An Exact Method for Optimizing a Linear Function over an Integer efficient Set

FATMA ZOHRA OUAIL
USTHB
Faculty of Mathematics
Department of Operations Research
BP 32 EL ALIA-ALGIERS
ALGERIA
ouail.fatmazohra@yahoo.fr

MOHAMED EL-AMINE CHERGUI & MUSTAPHA MOULAI
USTHB
Faculty of Mathematics
Department of Operations Research
BP 32 EL ALIA-ALGIERS
ALGERIA
mchergui@usthb.dz & mmoulai@usthb.dz

Abstract: In this article, a new methodology is proposed to solve the problem of optimizing a linear function over an integer efficient set, noted (OI). The great challenges on it were its classification as NP-hard and that only three methods were introduced in the literature during decades treating this issue. We propose in this study a generalization of our method [8], wherein all efficient solutions of a multiobjective integer linear program (MOILP) are achieved. Based upon the well known branch and bound technique and strengthened by efficient tests, the proposed methodology succeeds to find an optimal solution in a finite number of steps. The main feature is that it greatly saturates nodes in the tree, thus a large number of feasible solutions can be avoided for optimality or efficiency purposes. Also, we have chosen Jorge method to perform a comparative study.

Key–Words: Multiobjective programming; integer efficient set; optimizing over efficient set

1 Introduction

Whereas the multiobjective integer linear programming problem has received much more attention than ever (see for instance in a chronological order: [18],[12], [13], [2], [3], [24], [21], [8], [4], [19], etc.), the field of optimization over the efficient set of a MOILP remains vacant and challenging to investigate. It takes the motivation from practice, where among all efficient solutions, whose number can be very large, the decision maker is interested only by one that optimizes a new criterion (generally different from those considered in the original problem). This problem with continuous variables has been studied by several authors. In fact, one of the pioneers in this field is Philip [22] who studied this problem and suggested for solving it two procedures. In the first one, he supposed that the function to be optimized is a strictly positive aggregation of the criteria, where the restriction "*efficient set*" is replaced by the "*decision space*" and it becomes an ordinary linear programing problem. In the second, the optimization is done according to any linear function and a cutting plane technique is used to overcome the difficulty of non convexity of the efficient set and removes uninteresting (efficient) points. Later, Benson [5], [6], [7], Ecker & Song [11] and others have also investigated this issue and have proposed different approaches. Notice that, in this case, the efficient set has two interesting properties. First, the efficient set is connected and second the efficient set contains extreme points of the polyhedron ([20]).

Nevertheless, the integer case, has not received as much attention as did the continuous case. This is due in fact, not only for the unknown structure of the feasible region (non convex) but also due to the discrete aspect of the decision variables.

Despite its importance in real life applications, a bibliographical research allowed us to conclude that only three methods were proposed dealing with this topic. First by Nguyen and *al.* [20], who proposed algorithms for two special cases. In the first case, he reduced the multi-objective problem to a bi-objective one and in the other, the function to be optimized is a non-negative combination of the criteria. Second, Abbas & *al.* [1], introduced for solving (OI) cutting planes in the decision space based on the work of Ecker & Song [11] in the continuous case. Whereas Jorge [11], defined a sequence of progressively more constrained single-objective integer problems that eliminates unacceptable points. Unlike the previous methods, Jorge's one was implemented and the author shows the obtained results over different problem instances randomly generated.

In the present study, we propose a branch and bound based method for problem OI. Based upon efficient cuts, our approach succeeds to find the optimal solution for OI without having to compute all efficient solutions. Also, our method was implemented using Matlab2013a and a comparative study is reported with Jorge's at the end.

2 Mathematical formulation

Assume that $r \geq 2$ is an integer and that c^i, $i = 1, \cdots, r$ are row vectors of $\mathbb{R}^n$. Let C be the $r \times n$ matrix formed by the vectors c^i; $i = 1, \cdots, r$ and let S be a nonempty, compact polyhedron in $\mathbb{R}^n$. Also, S is defined by $\{x \in \mathbb{R}^n | Ax \leq b, x \geq 0\}$, where A is an $m \times n$ matrix of integers; $b \in \mathbb{Z}^m$ and D is the set of integer solutions in S. Then, the multiple objective integer linear programming problem MOILP, described as:

$$(P) \begin{cases} Max \quad Z_i(x) = c^i x; \ i = 1, \cdots, r \\ s.t. \ \ x \in D = S \cap \mathbb{Z}^n \end{cases} \tag{1}$$

is considered as the problem of finding the set of all solutions that are efficient in the sense of the following definition:

Definition 1. *A solution $x \in D$ is known as efficient, if there does not exist another solution $y \in D$ such that $Cy \geq Cx$ with at least one strict inequality. Otherwise, x is not efficient and the vector Cy dominates the vector Cx.*

The image of an efficient solution in the criterion space is called commonly *non-dominated* solution or *Pareto optimal* solution. Let X_E denotes the set of all efficient solutions of program *(P)*.

Basically, the set of efficient solutions of (MOILP) can be very sizable and the task to choose one that fit the decision preferences is very difficult. This evolves finding a most preferred efficient point according to the mathematical programming problem:

$$(OI) \begin{cases} Max \quad \varphi(x) \\ s.t. \ \ x \in X_E \end{cases} \tag{2}$$

where $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a continuous linear function. φ is not necessarily a combination of the MOILP's criteria and can be any linear function.

Let us consider the following linear programming problem at stage l, $l \geq 0$ of the proposed method:

$$(OI_l) \begin{cases} Max \quad \varphi(x) \\ s.t. \ \ x \in S_l \end{cases} \tag{3}$$

We denote by x_{opt} the best efficient solution of OI found till step l and φ_{opt} its corresponding criterion value. $S_0 = S$, and S_{l+1} is obtained from S_l by adding efficient cuts described below. To do so, let x_l^* be the first integer solution obtained through solving (OI_l) by using, eventually, the branching process well known in branch and bound method.

Throughout the present paper, we will use the following notations:

- ○ By B_l (N_l), we mean the set index of basic (respectively non basic) variables of x_l^*;
- ○ Let $\bar{c}_j^i$ be the j^{th} component of the reduced cost vector c^i for each $i = 1, \cdots, r$ at the latest simplex tableau;
- ○ Using the concept of cuts is overriding in our methodology to solve OI. To do so, two types of cuts are built. In fact, to construct **cut of type I**, we define the set H_l as

$$H_l = \left\{ j \in N_l | \exists i = 1, \cdots, r; \ \bar{c}_j^i > 0 \right\} \cup$$
$$\left\{ j \in N_l | \bar{c}_j^i = 0 \ \forall i = 1, \cdots, r \right\}$$

  and the efficient cut

$$\sum_{j \in H_l} x_j \geq 1$$

  has the property of removing non efficient solutions without having to enumerate them. In the other hand, **cut of type II** is constructed according to the following inequality

$$\varphi(x) \geq \varphi_{opt}$$

  to allow removing uninteresting points regarding optimality.

- ○ We define the following two sets at node l of type I (at an integer solution):

$$S_{l+1}^1 = \left\{ x \in S_l | \sum_{j \in H_l} x_j \geq 1 \right\}$$

- ○ Also, the following set is considered at node l of type II (at non integer solution):

$$S_{l+1}^2 = \{ x \in S_l | \varphi(x) \geq \varphi_{opt} \}$$

- ○ $S_{l+1} = S_{l+1}^1 \cup S_{l+1}^2$
- ○ Eff_l is the set of potentially efficient solutions of MOILP obtained until step l. All solutions in Eff_l are feasible integer such as none of them dominates the others in the criterion space. The set Eff_l is updated, testing each time an integer solution x_l^* is reached, whether $Z(x_l^*)$ is dominated or not. If

$Z(x_l^*)$ is not dominated by any vector $Z(x)$, $x \in Eff_{l-1}$, then $Eff_l = Eff_{l-1} \cup \{x_l^*\}$ and removing also solutions from Eff_{l-1} whose criterion vector is dominated by $Z(x_l^*)$ or by $Z(y)$, y_l solution of $P(x_l^*)$.

So, we have to test the efficiency of the solutions via two options. The first one consists of considering the set Eff_l initially empty set and at each step l is updated. The second one concerns the resolution of the following mixed-integer program as it is reported in ([1], [17] and [20]):

Given a point $x_l^* \in D$, let $(P_{x_l^*})$ denotes the linear program:

$$(P_{x_l^*}) \begin{cases} Max\ e^t s \\ s.t.\ Cx = Is + Cx_l^*, \\ Ax \leq b,\ x \in D,\ s \geq 0 \end{cases} \quad (4)$$

where e is a vector column of ones and I is an identity matrix $(r \times r)$.

x_l^* is efficient if and only if $(P_{x_l^*})$ has a maximum value of zero. Otherwise (the maximum value of $(P_{x_l^*})$ is finite nonzero), the obtained solution is efficient. In this manner, obviously the first option is prior to the second to avoid solving at each step program (Px_l^*).

3 Methodology

The proposed algorithm generates the optimal solution of OI without having to enumerate X_E (recall that X_E is the efficient set of (P)). Based on branch and bound technique, the method is reinforced by efficient cuts and additional saturating tests allowing a smart search for the optimal solution. We start by solving the program (OI_l) defined by program 3 using the simplex method at step l of the algorithm (eventually dual simplex method). Then, to catch how the criteria vectors move from basis to basis, r lines are added to the basic simplex tableau and reduced costs are calculated in respect of the corresponding basis. If the obtained solution is non integer, then we still imposing integrity restrictions on the original variables of the program till getting integer ones. Once an integer solution x_l^* is achieved, new cuts are established and added to the current simplex tableau which allow reduce the search area considerably (containing non efficient and non interesting solutions for (OI)). We consider henceforth two types of nodes, those relative to branching process (type 1) and others to efficient cuts (type2). So, a node of type 2 is pruned if no improvement of the criteria can be done along the remaining domain or if an efficient solution is reached at a stage l. A node of type 1 is fathomed if φ_{opt} the best value of φ obtained till stage l is greater than or equal to value of φ at that node, even the corresponding solution is non integer or the domain becomes infeasible.

The algorithm is summarized as follows:

Step1 (initialization): Initialize the program index $l = 0$ and the optimal value of the objective function $\varphi_{opt} = -\infty$ to which it corresponds no optimal solution yet (x_{opt} unknown at the beginning), Eff_l the set of potentially efficient solutions of (P), $Eff_0 = \emptyset$.
Step2 (main step): While there is no saturated node in the tree search, solve (OI_l) using simplex or dual simplex method, it depends on the sign of the right hand side of the program. Go to ***Step3.1***.
Step3 (tests):

3.1 ***Feasibility test:*** If (OI_l) is infeasible, then stop and the node l is saturated, else, let x_l^* be the solution, if $\varphi_{opt} \geq \varphi(x_l^*)$, the node l is fathomed else, go to ***step3.2***;

3.2 ***Integrity test:*** If x_l^* is integer, update Eff_l and go to ***step3.3***, else go to ***step4***;

3.3 ***efficiency test:*** If x_l^* is not kept within Eff_l, x_l^* is not efficient and go to ***step5***, else, solve $(P_{x_l^*})$. If x_l^* is efficient then update eventually $\varphi_{opt} = \varphi(x_l^*)$ and $x_{opt} = x_l^*$; the node l is pruned since no improvement of φ further, else, let y_l be the solution of $(P_{x_l^*})$, update if necessary φ_{opt}, x_{opt} and the set Eff_l as well, go to ***step5***.

Step4 (branching): Choose one coordinate x_j of x_l^* such that $x_j = \alpha_j$, with α_j a fractional number. Then, split the program (OI_l) into two sub programs, by adding the constraints $x_j \leq \lfloor \alpha_j \rfloor$ to obtain (OI_{l_1}), $x_j \geq \lfloor \alpha_j \rfloor + 1$ and construct S_{l+1}^2 to obtain (OI_{l_2}) such that $l_1 > l+1$, $l_2 > l+1$ and $l_1 \neq l_2$, go to ***step2***. In fact, since the tree is treated according to the principle depth first, we add the cut $\varphi(x) \geq \varphi_{opt}$ in the second branch l_2.
Step5 (efficient cut): Construct the set H_l. If $H_l = \emptyset$; the node l is fathomed since no efficient solution exists afterward, otherwise, construct set S_{l+1}^1 (adding efficient cut), go to ***step2***.

4 Theoretical results

In order to justify the different steps of the proposed algorithm, the following results are established. We denote by D_l the set $D_l = S_l \cap \mathbb{Z}^n$.

Theorem 2. *Suppose that $H_l \neq \varnothing$ at the current integer solution x_l^*. If $x \neq x_l^*$ is an optimal solution of program OI in domain S_l, then $x \in S_{l+1}$.*

Proof: Let $x \neq x_l^*$ be an integer solution in domain S_l such that $x \notin S_{l+1}$, then $x \notin S_{l+1}^1 \vee x \notin S_{l+1}^2$.

- if $x \notin S_{l+1}^1$, then $x \in \left\{ x \in S_l | \sum\limits_{j \in N_l \setminus H_l} x_j \geq 1 \right\}$.
  Therefore, the coordinates of x check the following inequalities: $\sum\limits_{j \in H_l} x_j < 1$ and $\sum\limits_{j \in N_l \setminus H_l} x_j \geq 1$. It follows that $x_j = 0$ for all $j \in H_l$, and $x_j \geq 1$ for at least one index $j \in N_l \setminus H_l$. Using the simplex table in x_l^*, the following equality is supported by all criterion $i \in \{1, \cdots, r\}$:
  $c^i x = c^i x_l^* + \sum\limits_{j \in N_l} \widehat{c}_j^i x_j$
  $\Rightarrow c^i x = c^i x_l^* + \sum\limits_{j \in H_l} \widehat{c}_j^i x_j + \sum\limits_{j \in N_l \setminus H_l} \widehat{c}_j^i x_j$
  $\Rightarrow c^i x = c^i x_l^* + \sum\limits_{j \in N_l \setminus H_l} \widehat{c}_j^i x_j$
  Thus, $c^i x \leq c^i x_l^*$ for all criterion $i \in \{1, \cdots, r\}$, with $c^i x < c^i x_l^*$ for at least one criterion since $\widehat{c}_j^i \leq 0$ for all $j \in N_l \setminus H_l$.
  We conclude that solution x is not efficient and then, all efficient integer solutions belong to domain S_{l+1}^1, ...(*).
- If $x \notin S_{l+1}^2$, x is not optimal, contradiction, ...(**).

From (*) and (**), we conclude that $x \in S_{l+1}$. □

Theorem 3. *Let x_l^* be the current integer solution of program (OI_l), then if x_l^* is efficient for program (P), then it is an optimal solution of program OI over D_l.*

Proof: Suppose that x_l^* is not optimal for program OI. Then, $\exists x \in D_l,\ x \neq x_l^*$ such that $\varphi(x) > \varphi_{opt}$ from Theorem 2. However, x_l^* being efficient, which means that $\varphi_{opt} \geq \varphi(x_l^*)$. Thus $\varphi(x) \geq \varphi(x_l^*)$. In the other hand, at the current simplex tableau, the expression of φ can be written as:

$\varphi(x) = \varphi(x_l^*) + \sum\limits_{j \in N_l} \widehat{\varphi}_j x_j$
$\Rightarrow \varphi(x_l^*) + \sum\limits_{j \in N_l} \widehat{\varphi}_j x_j > \varphi(x_l^*)$
$\Rightarrow \sum\limits_{j \in N_l} \widehat{\varphi}_j x_j > 0$
which contradicts the fact that $\widehat{\varphi}_j \leq 0,\ \forall j \in N_l$.
□

Proposition 4. *If $H_l = \emptyset$, then $\forall x \in D_{l+1}$, x is not efficient.*

Proof: $H_l = \emptyset$, then $\forall i \in \{1, \cdots, r\}$, $\forall j \in N_l$, we have $\hat{c}_j^i \leq 0$ and $\exists i_0 \in \{1, \cdots, r\}$ such that $\hat{c}_j^{i_0} < 0$ $\forall j \in N_l$. So, x_l^* dominates all points x, $x \neq x_l^*$ of domain D_l. □

Proposition 5. *If $\varphi_{opt} \geq \varphi(x_l^*)$, then $\nexists x \in D_l$ such that $\varphi(x) > \varphi_{opt}$.*

Proof: It is obvious that all solutions x for which $\varphi(x) < \varphi_{opt}$ are not interesting even efficient, since the existence of an efficient solution giving already the best value of φ. □

Theorem 6. *The algorithm terminates in a finite number of iterations and returns the optimal solution of program OI.*

Proof: The set S of feasible solutions of program (P) being compact, it contains a finite number of integer solutions. At each step l of the algorithm, if an integer solution x_l^* is reached, we proceed to eliminate it as well as a subset of integer non interesting solutions by taking into account Theorem 2above (adding cuts). In the other hand, four saturating tests are used without loss of the optimal solution of OI. First, when the set H_l is empty the corresponding solution x_l^* is an ideal point and the current node can be pruned since no criterion can be improved. Secondly, if at a stage l, the current integer solution x_l^* is efficient, the corresponding node is fathomed since x_l^* is optimal for OI over D_l. Third, if φ_{opt} (value of the best efficient solution found for OI) is greater than that of the optimal solution over D_l, the node l also is fathomed. Finally, the trivial case when the reduced domain becomes infeasible. Hence, the algorithm converges toward an optimal solution for OI in finite number of steps. □

5 Illustrative example

Let us consider the following problem of optimizing a linear function over an integer efficient set, treated by Jesus M. Jorge in [17]:

$$(OI) \begin{cases} Max \ \ -x_1 - 2x_2 \\ x \in X_E \end{cases} \tag{5}$$

where X_E is the efficient set of the following program:

$$(P) \begin{cases} Max \ \ x_1 - 2x_2 \\ Max \ \ -x_1 + 4x_2 \\ s.t. \ \ -2x_1 + x_2 \leq 0 \\ \qquad x_1 \leq 3 \\ \qquad x_2 \leq 2 \\ x_1, x_2 \geq 0, \ integers \end{cases} \tag{6}$$

Step1: Set $\varphi_{opt} = -\infty$ and $l = 0$. After solving the program (OI_0), the optimal solution thus obtained is $x_0^* = (0,0)$ which is not efficient, but the optimal solution obtained from solving $(P_{x_0^*})$ which is $(3,1)$ is efficient. Update $\varphi_{opt} = -5$ and $x_{opt} = (3,1)$.

$H_0 = \{1, 2\} \neq \emptyset$.

	x_1	x_2	b_i
x_3	-2	1	0
x_4	1	0	3
x_5	0	1	2
$-\varphi$	-1	-2	0
$-c^1$	1	-2	0
$-c^2$	-1	4	0

Apply the efficient cut $x_1 + x_2 \geq 1$ and use the dual simplex technique to obtain the following tableau:

	x_6	x_2	b_i
x_3	-2	3	2
x_4	1	-1	2
x_5	0	1	2
x_1	-1	1	1
$-\varphi$	-1	-1	-1
$-c^1$	1	-3	1
$-c^2$	-1	5	-1

Solution $x_1^* = (1, 0)$ is obtained but it is not efficient (test of efficiency) and the obtained integer solution from solving (Px_1^*), $(3, 1)$ is efficient. $H_1 = \{6, 2\} \neq \emptyset$, apply the efficient cut $x_2 + x_6 \geq 1$ and dual simplex technique :

	x_7	x_3	b_i
x_6	$-\frac{3}{5}$	$-\frac{1}{5}$	$\frac{1}{5}$
x_4	$\frac{1}{5}$	$\frac{2}{5}$	$\frac{13}{5}$
x_5	$\frac{2}{5}$	$-\frac{1}{5}$	$\frac{6}{5}$
x_1	$-\frac{1}{5}$	$-\frac{2}{5}$	$\frac{2}{5}$
x_2	$-\frac{2}{5}$	$\frac{1}{5}$	$\frac{4}{5}$
$-\varphi$	-1	0	-2
$-c^1$	$-\frac{3}{5}$	$\frac{4}{5}$	$-\frac{6}{5}$
$-c^2$	$\frac{7}{5}$	$-\frac{6}{5}$	$\frac{14}{5}$

The optimal solution $x_2^* = (\frac{2}{5}, \frac{4}{5})$, we use branch and bound technique: $x_1 \leq 0$ or $x_1 \geq 1$:

- For $x_1 \leq 0$, program (OI) becomes infeasible;
- For $x_1 \geq 0$, the optimal solution found $x_3^* = (1, 1/2)$ is non integer:

	x_8	x_3	b_i
x_6	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
x_4	1	0	2
x_5	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
x_1	-1	0	1
x_2	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
x_3	$-\frac{5}{2}$	$\frac{1}{2}$	$\frac{3}{2}$
$-\varphi$	0	-1	-2
$-c^1$	2	-1	0
$-c^2$	-3	2	1

Two sub-programs are created (branching process): For $x_2 \leq 0$, the following tableau is obtained:

	x_7	x_9	b_i
x_6	-1	-1	1
x_4	1	2	1
x_5	0	-1	2
x_1	-1	-2	2
x_2	0	1	0
x_3	-2	-5	4
x_8	-1	-2	1
$-\varphi$	-1	0	-2
$-c^1$	1	4	2
$-c^2$	-1	-6	-2

The optimal solution is $x_4^* = (2, 0)$ which is integer and efficient (efficiency test), thus $\varphi_{opt} = -2$, $x_{opt} = (2, 0)$. For $x_2 \geq 1$, we have:

	x_9	x_8	b_i
x_6	-1	-1	1
x_4	0	1	2
x_5	1	0	1
x_1	0	-1	1
x_2	-1	0	1
x_3	1	-2	1
x_7	2	-1	1
$-\varphi$	-2	-1	-3
$-c^1$	-2	1	-1
$-c^2$	4	-1	3

The optimal solution is $x_5^* = (1, 1)$ and this node is fathomed because $\varphi_{opt} = -2$, $\varphi(x_5^*) = -3$ then $\varphi_{opt} > \varphi(x_5^*)$. Hence, the optimal solution for (OI) is: $x_{opt} = (2, 0)$ and $\varphi_{opt} = -2$.

6 Computational results and comparative study

The proposed method has been coded using MATLAB R2013a and run on a personal computer with 2.7 GHz

Core(TM) i7 CPU and 4GB of memory. We should notice that all subroutines were programed and no optimization packages were used. Furthermore, to test the efficiency of our algorithm, the method described in [16] (Jorge's method) was also programed using the same environment in order to compare performances of both of them.
To do so, we have chosen the third class of test problems provided by Gokhan Kirlik & Serpil Sayin in `//home.ku.edu.tr/~moolibrary/`. These test problems are the general multiobjective integer linear programming (MOILP) with m constraints, $m \in \{10, 15, 20\}$, n variables, $n = 2m$ and r objective functions , $r \in \{3, 4, 5\}$. Whereas the coefficients of function φ are generated randomly in [-100,100].

Instances	OCM Method	JORGE Method
ILP_r-3_n-20_m-10_ins-1	0.81	2.11
ILP_r-3_n-20_m-10_ins-2	0.48	2.883
ILP_r-3_n-20_m-10_ins-3	0.60	5.47
ILP_r-3_n-20_m-10_ins-4	1.79	8.10
ILP_r-3_n-20_m-10_ins-5	0.46	2.62
ILP_r-4_n-20_m-10_ins-1	0.153	1.02
ILP_r-4_n-20_m-10_ins-2	4.39	0.10
ILP_r-4_n-20_m-10_ins-3	0.11	0.27
ILP_r-4_n-20_m-10_ins-4	9.50	0.43
ILP_r-4_n-20_m-10_ins-5	5.09	4.55
ILP_r-5_n-20_m-10_ins-1	5.00	6.00
ILP_r-5_n-20_m-10_ins-2	0.221	0.505
ILP_r-5_n-20_m-10_ins-3	2.93	0.36
ILP_r-5_n-20_m-10_ins-4	0.14	0.22
ILP_r-5_n-20_m-10_ins-5	0.98	5.12
ILP_r-3_n-30_m-15_ins-1	22.71	616.24
ILP_r-3_n-30_m-15_ins-2	3.20	9.79
ILP_r-3_n-30_m-15_ins-3	10.99	92.20
ILP_r-3_n-30_m-15_ins-4	20.32	215.00
ILP_r-3_n-30_m-15_ins-5	2.33	17.291
ILP_r-4_n-30_m-15_ins-1	0.78	4.02
ILP_r-4_n-30_m-15_ins-2	6.65	4.67
ILP_r-4_n-30_m-15_ins-3	0.70	53.80
ILP_r-4_n-30_m-15_ins-4	17.75	553.40
ILP_r-4_n-30_m-15_ins-5	0.72	33.36
ILP_r-5_n-30_m-15_ins-1	0.54	21.06
ILP_r-5_n-30_m-15_ins-2	1.36	100.84
ILP_r-5_n-30_m-15_ins-3	4.13	302.95
ILP_r-5_n-30_m-15_ins-4	0.99	1.77
ILP_r-5_n-30_m-15_ins-5	18.64	306.00
ILP_r-3_n-40_m-20_ins-1	8.59	161.21
ILP_r-3_n-40_m-20_ins-2	37.83	413.45
ILP_r-4_n-40_m-20_ins-1	84.221	900.35
ILP_r-4_n-40_m-20_ins-2	9.40	1760.81
ILP_r-5_n-40_m-20_ins-1	9.78	48.91
ILP_r-5_n-40_m-20_ins-2	1.78	230.25

Table 1: CPU time (seconds)

As both methods have different architectures, we have chosen the CPU time elapsed in seconds to be the only metric to compare their performances. Table

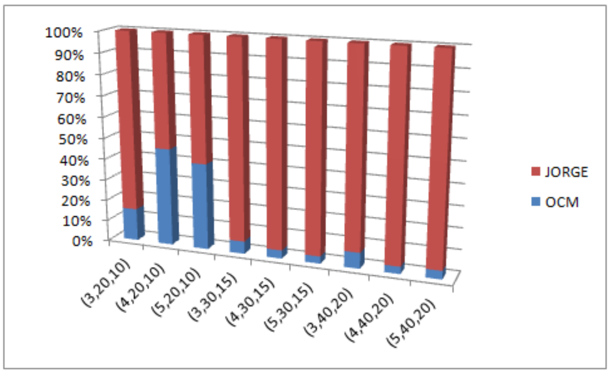

Figure 1: Histogram

1 summarizes the results obtained from experiencing them on several identical instances.

We can see that Jorge method outperforms our method on only 4 instances (7, 9, 13 and 22) of the 36 considered instances with minimum CPU deviation $0,07s$ and a maximum CPU deviation $4,29s$. However, our method is better on the remaining 32 instances with minimal CPU deviation $0,08s$ and a maximum CPU deviation equal to $1751,41s$. Furthermore, CPU time of our method is much more attractive than Jorges one. Also, our method has the advantage to be applied to problems with real objective functions coefficients while Jorge method can only be applied for integer objective functions coefficients as described by the author.

For a better display, the execution time for both method was represented by an histogram as a percentage Jorge/OCM %. The average is taken for each triplet (r, n, m).

7 Conclusion

In this paper, we proposed a branch and bound based method to optimize a linear function over the efficient set of a MOILP problem. Two types of cuts are used, the cuts of type 1 devoted to avoid the search in areas not containing efficient solutions and those of type 2 that delete domains not containing optimal solution. Reading the results of the experimentation shows that the proposed method outperforms 89 percent the Jorge's method, moreover, it has the advantage that it can be applied even if the coefficients of objective functions are real. Another advantage resides in the fact that our method can be extended to other global optimization problems with nonlinear objective functions, particularly for problems dealing with hyperbolic functions [9].

References:

[1] M. Abbas and D. Chaabane, Optimizing a linear function over an integer efficient set, *European Journal of Operational Research* 174, 2006, pp. 1140-1161.

[2] M. Abbas and M. Moulaï, Solving multiple objective integer linear programming, *Journal of the Italian Operations Research Society (Ricerca Operativa)* 29, 1999, pp. 15-38.

[3] M. Abbas and D. Chaabane, An algorithm for solving multiple objective integer linear programming problem, *RAIRO Operations Research* 36, 2002, pp. 351-364.

[4] M. Abbas, Mohamed El-Amine Chergui and Meriem Ait Mehdi, Efficient cuts for generating the non-dominated vectors for Multiple Objective Integer Linear Programming, *International Journal of Mathematics in Operational Research (IJMOR)*, Vol. 4, No. 3, 2012.

[5] H. Benson, Optimization over the efficient set, *Journal of Mathematical Analysis and Applications* 98, 1984, pp. 562-580.

[6] H. Benson, An algorithm for optimizing over the weakly efficient set, *European Journal of Operational Research* 25, 1986, pp. 192-199.

[7] H. Benson , A Bisection-Extreme Point Search Algorithm for Optimizing over the Efficient Set in the Linear Dependence Case, Discussion paper, 1992.

[8] M. E-A. Chergui , M. Moulaï and F.Z. Ouaïl, Solving the Multiple Objective Integer Linear Programming Problem, *Modeling, Computation and Optimization in Information Systems and Management Sciences Communications in Computer and Information Science* Volume 14, 2008, pp. 69-76, .

[9] M. E-A. Chergui and M. Moulaï, An exact method for a discrete multiobjective linear fractional optimization, Journal of Applied Mathematics and Decision Science, Vol. 2008, Article ID 760191, 12 pages.

[10] J, Ecker and I. Kouada, Finding efficient points for linear multiple objective programs, *Mathematical Programming* 8, 1975, pp. 375-377.

[11] J, Ecker and J.H. Song, Optimizing a linear function over an efficient set, *Journal of Optimization Theory and Application* Vol 83, , 1994, pp. 541-563.

[12] R. Gupta and R. Malhotra, Multi-criteria integer linear programming problem,*Cahiers de CERO* 34, , (1992), pp. 51-68.

[13] R. Gupta and R. Malhotra, Multi-criteria integer linear fractional programming problem, *Optimization* 35, 1995, pp. 373-389.

[14] Isermann, Computational experience concerning payoff tables and minimum criterion values over the efficient set, *European Journal of Operational Research* 33, 1987, pp. 91-97.

[15] H. Isermann, Proper efficiency and the linear vector maximization problem, *Operations Research* 22, 1974, pp. 189-191.

[16] J. Jorge, A bilinear algorithm for optimizing a linear function over the efficient set of a multiple objective linear programming problem, *Journal of Global Optimization* 31, 2005, pp. 1-16.

[17] J. Jorge, An algorithm for optimizing a linear function over an integer efficient set, *European Journal of Operational Research* 195, 2009, pp. 98-103.

[18] D. Klein and E. Hannan, An algorithm for multiple objective integer linear programming problem, *European Journal of Operational Research* 9, 1982, pp. 378-385.

[19] Kirlik, Gokhan and Sayin, Serpil, A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems, *European Journal of Operational Research*, 232, issue 3, 2014, pp. 479-488.

[20] N.C. Nguyen, An algorithm for optimizing a linear function over the integer efficient set, *Konrad-Zuse-Zentrum fur Informationstechnik Berlin*, 1992.

[21] M. Ozlen and M. Azizoglu, Multi-objective integer programming: A general approach for generating all non-dominated solutions, *presented at European Journal of Operational Research*, 2009, pp.25-35.

[22] J. Philip, Algorithm for the Vector Maximization Problem, *Mathematical Programming* 2, 1972, pp. 207-229.

[23] G.R. Reeves and R.C. Reid ,Minimum values over the efficient set in multiple objective decision making, *European Journal of Operational Research* 36, 1988, pp. 334-338.

[24] J. Sylva and A. Crema, A method for finding the set of non-dominated vectors for multiple objective integer linear programs, *European Journal of Operational Research, in press*, 2003.

[25] D.J. White ,The maximization of a function over the efficient set via a penalty function approach, *European Journal of Operational Research* 94, 1996, pp. 143-153.