1,2\}.
\end{array}
$$

[4]If $y_1^2 = 1$ we will be looking for solutions with first objective value greater than that of the second solution (0), then it will also be better than the respective value in the first solution (-2), that is we can assume that $y_1^1 = 1$.

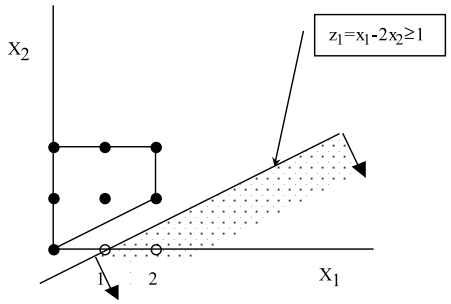

FIGURE 7. No feasible solution has a better first objective.

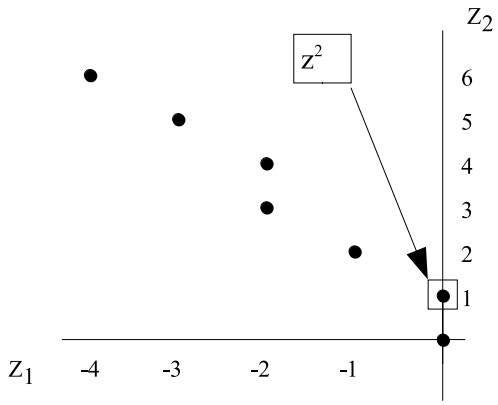

FIGURE 8. No feasible solution has a better first objective than $z^2 = (0,1)^T$.

In this problem we are trying to find a solution with a better first objective function value than the two previous solutions (Fig. 7). This problem is infeasible so we add a new constraint to L:

$$
\begin{array}{lll}
(L) & \min & y_1^1 + y_2^1 + y_1^2 + y_2^2 \\
 & \text{s.t.} & y_1^1 + y_2^1 \geq 1 \\
 & & y_1^2 + y_2^2 \geq 1 \\
 & & y_1^1 \geq y_1^2 \\
 & & y_2^2 \geq y_2^1 \\
 & & y_1^1 + y_1^2 \leq 1 \\
 & & y_1^1, y_2^1, y_1^2, y_2^2 \in \{0,1\}\,.
\end{array}
$$

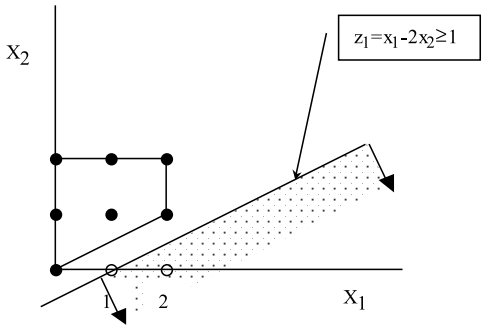

FIGURE 9. Higher first objective than x^1 and second objective than x^2.

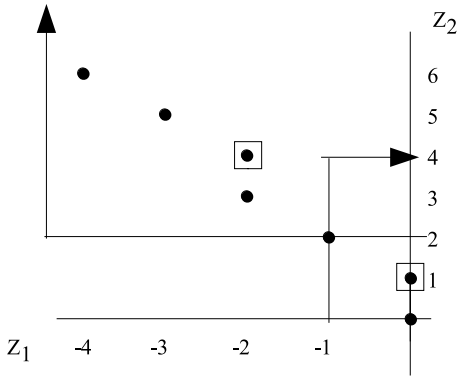

FIGURE 10. Looking for a first objective better than $(-2,4)^T$ and second objective better than than $(0,1)^T$, we find vector $(-1,2)^T$.

This problem admits the solution $y_1^1 = y_2^2 = 1$, $y_2^1 = y_1^2 = 0$; that is, we will now try to find a solution with a higher first objective value than x^1 and a second objective value better than x^2 (Fig. 9). The corresponding problem (P_λ^2) is:

$$
\begin{array}{lll}
(P_\lambda^2) & \max & x_1 + x_2 \\
& \text{s.t.} & x_1 - 2x_2 \leq 0 \\
& & x_1 - 2x_2 \geq -1 \\
& & -x_1 + 3x_2 \geq 2 \\
& & x_1, x_2 \in \{0,1,2\}\,.
\end{array}
$$

This problem has an optimal solution $x^3 = (1,1)^T$ with an objective vector $(-1,2)^T$.

If we continue this procedure until the problem (L) is infeasible, the process ends and we will have all non dominated vectors and one efficient solution corresponding to each one of them (Tab. 1).

TABLE 1. Efficient solutions and non dominated vectors (P).

(x_1, x_2)	(z_1, z_2)
$(2, 2)$	$(-2, 4)$
$(2, 1)$	$(0, 1)$
$(1, 1)$	$(-1, 2)$
$(1, 2)$	$(-3, 5)$
$(0, 2)$	$(-4, 6)$

4. Computational results

The methods were programmed in MS Visual C++ 2005 Express Edition Beta 2 and executed on a Sony VAIO VGN-A190 under Windows XP SP2. Linear integer problems were solved using the branch and bound routine of the COIN-OR library [4]. For each problem size, 30 different problems were randomly generated and executed five times[5]. For each problem size, mean and maximal number of solutions and CPU times are reported. The method presented in this article is referred to as "undirected search" while our previous methods are labeled as "weighted objective" [10] and "maximal dispersion" [11]. In all cases, weighted problems were solved giving an equal importance to each objective. The integer linear programs are not solved incrementally although the last basis of the linear relaxation is used as an initial basis for the next iteration.

The first batch of problems are multiconstrained 0-1 knapsack problems with two objectives functions. Objective functions and constraint coefficients are uncorrelated integers uniformly distributed between 1 and 99. For each constraint, the right-hand side value is set to a (truncated) 50% of the sum of its coefficients. For all these problems, the complete set of non-dominated vectors was generated. Results are shown on Table 2.

The second group of tests involves randomly generated multiconstrained 0-1-2 knapsack problems with two objectives functions. Objective functions and constraint coefficients are uncorrelated integers uniformly distributed between 1 and 99. For each constraint, the right-hand side value is set to the sum of its coefficients. The complete set of non-dominated vectors was generated. Table 3 shows the results of these experiments.

The method was also tested with multiconstrained 0-1 knapsack problems with three objective functions. Objective functions and constraint coefficients are randomly generated integers between 1 and 999 and the right-hand side values of the constraints are set to one half of the sum of its coefficients. In this case, there is obviously a much larger set of non-dominated vectors and the complexity of integer linear programs grows at a faster rate with every new solution found. Therefore, only a subset solutions was generated; this was accomplished by stipulating an increment of at least 500 units in some objective for any new solution generated.

[5]Experiments are repeated five times in order to average execution times which are variable when working under windows XP.

TABLE 2. Knapsack problems, 2 objectives, m constraints and n variables.

			Method					
			Undirected search		**Weighted objective**		**Maximal dispersion**	
m	**n**		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
5	15	Solutions	7.2	13	7.2	13	7.2	13
		Time[s]	0.774	2.544	1.795	8.192	1.783	10.064
	20	Solutions	12.8	22	12.8	22	12.8	22
		Time[s]	2.700	8.251	12.184	45.085	15.808	63.281
	25	Solutions	14.5	25	14.5	25	14.5	25
		Time[s]	5.592	25.357	33.279	273.884	36.720	223.962
	30	Solutions	20.8	36	20.8	36	20.8	36
		Time[s]	13.014	33.307	120.244	410.430	132.971	438.601
10	15	Solutions	7.3	19	7.3	19	7.3	19
		Time[s]	1.190	5.769	2.824	26.007	3.752	34.099
	20	Solutions	13.3	30	13.3	30	13.3	30
		Time[s]	5.713	26.158	30.925	261.036	38.339	322.093
	25	Solutions	16.7	28	16.7	28	16.7	28
		Time[s]	8.320	18.977	50.074	219.285	66.567	251.832
	30	Solutions	21.0	33	21.0	33	21.0	33
		Time[s]	23.697	87.987	221.954	942.885	271.993	1147.180

TABLE 3. Knapsack, 2 objectives, m constraints and n 0/1/2 variables.

			Method					
			Undirected search		**Weighted objective**		**Maximal dispersion**	
m	**n**		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
5	12	Solutions	12.4	31	12.4	31	12.4	31
		Time[s]	1.765	14.521	9.616	158.628	9.911	142.695
	15	Solutions	17.5	37	17.5	37	17.5	37
		Time[s]	4.062	14.311	34.170	203.092	36.212	221.699
	18	Solutions	19.7	41	19.7	41	19.7	41
		Time[s]	6.657	28.100	74.958	647.260	67.221	375.030
10	12	Solutions	10.2	20	10.2	20	10.2	20
		Time[s]	1.918	6.289	7.044	36.032	7.564	39.166
	15	Solutions	18.2	35	18.2	35	18.2	35
		Time[s]	6.488	16.414	48.948	217.382	57.977	252.543
	18	Solutions	22.8	48	22.8	48	22.8	48
		Time[s]	13.591	59.336	141.323	1103.810	179.028	1683.020

The same parameter was used in the two other methods and results are shown on Table 4.

Another group of experiments consisted in General Assignment Problems (GAP) with two objective functions. The GAP deals with the optimal allocation of s agents to a group of t tasks in such a way that each task j is assigned to exactly

TABLE 4. Knapsack , 3 objectives, m restricciones and n variables.

			Method					
			Undirected search		**Weighted objective**		**Maximal dispersion**	
m	**n**		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
5	15	Solutions	3.4	8	3.4	7	4.7	12
		Time[s]	0.282	1.142	0.475	2.994	1.271	11.006
	20	Solutions	4.6	8	4.6	8	7.2	12
		Time[s]	0.701	1.933	1.408	5.719	4.398	16.654
	25	Solutions	6.6	13	6.6	13	11.4	21
		Time[s]	1.841	6.419	6.233	29.993	21.243	103.238
	30	Solutions	8.8	17	8.3	16	14.3	27
		Time[s]	4.180	12.097	19.313	89.789	60.956	344.676
10	15	Solutions	3.4	6	3.3	6	5.0	9
		Time[s]	0.504	1.452	0.802	3.605	2.496	12.098
	20	Solutions	4.8	8	4.7	7	7.4	12
		Time[s]	1.455	4.376	3.112	10.455	8.372	31.235
	25	Solutions	6.7	13	6.7	13	11.4	23
		Time[s]	3.937	16.093	14.218	95.958	49.926	285.16
	30	Solutions	9.6	14	9.1	14	15.7	22
		Time[s]	9.349	35.551	40.192	182.473	123.758	468.254

one agent i incurring a cost c_{ij} (a vector in the multiple objective case) and consuming r_{ij} units of a single resource subject to an availability of b_i for each agent. This results in the formulation:

$$
\begin{array}{lll}
(GAP): & \text{“min”} & \sum_{i=1}^{s}\sum_{j=1}^{t} c_{ij}x_{ij} \\
& \text{s.t.} & \sum_{j=1}^{t} r_{ij}x_{ij} \leq b_i;\ i = 1, ..., s \\
& & \sum_{i=1}^{s} x_{ij} = 1;\ j = 1, ..., t \\
& & x_{ij} \in \{0, 1\}_i;\ i = 1, ..., s;\ j = 1, ..., t.
\end{array}
$$

This formulation is easily modified to a bicriterion problem by considering each c_{ij} to be a vector instead of a number. In this experiments, c_{ij} components and r_{ij} are randomly selected integers between 1 and 999 and each b_i is fixed at a 50% of the sum of the corresponding r_{ij}. Only a subset of the non-dominated solutions was generated using a parameter value of 500. Tables 5–7 show the results of these tests.

5. CONCLUSIONS

According to the experimental results, the time performance of the method is superior than our two previous methods. On the qualitative aspect however the resulting nondominated vectors do not follow a useful order.

TABLE 5. GAP, 5 agents, t tasks.

		Method					
		Undirected search		**Weighted objective**		**Maximal dispersion**	
t		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
60	Solutions	21.2	32	21.2	32	28.1	37
	Time[s]	3.189	6.379	7.929	21.922	51.296	102.318
70	Solutions	24.2	31	24.2	31	31.7	42
	Time[s]	4.725	6.710	14.338	24.525	83.966	139.240
80	Solutions	28.7	32	28.8	32	37.3	44
	Time[s]	7.312	11.386	25.110	37.253	150.319	279.592
90	Solutions	31.7	36	31.7	36	42.6	50
	Time[s]	10.050	17.114	36.861	59.375	235.891	355.091
100	Solutions	37.1	45	37.1	45	48.5	57
	Time[s]	14.668	21.741	63.514	99.864	384.791	598.27

TABLE 6. GAP, 10 agents, t tasks.

		Method					
		Undirected search		**Weighted objective**		**Maximal dispersion**	
t		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
60	Solutions	23.5	27	23.5	27	31.1	37
	Time[s]	7.260	10.275	21.316	34.420	138.672	189.142
70	Solutions	27.2	35	27.2	35	38	46
	Time[s]	12.542	20.490	40.021	80.195	255.045	375.800
80	Solutions	30.7	39	30.8	39	43.4	53
	Time[s]	17.858	27.580	63.104	128.956	400.402	659.578
90	Solutions	35.3	41	35.3	41	48.8	59
	Time[s]	25.152	37.764	99.179	159.809	615.214	1024.02
100	Solutions	39.2	44	39.2	44	56.2	67
	Time[s]	33.502	53.146	142.537	205.546	958.227	1300.340

TABLE 7. GAP, 15 agents, t tasks.

		Method					
		Undirected search		**Weighted objective**		**Maximal dispersion**	
t		*mean*	*maximum*	*mean*	*maximum*	*mean*	*maximum*
60	Solutions	22.6	28	22.6	28	31.1	39
	Time[s]	12.428	17.415	33.896	51.364	197.895	298.349
70	Solutions	26.0	29	26.0	29	37.1	42
	Time[s]	19.078	29.202	56.097	89.989	378.447	533.127
80	Solutions	29.5	34	29.5	34	42.7	52
	Time[s]	28.825	44.194	94.939	142.875	618.016	997.074
90	Solutions	33.2	39	33.2	39	48.2	56
	Time[s]	39.202	57.232	139.680	210.232	948	1310.920
100	Solutions	36.9	42	36.9	42	54.4	62
	Time[s]	55.628	71.573	218.561	306.851	1487.194	2129.710

One possible advantage of this method is that it only adds a few additional constraints to the original problem which makes it easier to develop a specialized algorithm for a given problem class. It is also possible to develop a specific algorithm for the auxiliary binary problem L.

The method can be adapted to find subsets of nondominated vectors by introducing a step parameter although there is not an easy procedure for estimating a convenient value of this parameter. When this parameter is used, the method can be applied to problems with continuous variables.

References

[1] M.J. Alves and J.o Clímaco, An interactive reference point approach for multiobjective mixed-integer programming using branch and bound. *Eur. J. Oper. Res.* **124** (2000) 478–494.

[2] G.R. Bitran, Linear multiple objective programs with zero-one variables. *Math. Program.* **13** (1977) 121–139.

[3] J. Climaco, C. Ferreira and M.E. Captivo, Multicriteria integer programming: An overview of different algorithmic approaches, in *Multicriteria Analysis*, edited by J. Climaco, Springer-Verlag, Berlin, 1997, pp. 248–258.

[4] COIN-OR, *Computational infrastructure for operations research home page*, `http://www.coin-or.org/`, Acceso 26/03/2006.

[5] J. Karaivanova, S. Narula and V. Vassilev, An interactive algorithm for integer programming. *Eur. J. Oper. Res.* **68** (1993) 344–351.

[6] M. Laumanns, L. Thiele and E. Zitzler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *Eur. J. Oper. Res.* **169** (2006) 932–942.

[7] L.M. Rasmussen, Zero-one programming with multiple criteria. *Eur. J. Oper. Res.* **26** (1986) 83–95.

[8] R.M. Soland, The design of multiactivity multifacility systems. *Eur. J. Oper. Res.* **12** (1983) 95–104.

[9] R.E. Steuer, *Multiple criteria optimization-theory, computation and application*, John Wiley and Sons, (1986).

[10] J. Sylva and A. Crema, A method for finding the set of nondominated vectors for multiple objective integer linear programs. *Asia-Pacific J. Oper. Res.* **158** (2004) 46–55.

[11] J. Sylva and A. Crema, A method for finding well-dispersed subsets of nondominated vectors for multiple objective mixed integer linear programs. *Eur. J. Oper. Res.* **180** (2007) 1011–1027.

[12] J. Teghem and P.L. Kunsch, A survey of techniques for finding efficient solutions to multi-objective integer linear programming. *Asia-Pacific J. Oper. Res.* **3** (1986) 95–108.

[13] D. Tenfelde-Podehl, *A recursive algorithm for multiobjective combinatorial optimization problems with q criteria.* Tech. report, Institut für Mathematik, Technische Universität Graz, (2003).

[14] E.L. Ulungu and J. Teghem, Multi-objective combinatorial optimization problems: A survey. *J. Multi-Criteria Decision Anal.* **3** (1994), 83–104.