Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 158 (2004) 46–55

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

A method for finding the set of non-dominated vectors for multiple objective integer linear programs

John Sylva, Alejandro Crema [*]

Facultad de Ciencias, Universidad Central de Venezuela, Escuela de Computacion, Apdo. 47002, Caracas 1041-A, Venezuela

Received 5 September 2001; accepted 13 May 2002
Available online 9 August 2003

Abstract

An algorithm for enumerating all non-dominated vectors of multiple objective integer linear programs is presented. Using a straightforward theoretical approach, the problem is solved using a sequence of progressively more constrained integer linear programs generating a new solution at each step. The algorithm can also give subsets of efficient solutions that can be useful for designing interactive procedures for large, real-life problems.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Integer programming; Multiple objective programming; Parametric programming

1. Introduction

Multiple Objective Integer Linear Programming (MOILP) arises in Multiple Criteria Decision Making (MCDM) problems involving discrete decisions [16]. In the last decades, several methods have been developed to solve the MOILP, ranging from algorithms that find all non-dominated. vectors to methods that only try to find interesting solutions for decision-making purposes. Most methods for generating all non-dominated vectors are restricted to zero–one decision variables [1–4,7,8], though some can deal with general integer variables [5,17]. Surveys considering most of this methods are also available [12,18]. Some methods, published more recently, can be difficult to implement [13] or do not yield the complete set of non-dominated vectors [10].

An interesting approach by Klein and Hannan [9] solves a sequence of progressively more constrained Integer Linear Programming (ILP) problems. The objective function in these ILP problems is always a specific objective from the MOILP problem. At each stage of the algorithm, new constraints are added to eliminate solutions dominated by known non-dominated vectors producing a sequence of efficient solutions sorted by the selected objective function value. The authors of this method highlight two advantageous features for large MOILP problems: (1) A representative subset of efficient solutions can be generated by imposing constraints which allow only solutions that differ by at least some fixed amount in an objective function, and (2) As solutions are created

[*] Corresponding author. Tel.: +58-212-605-1037; fax: +58-212-605-1131.
E-mail address: acrema@kuaimare.ciens.ucv.ve (A. Crema).

sequentially, some efficient solutions can be found without solving the entire problem.

Unfortunately, this second remark may not be useful in the context of MCDM as the solutions are obtained in decreasing order in some fixed objective function (some good solutions could be found in later stages of the process). Moreover, maximizing an objective function will not necessarily yield an efficient solution to the MOILP (there could be an alternative optimum solution dominating the found one).

In this paper we present a variation of the Klein–Hannan algorithm, maximizing at each stage, not a single criterion, but a positive combination of all objective functions. With this approach, the solution of each ILP will be efficient; furthermore, if the weights assigned to each objective function represent a rough linear estimate of the Decision Maker's (DM) utility function, then the most interesting solutions will be generated at early stages of the process.

2. Definitions and theorems

The MOILP problem can be stated as

$$(P): \quad \text{“max”}\{Cx : Ax = b,\ x \geqslant 0,\ x \in Z^n\},$$

where $C \in Z^{p\times n}$, $A \in R^{m\times n}$ and $b \in R^m$. Cx represents p objective functions, $Ax = b$ represents m linear constraints and x represents n integer decision variables. The feasible set of problem (P) will be denoted $F(P)$.

Because of conflicting objectives, there is not usually a maximum solution but 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 $Cx \geqslant Cx^*$ 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 [15]:

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

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

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

$$\text{“max”}\{Cx : x \in S\}.$$

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 reciprocal of this theorem does not hold for MOILP [1] as some efficient solutions (known as *non-supported efficient solutions*) may not be optimal for any $\lambda > 0$. Nevertheless, new (supported and non-supported) solutions can be found if known efficient solutions (as well as solutions dominated by these) are removed from the feasible set.

Proposition 1. *Let $x^1, \ldots, x^l$ be efficient solutions to problem (P) and $D_s = \{x \in Z^n : Cx \leqslant Cx^s\}$.*

Let x^* be an efficient solution to the Multiple Objective Integer Problem

$$(\text{MOP}_l): \text{“max”}\{Cx : x \in F(P) - \cup_{s=1}^{l} D_s\}.$$

Then x^* is an efficient solution to problem (P). Furthermore, if problem (MOP_l) is unfeasible then $\{Cx^s\}_{s=1}^{l}$ is the entire set of non-dominated criterion vectors for problem (P).

Proof. Let us suppose there exists $x' \in F(P)$ such that $Cx^* \leqslant Cx'$ with at least one strict inequality. As x^* is an efficient solution to (MOP_l), x' cannot belong to $F(\text{MOP}_l)$, thus $x' \in \cup_{s=1}^{l} D_s$. Therefore $x' \in D_s$ for some $s \in \{1, \ldots, l\}$ and, accordingly to the definition of D_s, $Cx' \leqslant Cx^s$. As $Cx^* \leqslant Cx' \leqslant Cx^s$ we have that $x^* \in D_s$, contradicting that $x^* \in F(P) - \cup_{s=1}^{l} D_s$.

If (MOP_l) is unfeasible then $F(P) \subseteq \cup_{s=1}^{l} D_s$ and for any $x \in F(P)$ there exists an x^s such that $Cx \leqslant Cx^s$. Then we must have that $Cx = Cx^s$ (and $Cx \in \{Cx^s\}_{s=1}^{l}$) or $Cx \leqslant Cx^s$ with at least one strict inequality (and Cx is a dominated vector). □

From this result and Theorem 1, we have:

Corollary 1. *Let $x^1, \ldots, x^l$ be efficient solutions to problem (P) and $D_s = \{x \in Z^n : Cx \leqslant Cx^s\}$.*

If x^* is an optimal solution to

$$(PN_l) : \max\{\lambda^t Cx : x \in F(P) - \cup_{s=1}^{l} D_s\}$$

for some $\lambda \in R^p$, $\lambda > 0$, then x^* is an efficient solution to problem (P).

For implementation purposes, a linear version of problem (PN_l) will be formulated in the following section.

3. Implementation

The properties exposed in the previous section can be used to implement an algorithm for the bounded MOILP. After choosing a weight vector $\lambda > 0$, the first step of the algorithm consists in solving the ILP problem

$$(P_0) : \max\{\lambda^t Cx : Ax = b, x \geqslant 0, x \in Z^n\}.$$

If this problem is unfeasible, then problem (P) is unfeasible; otherwise, an optimal solution x^1 is found and, in accordance with Theorem 1, it is an efficient solution to Problem (P). Then a sequence of progressively more constrained problems (P_l) is solved. After l steps of the process, if problem (P_{l-1}) is unfeasible then the algorithm ends; otherwise, a new efficient solution x^l is found and a new problem (P_l) is defined deleting from the feasible set of (P_{l-1}) all solutions such that $Cx \leqslant Cx^l$. In order to use available ILP computer packages, this is implemented adding linear constraints to (P_{l-1}):

$$(Cx)_k \geqslant ((Cx^l)_k + 1)y_k^l - M_k(1 - y_k^l) \quad \text{for } k = 1, \ldots, p,$$

$$\sum_{k=1}^{p} y_k^l \geqslant 1, \quad y_k^l \in \{0, 1\} \quad \text{for } k = 1, \ldots, p,$$

where $-M_k$ is a lower bound for any feasible value of the kth objective function (for example, if all entries in C are positive, M_k can be set to 0 for all k). As adding these constraints is equivalent to deleting the set D_l from the feasible region, problem (P_l) is a linear equivalent to (PN_l) and any solution to this problem will be efficient to problem (P_l) as stated in Corollary 1:

$$\begin{aligned}
(P_l) : \max \quad & \lambda^t Cx \\
\text{s.t.} \quad & Ax = b, \\
& (Cx)_k \geqslant ((Cx^s)_k + 1)y_k^s - M_k(1 - y_k^s), \\
& \quad \text{for } s = 1, \ldots, l; \quad k = 1, \ldots, p, \\
& \sum_{k=1}^{p} y_k^s \geqslant 1, \quad y_k^s \in \{0, 1\} \\
& \quad \text{for } s = 1, \ldots, l; \quad k = 1, \ldots, p, \\
& x \geqslant 0, \quad x \in Z^n.
\end{aligned}$$

For large problems, the enumeration of all non-dominated vectors may not be practical. A representative subset of the non-dominated vectors can be generated changing problems (P_l) to

$$\begin{aligned}
(P_l) : \max \quad & \lambda^t Cx \\
\text{s.t.} \quad & Ax = b, \\
& (Cx)_k \geqslant ((Cx^s)_k + f_k)y_k^s - M_k(1 - y_k^s) \\
& \quad \text{for } s = 1, \ldots, l; \quad k = 1, \ldots, p, \\
& \sum_{k=1}^{p} y_k^s \geqslant 1, \quad y_k^s \in \{0, 1\} \\
& \quad \text{for } s = 1, \ldots, l; \quad k = 1, \ldots, p, \\
& x \geqslant 0, \quad x \in Z^n,
\end{aligned}$$

where f_k represents the minimal improvement in objective function k for a new non-dominated vectors to be noteworthy. Repeating until unfeasibility, this sequence of problems generates a set of "well spaced" non-dominated vectors.

If $f_k = 1$, for $k = 1, \ldots, p$, the algorithm will produce the whole set of non-dominated vectors (but not necessarily all efficient solutions) if the cost matrix C has integer elements. The procedure can be extended to the case of real costs if, for each criterion, a lower bound to the difference between any couple of feasible objective values is available; e.g., for rational costs $c_{kj} = q_{kj}/r_{kj}$ (integer q_{kj} and r_{kj}), $f_k = 1/\text{lcm}\{r_{kj}\}_{j=1,n}$ can be used.

When a good linear estimate of DM's utility function is available, it is not necessary to run the procedure until an unfeasible (P_l) as the most attractive solutions will be generated first.

This approach (for $f_k = 1$) is identical to solving the parametric problem [14]

$$
\begin{aligned}
(P_{\lambda,s}):\ \max\quad & \lambda^t Cx\\
\text{s.t.}\quad & Ax = b,\\
& Cx \geqslant s,\\
& x \geqslant 0,\quad x \in Z^n,
\end{aligned}
$$

for an arbitrarily fixed $\lambda > 0$ and all $s \in R^n$, using the multiparametric analysis method designed by Crema [6].

4. Numerical example

Let us consider the MOILP problem (Fig. 1)

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

In order to enumerate all non-dominated objective function values, f_1 and f_2 are set equal to 1. For this example, $M_1 = 4$ and $M_2 = 2$ are appropriate ($-M_1$ and $-M_2$ must be lower bounds of the corresponding objective functions). The weight vector λ can be any strictly positive vector, here we choose $\lambda = (4,3)^{\mathrm{T}}$. With this values, problem (P_0) is:

$$
\begin{aligned}
(P_0)\ \max\quad & x_1 + x_2\\
\text{s.t.}\quad & x_1 - 2x_2 \leqslant 0,\\
& x_1, x_2 \in \{0,1,2\}.
\end{aligned}
$$

An optimal solution to this ILP problem is $x^1 = (2,2)^{\mathrm{T}}$ with an optimal objective function value $v(P_0) = 4$. Then $x^1 = (2,2)^{\mathrm{T}}$ is an efficient

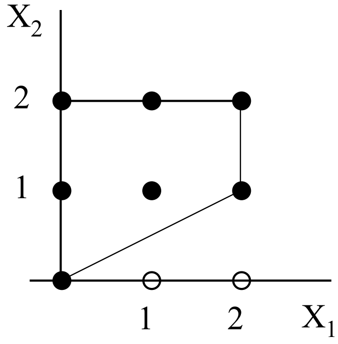

Fig. 1. Feasible region for P.

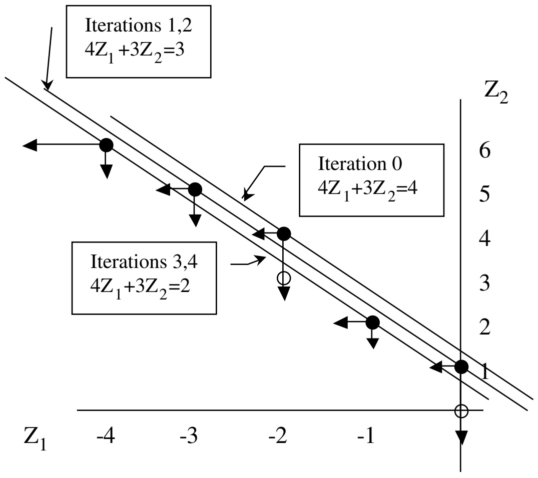

Fig. 2. Criterion space for problem P.

solution to problem (P) with a (non-dominated) objective function value vector equal to $(-2,4)^{\mathrm{T}}$. This non-dominated vector is shown in Fig. 2.

The next ILP problem to be solved is

$$
\begin{aligned}
(P_1)\ \max\quad & x_1 + x_2\\
\text{s.t.}\quad & x_1 - 2x_2 \leqslant 0,\\
& x_1 - 2x_2 \geqslant -y_1^1 - 4(1 - y_1^1),\\
& -x_1 + 3x_2 \geqslant 5y_2^1 - 2(1 - y_2^1),\\
& y_1^1 + y_2^1 \geqslant 1,\\
& x_1, x_2 \in \{0,1,2\},\\
& y_1^1, y_2^1 \in \{0,1\}.
\end{aligned}
$$

This new problem is a more constrained version of (P_0) where all points with objective function vector equal to or dominated by $(-2,4)^{\mathrm{T}}$ have been deleted from the feasible set; namely, the point $(0,1)^{\mathrm{T}}$ and the point x^1 itself (Fig. 3).

An optimal solution is $x_1 = 2$, $x_2 = 1$, $y_1^1 = 1$, $y_2^1 = 0$ with an optimal objective function value $v(P_1) = 3$. Then $x^2 = (2,1)^{\mathrm{T}}$ is another efficient solution to problem (P) with an objective function value vector equal to $(0,1)^{\mathrm{T}}$. Notice that the solution $x_1 = 1$, $x_2 = 2$, corresponding to the objective vector $(-3,5)$ is also optimal (Fig. 2) and could have been chosen as the efficient solution for this iteration. As the second solution corresponds to a different criterion vector, it will not be deleted from the feasible region and will be optimal at next iteration.

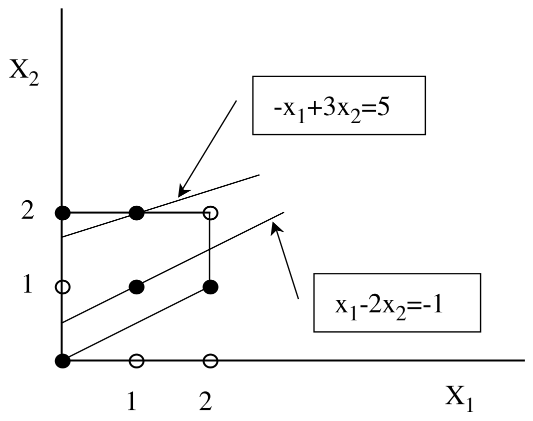

Fig. 3. Feasible region for $P1$.

$$
\begin{aligned}
(P_2)\ \max\quad & x_1+x_2\\
\text{s.t.}\quad & x_1-2x_2 \leqslant 0,\\
& x_1-2x_2 \geqslant -y_1^1-4(1-y_1^1),\\
& -x_1+3x_2 \geqslant 5y_2^1-2(1-y_2^1),\\
& x_1-2x_2 \geqslant y_1^2-4(1-y_1^2),\\
& -x_1+3x_2 \geqslant 2y_2^2-2(1-y_2^2),\\
& y_1^1+y_2^1 \geqslant 1,\\
& y_1^2+y_2^2 \geqslant 1,\\
& x_1,x_2 \in \{0,1,2\},\\
& y_1^1,y_2^1,y_1^2,y_2^2 \in \{0,1\}.
\end{aligned}
$$

The new constraints delete non-efficient point $(0,0)^{\mathrm{T}}$ as well as efficient point x^2. Notice that $(0,0)^{\mathrm{T}}$ is an optimal solution for the first objective function but it is dominated by x^2 (Fig. 4).

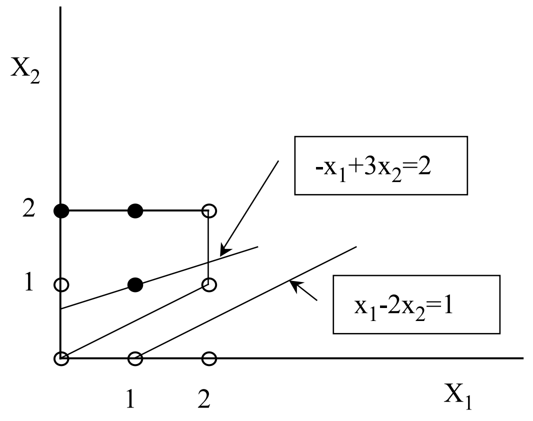

Fig. 4. Feasible region for $P2$.

An optimal solution to this problem is $x_1=1$, $x_2=2$, $y_1^1=y_1^2=0$, $y_2^1=y_2^2=1$ with $v(P_2)=3$. This corresponds to a new efficient point $x^3=(1,2)^{\mathrm{T}}$ with an objective function value vector equal to $(-3,5)^{\mathrm{T}}$.

The following step adds constraints that delete only the efficient point x^3:

$$
\begin{aligned}
(P_3)\ \max\quad & x_1+x_2\\
\text{s.t.}\quad & x_1-2x_2 \leqslant 0,\\
& x_1-2x_2 \geqslant -y_1^1-4(1-y_1^1),\\
& -x_1+3x_2 \geqslant 5y_2^1-2(1-y_2^1),\\
& x_1-2x_2 \geqslant y_1^2-4(1-y_1^2),\\
& -x_1+3x_2 \geqslant 2y_2^2-2(1-y_2^2),\\
& x_1-2x_2 \geqslant -2y_1^3-4(1-y_1^3),\\
& -x_1+3x_2 \geqslant 6y_2^3-2(1-y_2^3),\\
& y_1^s+y_2^s \geqslant 1;\quad s=1,2,3,\\
& x_1,x_2 \in \{0,1,2\},\\
& y_k^s \in \{0,1\};\quad k=1,2;\ s=1,2,3.
\end{aligned}
$$

An optimal solution to the last problem is $x_1=0$, $x_2=2$, $y_1^1=y_1^2=y_1^3=0$, $y_2^1=y_2^2=y_2^3=1$ with $v(P_3)=2$. This corresponds to an efficient solution $x^4=(0,2)^{\mathrm{T}}$ with an objective function value vector equal to $(-4,6)^{\mathrm{T}}$.

Now, problem (P_4) is defined:

$$
\begin{aligned}
(P_4)\ \max\quad & x_1+x_2\\
\text{s.t.}\quad & x_1-2x_2 \leqslant 0,\\
& x_1-2x_2 \geqslant -y_1^1-4(1-y_1^1),\\
& -x_1+3x_2 \geqslant 5y_2^1-2(1-y_2^1),\\
& x_1-2x_2 \geqslant y_1^2-4(1-y_1^2),\\
& -x_1+3x_2 \geqslant 2y_2^2-2(1-y_2^2),\\
& x_1-2x_2 \geqslant -2y_1^3-4(1-y_1^3),\\
& -x_1+3x_2 \geqslant 6y_2^3-2(1-y_2^3),\\
& x_1-2x_2 \geqslant -3y_1^4-4(1-y_1^4),\\
& -x_1+3x_2 \geqslant 7y_2^4-2(1-y_2^4),\\
& y_1^s+y_2^s \geqslant 1;\quad s=1,\dots,4,\\
& x_1,x_2 \in \{0,1,2\},\\
& y_k^s \in \{0,1\};\quad k=1,2;\ s=1,\dots,4.
\end{aligned}
$$

This problem has a single feasible point: $x_1 = x_2 = 1$, $y_1^1 = y_1^3 = y_1^4 = 1$, $y_1^2 = 0$, $y_2^1 = y_2^3 = y_2^4 = 0$, $y_2^2 = 1$ with an objective function value $v(P_4) = 2$. This indicates a new efficient point $x^4 = (1,1)^{\mathrm{T}}$ with an objective function value vector equal to $(-1,2)^{\mathrm{T}}$. Note that x^4 is a non-supported efficient solution (Fig. 5) as it is not optimal in $F(P)$ for any positive combination of the objective functions of problem (P).

The next ILP to be solved is

$$
\begin{aligned}
(P_5)\ \max\quad & x_1 + x_2 \\
\text{s.t.}\quad & x_1 - 2x_2 \leqslant 0, \\
& x_1 - 2x_2 \geqslant -y_1^1 - 4(1 - y_1^1), \\
& -x_1 + 3x_2 \geqslant 5y_2^1 - 2(1 - y_2^1), \\
& x_1 - 2x_2 \geqslant y_1^2 - 4(1 - y_1^2), \\
& -x_1 + 3x_2 \geqslant 2y_2^2 - 2(1 - y_2^2), \\
& x_1 - 2x_2 \geqslant -2y_1^3 - 4(1 - y_1^3), \\
& -x_1 + 3x_2 \geqslant 6y_2^3 - 2(1 - y_2^3), \\
& x_1 - 2x_2 \geqslant -3y_1^4 - 4(1 - y_1^4), \\
& -x_1 + 3x_2 \geqslant 7y_2^4 - 2(1 - y_2^4), \\
& x_1 - 2x_2 \geqslant -4(1 - y_1^5), \\
& -x_1 + 3x_2 \geqslant 3y_2^5 - 2(1 - y_2^5), \\
& y_1^s + y_2^s \geqslant 1; \quad s = 1, \ldots, 5, \\
& x_1, x_2 \in \{0, 1, 2\}, \\
& y_k^s \in \{0, 1\}; \quad k = 1, 2; \quad s = 1, \ldots, 5.
\end{aligned}
$$

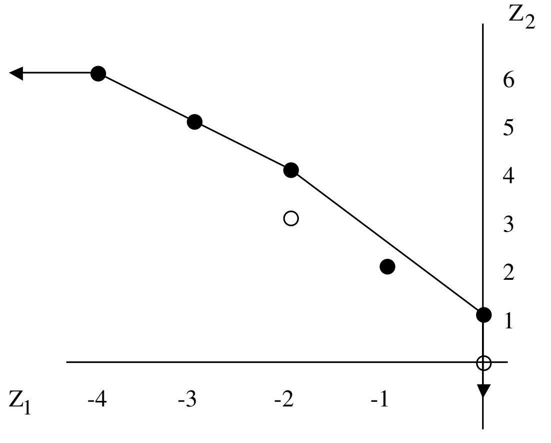

Fig. 5. Non-supported efficient vector $(-1,2)$.

Table 1
Solutions for problem (P)

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

As this problem is unfeasible, the process is complete and we have the complete set of non-dominated objective vectors as well as an efficient decision point for each of those (Table 1).

5. Computational results

The method was implemented in a C program using OSL Release 2.1 [11]. The integer linear problems were solved using the (general) routines of this library and the structure of the original integer multiobjective problem was not exploited. For the branching process, artificial variables y_k^l had a higher priority over the original x_j variables. The program was run on a RISC 6000/250 in a multiuser environment under AIX 3.02.

The method was tested with randomly generated multiconstrained 0–1 knapsack problems with two objectives functions. Objective function 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. Table 2 shows cumulative results for batches of 30 problems of different dimensions.

As seen in these results, the total number of nodes and simplex iterations grow quickly when the number of variables (n) is incremented. This is partially due to the fact that the number of integer problems (IPs) that must be solved is bigger when n is increased (for each problem the number of IPs is the number of non-dominated vectors plus one unfeasible problem). When the number of constraints (m) is incremented, there is also a larger amount of nodes and simplex iterations though the number of integer problems solved is not much more considerable.

The method was also tested with multiconstrained 0–1 knapsack problems with three objective functions. 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 small subset of well discriminated solutions was generated; this was accomplished by stipulating an increment of at least f_k units in some objective for any new solution generated. The value of f_k chosen for each MILP is proportional to the number of variables in order to get about 10% of the order of magnitude of the objective functions. Objective function 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. Results are shown in Table 3.

A third 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 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

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

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

Table 2
0–1 Knapsack problems with 2 objective functions

m	n	IPs		Nodes		Simplex iterations	
		Mean	Max	Mean	Max	Mean	Max
5	15	8.2	14	3586.2	17 036	7625.2	36 191
5	20	13.8	23	21 839.9	71 562	48 730.4	153 042
5	25	15.5	26	50 317.6	346 483	127 435.0	936 581
5	30	21.8	37	146 823.4	440 782	377 892.4	1 115 816
10	15	8.3	20	5448.0	47 920	13 177.2	116 644
10	20	14.3	31	47 269.7	350 537	127 575.2	929 335
10	25	17.6	29	64 527.1	208 777	186 130.5	597 954
10	30	21.0	33	224 967.5	840 638	702 280.9	2 698 297

Table 3
0–1 Knapsack problems with 3 objective functions

m	n	IPs		Nodes		Simplex iterations		f_k
		Mean	Max	Mean	Max	Mean	Max	
5	15	4.3	8	850.9	5139	2115.2	12 766	525
5	20	4.1	6	1051.2	3330	2815.9	9003	700
5	25	4.4	6	2000.2	5138	5704.5	14 945	875
5	30	4.1	7	1717.2	4678	4784.8	14 318	1050
10	15	4.1	7	1088.6	4664	3321.1	15 418	525
10	20	3.9	6	1532.0	4972	4988.1	14 874	700
10	25	4.3	7	2660.4	7910	9337.0	28 369	875
10	30	4.2	6	3268.0	9030	11 650.4	35 552	1050

Table 4
GAP problems with 2 objective functions

s	t	IPs		Nodes		Simplex iterations		f_k
		Mean	Max	Mean	Max	Mean	Max	
5	50	19.1	26	2080.4	4190	7831.5	14 536	500
5	60	18.7	22	2368.0	3815	9019.3	12 705	600
5	70	18.7	24	2899.9	5762	10 862.4	18 092	700
5	80	19.4	22	3717.3	6043	13 505.1	20 759	800
5	90	18.2	21	4090.0	5525	15 236.0	22 608	900
5	100	19.3	24	4749.6	8738	18 661.1	34 444	1000
10	50	20.0	24	3056.4	5298	12 690.2	18 510	500
10	60	20.7	23	3721.9	5042	16 829.9	23 380	600
10	70	20.9	26	4686.9	8112	21 597.5	45 421	700
10	80	20.5	26	5035.0	8121	22 040.6	36 889	800
10	90	21.0	24	5513.9	7664	25 114.0	43 086	900
10	100	20.9	24	6096.2	10177	29 190.0	44 831	1000
15	50	20.0	23	3427.4	5866	15 365.0	26 117	500
15	60	18.9	22	3610.3	6112	16 339.0	23 873	600
15	70	19.8	22	4078.9	6253	20 735.0	32 778	700
15	80	19.6	23	4805.0	7861	22 427.0	35 327	800
15	90	19.7	23	5401.9	7666	25 149.0	32 077	900
15	100	19.7	22	5814.7	7771	29 233.8	41 281	1000

corresponding r_{ij}. In this case, the value of f_k is proportional to the number of tasks, about 1% of the order of magnitude of the objective functions. Results are shown in Table 4.

Although the particular structure of GAP is not exploited, the method works better than in the knapsack problems as the number of nodes and simplex iterations is small compared with the number of solutions generated.

A fourth group of tests involves randomly generated multiconstrained 0–1–2 knapsack problems with two objectives functions. Objective function 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. In order to generate the complete set of non-dominated vectors, a value of $f_k = 1$ is used. Table 5 shows the results of these experiments.

6. An extension: Generating all efficient solutions to zero–one problems

As pointed out in previous sections, the algorithm can find all non-dominated objective vectors but will not necessarily generate all efficient decision vectors. Every time a new efficient solution x^* is found, the set of x such that $Cx \leqslant Cx^*$ is deleted from the feasible set, eliminating not only non-

Table 5
0–1–2 Knapsack problems with 2 objective functions

m	n	IPs		Nodes		Simplex iterations	
		Mean	Max	Mean	Max	Mean	Max
5	12	13.4	32	19 194.1	284 638	34 074.93	485 398
10	12	11.2	21	11 584.1	49 085	24 577.9	108 242
5	15	18.5	38	50 807.2	238 133	102 584.6	462 473
10	15	19.2	36	66 874.6	208 002	152 797.9	487 448
5	18	20.7	42	88 910.0	512 884	188 552.4	1 131 348
10	18	23.8	48	164 930.0	1 022 029	399 547.1	2 377 710

efficient decisions but also efficient decisions x such that $Cx = Cx^*$. Nevertheless, for each non-dominated objective vector, the method gives a representative element of an equivalence class of decisions. A complete enumeration of all decisions will require additional work, for example, the following procedure can be used for the specific case of zero–one decision variables.

For each efficient decision x^* found, define $K_0(x^*) = \{j : x_j^* = 0\}$ and $K_1(x^*) = \{j : x_j^* = 1\}$ and solve

$$\begin{aligned}(Q):\max\quad & 0\\ \text{s.t.}\quad & Ax = b,\\ & Cx = Cx^*,\\ & \sum_{j\in K_1(x^*)} x_j - \sum_{j\in K_0(x^*)} x_j \leqslant |K_1(x^*)| - 1,\\ & x \geqslant 0,\quad x\in Z^n.\end{aligned}$$

As the constraint $\sum_{j\in K_1(x^*)} x_j - \sum_{j\in K_0(x^*)} x_j \leqslant |K_1(x^*)| - 1$ is satisfied by all zero–one vectors except for x^*, a new efficient solution x^{**} will be found if it exists. Adding similar constraints ($\sum_{j\in K_1(x^{**})} x_j - \sum_{j\in K_0(x^{**})} x_j \leqslant |K_1(x^{**})| - 1$), and repeating the process until unfeasibility, all solutions x such that $Cx = Cx^*$ are found.

7. Future research

As currently implemented, the method can solve medium-sized problems with two objective functions. Nevertheless, its performance may not be satisfactory for problems with a larger number of objective functions. As these problems usually lead to a very large number of non-dominated vectors, a complete enumeration of these is out of the question; but in the worst of cases, a few representative solutions can still be generated.

There are several aspects of the implementation that can be improved. This includes, order of selection of the branching variables, detection of redundant constraints and exploitation of results of previously solved integer problems (currently, each new IP is solved from scratch).

Additional enhancements can be achieved customizing the method for specific problems (for example, Balas algorithm for zero–one problems as in [9]).

Although originally thought as an algorithm for finding all non-dominated vectors, this method has features that allow it to be used as a tool for more practical purposes. Using large values of f_k, it can generate a few well spaced solutions giving a DM a general view of his/her possibilities; on the other hand, small values of f_k (without running the process until completion) can be used to find alternatives close to an already-known good solution. This characteristics can make this algorithm a good starting point for designing interactive MCDM methods.

8. Conclusions

Most previous efforts for solving MOILP problems have focused either in theoretical approaches for finding the complete set of non-dominated solutions or in practical (frequently interactive) algorithms for finding some solutions for MCDM problems.

Using a simple, straightforward theoretical basis, this method can generate all non-dominated solutions for a MOILP with integer (not necessarily positive) coefficients and bounded (but not necessarily zero–one) integer decision variables. Its current computational implementation can solve fairly-sized problems using available Integer Programming libraries such as OSL. Furthermore, the method is flexible enough to give a partial (but useful) set of solutions in larger problems allowing it to handle system resource limitations and to be used as a building block for interactive Decision Making procedures.

References

[1] G.W. Bitran, Linear multiple objective programs with zero–one variables, Mathematical Programming 13 (1977) 121–139.

[2] G.W. Bitran, Theory and algorithms for linear multiple objective programs with zero–one variables, Mathematical Programming 17 (1979) 76–85.

[3] R.E. Burkard, H. Keiding, J. Krarup, P.M. Pruzan, A relationship between optimality and efficiency in

multicriteria 0–1 programming problems, Computers and Operations Research 8 (1981) 241–247.
[4] R.E. Burkard, J. Krarup, P.M. Pruzan, Efficiency and optimality in minisum, minimax 0–1 programming problems, Journal of the Operational Research Society 33 (1982) 137–151.
[5] L.G. Chamlet, L. Lemonidis, D.J. Elzinga, An algorithm for the bi-criterion integer programming problem, European Journal of Operational Research 25 (1986) 292–300.
[6] A. Crema, A contraction algorithm for the multiparametric integer linear programming problem, European Journal of Operational Research 101 (1997) 130–139.
[7] R.F. Deckro, E.P. Winkofsky, Solving zero–one multiple objective programs through implicit enumeration, European Journal of Operational Research 12 (1983) 362–374.
[8] G. Kiziltan, E. Yucaoglu, An algorithm for multiobjective zero–one linear programming, Management Science 29 (12) (1983) 1444–1453.
[9] D. Klein, E. Hannan, An algorithm for the multiple objective integer linear programming problem, European Journal of Operational Research 9 (4) (1982) 378–385.
[10] F. Liu, C. Huang, Y. Yen, Using DEA to obtain efficient solutions for multi-objective 0–1 linear programs, European Journal of Operational Research 126 (2000) 51–68.
[11] Optimization Subroutine Library Release 2.1, IBM, 1992.
[12] L.M. Rasmussen, Zero–one programming with multiple criteria, European Journal of Operational Research 26 (1986) 83–95.
[13] D. Schweigert, P. Neumayer, A reduction algorithm for integer multiple objective linear programs, European Journal of Operational Research 99 (1997) 459–462.
[14] R.M. Soland, The design of multiactivity multifacility systems, European Journal of Operational Research 12 (1983) 95–104.
[15] R.E. Steuer, Multiple Criteria Optimization—Theory, Computation and Application, John Wiley and Sons, 1986.
[16] E.L. Ulungu, J. Teghem, Multi-objective combinatorial optimization problems: A survey, Journal of Multi-Criteria Decision Analysis 3 (1994) 83–104.
[17] B. Villarreal, M.H. Karwan, Multicriteria integer programming: A (hybrid) dynamic programming recursive approach, Mathematical Programming 21 (1981) 204–223.
[18] J. Teghem Jr., P.L. Kunsch, A survey of techniques for finding efficient solutions to multi-objective integer linear programming, Asia-Pacific Journal of Operational Research 3 (1986) 95–108.