Les Annales RECITS (http://www.lrecits.usthb.dz)
Vol. 06, 2019, pages 1-17
Reçu le 01 avril 2019
Accepté le 02 avril 2019

Solving the multiple-objective nonlinear convex integer programming problems: parallel processing general purpose

DIKES Abdellatif and CHERGUI Mohamed El-Amine

RECITS laboratory, USTHB, Bab Ezzouar, Algiers, Algeria

abdou_dik@hotmail.com - mohamedelaminec@yahoo.com

Abstract: Through this paper, we describe a branch and bound based method coupled to cutting plane method (C-P) for solving the multiobjective nonlinear convex integer programming problems. A C-P based linearization process will relax the nonlinear domain to a convex polyhedron, while branch and bound separation-cuts and our efficient-cuts will respectively, generate new partitions of the feasible domain and truncate some inefficient parts of this later. Multiobjective fathoming tools are used to avoid exploring non-promising nodes. We implemented a synchronous and an asynchronous processing-approaches, the last one shows much more efficiency than the first, almost on all generated instances, as reported in the experimental study.

Keywords: Multiobjective optimization, discrete programming, nonlinear convex optimization, efficient solutions, parallel processing.

Résumé : A travers ce manuscrit, nous décrivons une approche basée sur les pricipes "séparation et évaluation" et "approximations-extérieures", en vu de résoudre les problèmes d'optimisation multiobjectif non-linéaires convexes à variables entières. Le domaine non linéaire est relaxé en un polyèdre convexe, tandis que les coupes de séparation généreront de nouvelles partitions du domaine réalisable, notre coupe éfficace tronquera certaines parties non-éfficaces du domaine. Des outils de sondage multiobjectifs sont utilisés pour éviter d'explorer des nœuds non prometteurs. deux procédure à calcul synchrone et asynchrone ont été implémentées, la dernière est plus efficiente que la première sur toutes les instances générées, comme indiqué dans l'étude expérimentale.

Mots clés : Optimisation multiobjectif, programmation-mathématique discrète, optimisation non-linéaire convexe, solutions éfficace, calcul parallèle

1 Introduction

In many real world problems, among industry, economy, engineering or even policy sector, decision makers are usually confronted to satisfy (optimize) several conflicting criteria. Definitely, any engineering or industry problem is potentially linked to four objectives: risks, effectiveness, costs and efficiency, clearly each pair can be in conflict. Far from engineering and industry, precisely in economic field, F.Y.Edgeworth in 1881 defined the theory of multi-criterion optimality for economic problems. Later, V. Pareto, who was one of those firsts to analyze mathematically an economical problem, founded the theory of Pareto optimality [6]. The nineties of the last century mark the development of several applied mathematics methods for multiobjective optimization problems, which contributed in introducing the theory of Multi-Objective Optimization (MOO) to several other areas as industry or engineering.

Several classifications can house these methods, the most popular is the one depending on decision-maker preferences articulation moment during resolution process. The applicability of methods can also makes a classification object, generalized rigorous methods based on robust mathematical tools such as mathematical programming, offer more accuracy in terms of deliverables than methods based on heuristics, what certainly compromises their performances on difficult problems.

In the following, we will focus on generalized methods with posteriori-articulation of preferences, specifically those consisting of determining the complete efficient solutions set. Note that a solution is said to be efficient, if it is not possible to improve any one of relative criteria without altering at least another one. Clearly this definition is a Pareto theory derivative. For more details on the history of MOO, the reader is referred to [6], [14], [16].

Back to problems resolution-difficulty, some problems do not admit fractional solutions and should be solved on discrete sets. Moreover, non-linearity is commonly present in real world problems, it states physical, chemical, engineering, or even economical phenomena. The occurrence of variables integrity constraints or nonlinear functions in a problem mathematical program increases its difficulty. Fortunately, developments recorded last forty years in computer science fields, particularly in architecture optimizing, boosted materials and systems capacities in terms of processing and memory, thus prompted researchers and solution-developers, to tilt back on exact methods, aiming rationality enhancement on proposed solutions and decisions.

Several generalized approaches have been presented in the literature to solve the Multi-Objective Integer Linear Programming (MOILP) problems (bi-objective or treating r criteria, with $r \geq 2$), see e.g. [1], [5], [10], [15], [18] and [21]. Such approaches can be classified into: simplex approaches and interior point approaches. Also, we can differentiate decision space approaches from objectives space ones, these lasts, work on criteria space, such as approaches projecting Benson's one [2], or Kirlik & Sayin method [13].

Generalized methods for Nonlinear Programming problems, mainly Convex (NLCP), have also been the subject for decades of several papers: [9], [11] and [12].

Recognition to Dakin, who perceived in 1965 that the well-known branch & bound algorithm does not require functions linearity, therefore it can be generalized to NLCP with integer variables [3]. Then, solving methods dedicated to Nonlinear Convex Integer Programming problems (NLCIP) (also, with Mixed variables MINLCP), were designed, such as approaches: [7], [19]. In 1995, T. Westerlund and F. Pettersson extended the above-mentioned Kelley's CPM method [12] mainly developed for solving NLCP, to solve the MINLCP [22].

The common point between the methods presented to solve NLCP and NLCIP and those proposed for MOILP, is that both are reduced to solving linear sub-problems, which inspired us to use the same idea in solving the Multi-Objective Nonlinear Convex Integer Programming problems (MONLCIP).

Few generalized methods, dedicated to Multi-Objective Nonlinear Convex Programming problems (MONLCP) have been developed, a recent approach, being proposed by by M. Ehrgott & al. [8], extends an improved version [20] of the the Benson's method dedicated principally to MOLP [2]. This generalized approach for MONLCP consists in sandwiching the non-dominated set (method on criteria space) by internal and external approximations of the non-dominated front. This method offers a set of weakly non-dominated solutions.

Lately, V. Cacchiani & C. DAmbrsio [4] proposed a branch and bound based heuristic-approach to generate an approximated set of non-dominated points for Multi-Objective Nonlinear Convex Mixed-Integer Programming problems. The authors used two scalarization approaches to determine a set of weakly non-dominated solutions in two distinct stages. Indeed, the ϵ-constraint method is firstly applied to generate a starting set formed of some feasible solutions, then, weighted-sum method is used to reach more potential non-dominated solutions. A multiobjective branch and bound that compares upper and lower bounds is used to assure node fathoming.

By the end of 2017, another approach has been proposed by F.Z. Ouaïl & M.E.A. Chergui [17], it consist in a general-solving method to generate the whole efficient set for a subclass of the MONLCIP, precisely the Multi-Objective Quadratic Integer Programming problems (MOQIP), where quadratic objectives are optimized over a convex polyhedron.

To our knowledge, the MONLCIP problem hasn't received much attention from researchers. This motivated us to conceive an approach generating the whole efficient solutions set for this problem kind. The rest of the paper is organized as follows. In the next Section, we introduce some mathematical concepts and definitions used along this paper and we present the principle of the elaborated approach. Main theoretical results are formulated and proven in Section 3. In Section 4 a refined and optimized variant designed by introducing a purely algorithmic concept by separation and parallel processing is presented; the experimental implementation results of the designed approach and the variant, on randomly generated MONLCIP problem instances, are exposed and criticized. A general conclusion is given in Section 5.

2 The MONLCIP elaborated solving-approach

2.1 Definitions and notations

First of all, the following sets and notations are defined:

$$V = \{x \in \mathbb{R}^n | Ax \leq b;\ 0 \leq x \leq Ub\}$$

where A is $p \times n$ integer matrix, b is p-integer vector and $Ub \in \mathbb{N}^n$ is the upper bounds vector. We assume that V is a non-empty, compact polyhedron set.

$$S = \{\ x \in \mathbb{R}^n \mid g(x) \leq 0;\ g(x) = (g_i(x))_{i=\overline{1,q}}\}$$

where, the vector function g is convex. Also, $g_i(x)\ \forall i = \overline{1,q}$, are differentiable functions on $\mathbb{R}^n$.

We consider also the set: $U = V \cap S$. Note that U is a compact convex set of $\mathbb{R}^n$ by construction.

The mathematical formulation of the multiobjective problem under study, is summarized as follows:

$$(P) \begin{cases} Max\ Cx \\ x \in U \cap \mathbb{N}^n \end{cases}$$

where $Cx = (C_m x)_{m=\overline{1,r}}$ and ${C_m}^t$ is the m^{th} criterion vector (${C_m}^t \in \mathbb{R}^n$).

In the following, we will assume that:

$$\|\nabla g_i(x)\| \leq \varphi,\ \forall x \in \mathbb{R}^n \tag{1}$$

with $\varphi \in \mathbb{R}^+$. Thus, g_i is finite valued $\forall\ i = \overline{1,q}$.

Since g_i are convex differentiable functions $\forall i = \overline{1,q}$, their supporting hyper-planes at any point $x^0 \in V$, can be substituted with second order Taylor's developments at the same point, that we note $Pg_i(x, x^0)$:

$$Pg_i(x, x^0) = \nabla g_i(x^0) \cdot (x - x^0) + g_i(x^0) \leq g_i(x)\ \ \forall i = \overline{1,q}$$

For each $x^0 \in V$, for $\hat{J} = \{i \in \{1, \ldots,\ q\} \setminus g_i(x^0) > 0\}$, we define the following set:

$$PG^0 = \bigcap_{i \in \hat{J}} \{x \in \mathbb{R}^n \setminus Pg_i(x, x^0) \leq 0\} \tag{2}$$

Definition 1 *A feasible solution x is said to be efficient for (P) if, and only if, $\nexists\ y \in U \cap \mathbb{N}^n$ such that $C_m y \geq C_m x\ \forall m = \overline{1,r}$, and $C_m y > C_m x$ for at least one index m. If such y exists, then we say that x is not efficient and Cx is dominated by Cy.*

Definition 2 *If we ignore the condition that $C_m y > C_m x$ for at least one index $m \in \{1, \ldots, r\}$ from **Definition 1**, then we speak about weakly-efficiency and weakly-dominance of x and Cx respectively.*

2.2 Principle of the approach

The elaborated approach called ***CIMOES***, is a MONLCIP general exact solving method. This approach allows to derive the whole set of efficient solutions for problem (P), denoted *Eff*.

In order to describe our branch and bound based approach using the simplex method, the following notations are also used throughout the paper.

Let f be the aggregation of the vector functions C defined as:

$f(x) = \sum_{m=1}^{r} \lambda_m (C_m x)$ where, $0 < \lambda_m < 1 \ \forall m = \overline{1,r}$ and $\sum_{m=1}^{r} \lambda_m = 1$.

At a stage l of ***CIMOES***, for x^{*l} an integer feasible solution of (P) and for N_l the set of the indexes of non-basic variables, obtained from the current simplex tableau, we derive Δ_l as the set of all criteria's potential improving directions:

$$\Delta_l = \left\{ j \in N_l | \exists m \in \{1, \ldots, r\} ; \ \hat{c}_m^j > 0 \right\} \cup \left\{ j \in N_l | \hat{c}_m^j = 0, \ \forall m \in \{1, \ldots, r\} \right\} \quad (3)$$

where $\hat{c}_m^j$ represents the final update of the j^{th} reduced cost of the m^{th} criterion from simplex tableau at stage l.

Let consider the efficient cut EEC:

$$\sum_{j \in \Delta_l} x_j \geq 1 \quad (4)$$

We consider also, $I^l = \left(I_m^l\right)_{m=\overline{1,r}}$ where, $I_m^l = max \{C_m x \backslash x \in V_l\}$ $m \in \{1, \ldots, r\}$, refers to the m^{th} coordinate of the ideal point I^l relatively to a subset V_l of V, $(\forall \ l \geq 0)$.

Starting from an extreme point of V, cutting plane method [12] is used to solve (PR) obtained from (P) by substituting $f(x)$ for Cx and relaxing variables integrity constraints:

$$(PR) \begin{cases} Max \ f(x) \\ x \in U \end{cases}$$

Let $\hat{x} \in U$ be an optimal solution of (PR). If $\hat{x} \in \mathbb{N}^n$, we introduce the efficient cut EEC written above in (4), at $x^{*l} = \hat{x}$. This has the effect to delete current integer solution x^{*l}, as well as a part of domain U not containing any efficient solution.

If $\hat{x} \notin \mathbb{N}^n$, let $\hat{x}^1 \in \mathbb{N}^n$ attained after optimizing the problem (PRL) by a branch and bound approach combined to the well-known simplex method, where (PRL) is a generic to final (PR) obtained by substituting U with only current final V noted V_l (without considering S) and reinserting variables integrity constraints.

$$(PRL) \begin{cases} Max \ f(x) \\ x \in V_l \cap \mathbb{N}^n \end{cases}$$

By construction, $\hat{x}^1$ is in V but not necessarily in U. For that, intermediate sequential calls to the cutting plane method combined with branch and bound, will insure convergence

to a point $x^{*l} \in U \cap \mathbb{N}^n$. Then we add the efficient cut EEC at x^{*l}.Note that, we use I^l at each stage l of ***CIMOES***, in order to provide a fathoming process to the branch and bound method.

On the stage $l+1$, we define the sets V_{l+1} a relaxed set obtained from V_l if $\Delta_l \neq \emptyset$, D_{l+1} and W_{l+1} as flows:

$$V_{l+1} = \left\{ x \in V_l \mid \sum_{j \in \Delta_l} x_j \geq 1 \right\}, \; V_0 = V \tag{5}$$

$$D_{l+1} = V_{l+1} \cap \mathbb{N}^n \tag{6}$$

$$W_{l+1} = \left\{ x \in V_l \mid \sum_{j \in N_l \setminus \Delta_l} x_j \geq 1 \right\} \cap \mathbb{N}^n, \; l \geq 0 \tag{7}$$

V is more reduced each time sections (efficient cut EEC or cutting planes forming PG^k sets defined above in (2)) are added. If required, the dual-simplex method is used to re-optimize f on the new obtained domain V, after adding such sections. Moreover, all the r criteria of basic problem (P) are updated at each simplex step.

In order to avoid exploring non-promising sub-domains, some multiobjective branch and bound fathoming rules are used. Fathoming process consists in comparing the current l node's upper bounds vector or ideal point relatively to current domain V_l, to all current lower bounds vectors, witch are the image on the criteria space, of the potential-efficient solutions set Eff at stage l (that we can abusively mention $C(Eff)$).

It should also be noted that we consider only linear criteria since, a simple transformation operated on (P) for problems with nonlinear criteria, allows to replace any nonlinear criterion by a linear one. Indeed, substituting the nonlinear criterion with an auxiliary variable (let note it ψ), and the translation of this criterion into a constraint with the mathematical added-artifice ψ as a lower bound (for maximization case), makes deflecting criteria non-linearity possible.

3 Theoretical results

By the following two lemmas and the first theorem, ***CIMOES***'s (PR) optimization-step convergence will be proven. Demonstration is inspired from Kelley's one stated in its Cutting Plane Method [12].

Lemma 1 *Let $x^k \in V$ and $x^k \notin S$:*

For all $i \in \{1, \ldots, q\}$ such that $g_i\left(x^k\right) > 0$ (only inactive constraints by x^k): x^k and domain generated by the constraint $g_i(x) \leq 0$, $\forall x \in \mathbb{R}^n$, are on opposite sides of the support hyper-plane $Pg_i\left(x, x^k\right) = 0$.

Proof. Let $x^k \in V$ *and* $x^k \notin S,\ (k \geq 0),\ x \in \mathbb{R}^n$.

Since $Pg_i\left(x, x^k\right) \leq g_i\left(x\right),\ \forall i = \overline{1,q}$. Then, $g_i\left(x\right) \leq 0 \Longrightarrow Pg_i\left(x, x^k\right) \leq 0$. On the other hand, $\forall i = \overline{1,q}$, if $g_i\left(x^k\right) > 0$, then $Pg_i\left(x^k, x^k\right) > 0$ because:

$$Pg_i\left(x^k, x^k\right) = \nabla g_i\left(x^k\right) \cdot \left(x^k - x^k\right) + g_i\left(x^k\right). \quad \blacksquare$$

Let us define the two sequences $\{V_k\}_{k\geq 0}$, and $\{x^k\}_{k\geq 0}$ respectively as follows:

$$\begin{cases} V_0 = V_l \\ V_k = V_{k-1} \cap PG^{k-1} \end{cases} \qquad \begin{cases} x^0 \in V_0,\ x^0 \notin S \\ x^k := arg(max\left\{f(x) \setminus x \in V_k\right\}) \end{cases}$$

Lemma 2 *If* $PG^{k-1} \neq \emptyset$ *then,* $x^{k-1} \notin V_k$ *Otherwise,* $x^{k-1} \in S$.

Proof. According to **Lemma 1**, If $PG^{k-1} \neq \emptyset$ then, $x^{k-1} \notin PG^{k-1}$ therefore, $x^{k-1} \notin V_k$.

On the other hand, $PG^{k-1} = \emptyset$, signifies that: $\nexists\ i \in \{1, \ldots, q\} /\ g_i\left(x^{k-1}\right) > 0$. Thus, all S constraints are active by x^{k-1}, leading to $x^{k-1} \in S$. $\blacksquare$

Theorem 3 *The sequence* $\{x^k\}_{k\geq 0}$ *converges to* $\hat{x}$.

Where: $\hat{x} = arg\left(max\left\{f(x) \setminus x \in U_l\right\}\right)$ *for* $U_l = V_l \cap S$, $l > 0$.

Moreover, no efficient solution of (P) *is skipped during* $\hat{x}$ *determination.*

Proof.

(a) The sequence $\{x^k\}_{k\geq 0}$ is defined on the V compact block. Any sequence defined on a compact is a Cauchy sequence, according to the well-known Bolzano Weierstrass theorem.

On the other side, while $\forall k \geq 0$, $V_k = V_{k-1} \cap PG^{k-1}$, and according to **Lemma 2** : $V_k \subset V_{k-1} \subset V_{k-2} \subset \cdots \subset V_0$.

Moreover, for $x^k \in V_k$, we certainly have that: $\forall j = \overline{0, k-1},\ x^k \in V_j$. What will involve:

$$\nabla g\left(x^j\right) \cdot \left(x^k - x^j\right) + g\left(x^j\right) \leq 0,\ \forall j = \overline{0, k-1} \tag{8}$$

Let also define according to k, two other sequences $\{g(x^k)\}_{k\geq 0}$ and $\{f(x^k)\}_{k\geq 0}$ it's clear that:

$$Lim_{k\to\infty}\left\{x^k\right\} = \hat{x} \Longleftrightarrow \begin{cases} Lim_{k\to\infty}\left\{g(x^k)\right\} \leq 0 \\ Lim_{k\to\infty}\left\{f(x^k)\right\} = f(\hat{x}) \end{cases}$$

Firstly, suppose that $\{x^k\}_{k\geq 0}$ does not admit a subset that converges to an S belonging point, this is alike saying: there exist independently from k, at least a t $(t > 0)$ such that, there exist at least one i_0, with $i_0 \in \{1, \ldots, q\}$, satisfying:

$$g_{i_0}\left(x^k\right) > t.$$

(*b*) $g_{i_0}\left(x^k\right) > t > 0,\ \forall k \geq 0 \Longrightarrow g_{i_0}\left(x^j\right) > t > 0,\ \forall j = \overline{0, k-1}$

And, from inequality (8) we have that:

$\nabla g\left(x^j\right) \cdot \left(x^j - x^k\right) \geq g\left(x^j\right),\ \forall j = \overline{0, k-1}$. In particular, we have:

$\nabla g_{i_0}\left(x^j\right) \cdot \left(x^j - x^k\right) \geq g_{i_0}\left(x^j\right) > t > 0,\ \forall j = \overline{0, k-1}$. Thus,

$$\left\|\nabla g_{i_0}\left(x^j\right)\right\| \cdot \left\|\left(x^j - x^k\right)\right\| > t,\ \forall k \geq 0,\ \forall j = \overline{0, k-1} \tag{9}$$

According to expression defined at (1), $\forall x^0 \in V,\ \forall i = \overline{1, q}$,

$$\left\|\nabla g_i\left(x^0\right)\right\| \leq \varphi \Longrightarrow \left\|\nabla g_{i_0}\left(x^j\right)\right\| \cdot \left\|\left(x^j - x^k\right)\right\| \leq \varphi \cdot \left\|\left(x^j - x^k\right)\right\| \tag{10}$$

From (9) and (10) we conclude that $\forall j = \overline{0, k-1},\ k \geq 0 : \left\|\left(x^j - x^k\right)\right\| > t/\varphi > 0$.

However, saying that $\{x^k\}_{k\geq 0}$ is not a Cauchy sequence contradicts the fact that, $\{x^k\}_{k\geq 0}$ is defined on a compact set. Therefore, $\forall k \geq 0$ and, $\forall i_0 \in \{1, \ldots, q\}$:

$$\nexists\ t\ (t > 0),\ \textit{such that}\ g_{i_0}\left(x^k\right) > t \tag{11}$$

(*c*) From (11), $\{x^k\}_{k\geq 0}$ converges to an S belonging point.

Secondly, while $U_l \subset V_l$ ($V_l = V_k$) and $x^k = arg\left(max\left\{f\left(x\right)\ \backslash\ x \in V_k\right\}\right)$, implies that:

$x^k = arg\left(max\left\{f\left(x\right)\ \backslash\ x \in U_l\right\}\right) \Longrightarrow \{f(x^k)\}_{k\geq 0}$ converges to $f(\hat{x})$.

We can conclude that $\{x^k\}_{k\geq 0}$ converges to $\hat{x}$.

Furthermore, when relaxing any nonlinear convex domain (**i.q.** S) to a polyhedron formed by substituting all nonlinear functions with theirs supporting hyper-planes (of form like PG), it's clear by definition that no feasible solutions of the nonlinear domain can be avoided (all $x \in S$ are well-preserved).

Moreover, since $U \subsetneq S$, all $x \in U$ are preserved during linearization process; transitively, no efficient solution of (P) can be skipped by adding PG cuts. ■

First of all, and since $U_l \subset V_l$ we can conclude that all results stated below for V_l are transitively valid for U_l, whatever stage l of ***CIMOES***.

For x^{*l} the integer optimal solution to a current (PRL), clearly $x^{*l} \in D_l$. Next results guarantee that no effcient solution is skipped at the next stage $l+1$.

Lemma 4 $D_l \backslash \left\{x^{*l}\right\} = D_{l+1} \cup W_{l+1}$.

Where D_{l+1} and W_{l+1} are defined respectively in (6) and (7).

Proof. First, it's clear by definition that $D_{l+1} \cup W_{l+1} \subset D_l \backslash \left\{x^{*l}\right\}$.

Let $x \in D_l$, $x \neq x^{*l}$ then, x is in the closed domain generated by the Dantzig cut:

$$\sum_{j\in N_l} x_j \geq 1 \Longleftrightarrow \sum_{j\in \Delta_l} x_j + \sum_{j\in N_l\backslash\Delta_l} x_j \geq 1$$

as the sets Δ_l and $N_l\backslash\Delta_l$ define a partition of set N_l.

If the solution x satisfies the inequality $\sum_{j\in\Delta_l} x_j \geq 1$ then $x \in D_{l+1}$. Otherwise, second inequality $\sum_{j\in\Delta_l\backslash N_l} x_j \geq 1$ is satisfied by x, so $x \in W_{l+1}$. Consequently $x \in D_{l+1} \cup W_{l+1}$, what implies that $D_l \backslash \left\{x^{*l}\right\} \subset D_{l+1} \cup W_{l+1}$.

We can conclude that $D_l \backslash \left\{x^{*l}\right\} = D_{l+1} \cup W_{l+1}$. ■

Corollary 5 *Let $x \neq x^{*l}$ be an efficient solution in domain D_l, then x is located in D_{l+1}.*

Proof. Let $x \in D_l$, $x \neq x^{*l}$, suppose that $x \notin D_{l+1}$, from **Lemma 4** $x \in W_{l+1}$. Accordingly, following inequalities are active by x coordinates:

$$\sum_{j\in\Delta_l} x_j < 1 \; ; \quad \sum_{j\in N_l\backslash\Delta_l} x_j \geq 1$$

While all D_l variables are positive integers, it will be equivalent to say:

$\forall j \in \Delta_l \; x_j = 0$ and $x_j \geq 1$ for at least one index $j \in N_l\backslash\Delta_l$. By using the simplex tableau of (PRL) in x^{*l} the following equations hold for all criteria $m \in \{1, \ldots, r\}$,

$$C_m x = C_m x^{*l} + \sum_{j\in N_l} \hat{c}_m^j x_j$$
$$\Longrightarrow C_m x = C_m x^{*l} + \sum_{j\in\Delta_l} \hat{c}_m^j x_j + \sum_{j\in N_l\backslash\Delta_l} \hat{c}_m^j x_j$$

$$\Longrightarrow C_m x = C_m x^{*l} + \sum_{j\in N_l\backslash\Delta_l} \hat{c}_m^j x_j \tag{12}$$

Moreover, for all $j \in N_l\backslash\Delta_l$ we have $\forall m = \overline{1,r}\ \hat{c}_m^j \leq 0$ (for a maximization problem), with at least one strict inequality. Consequently, $\forall m = \overline{1,r}\ :\ C_m x \leq C_m x^{*l}$, with at least one strict inequality.

We conclude that x isn't efficient, thus, all integer efficient solutions (other then x^{*l}), belonging to D_l belong to D_{l+1}. ■

Proposition 6 *Used fathoming rules are "effective" to avoid exploring some unpromising parts of domain U.*

Note that, fathoming "effectiveness" is not only about deleting some nodes of the branch and bound tree, but it's about a certainty that, used rules will allow to fathom only uninteresting tree-nodes (non containing any efficient solution).

Proof. Problem (P) resolution will be significant only if $Eff \subsetneq U$, accordingly, there will be certainly some sub-domains of U not containing any efficient solutions.

From another side, since V_l is a relaxation of U_l ($U_l \subset V_l$) we can say that, the V_l ideal point I^l, weakly-dominates U_l ideal point. In view of that, if $\exists\ y \in Eff$ such that Cy dominates I^l (y is an existing potentially-efficient point) we can conclude that $\forall x \in U_l$ Cy dominates Cx. As a result, all U_l (transitively $U_l \cap \mathbb{N}^n$) elements cannot be efficient. ■

Theorem 7 *CIMOES generates the whole set of efficient solutions for problems of the form:*

$$(P)\begin{cases} Max\ Cx \\ x \in U \cap \mathbb{N}^n \end{cases}$$

and converges in a finite number of iterations.

Proof. According to ***Theorem 3***, by the end of the optimization of the relaxed problem (PR) corresponding to V_l, an optimal solution $\hat{x} \in U_l$ is given. Moreover, no efficient solution can be removed from domain U_l.

Each time an integer solution $x^{*l} \in D_l$ is generated, and once potential-efficiency test achieved, a set containing at least x^{*l} and eventually some non-efficient solutions, is eliminated from the new domain D_{l+1} according to ***Lemma 4*** and ***Corollary 5*** respectively. Therefore, new domains V_{l+1} and U_{l+1} are engendered since, $D_l \subset V_l$ and $U_{l+1} = S \cap V_{l+1}$.

At a stage l, $\Delta_l = \emptyset$ means that no improvement of any criterion is possible at x^{*l}. Indeed all criteria are non-increasing in feasible directions N_l. It follows that, Cx^{*l} dominates ideal point of current domain V_l and transitively U_l ideal point since, $U_l \subsetneq V_l$, consequently $U_l \cap \mathbb{N}^n \backslash \left\{x^{*l}\right\}$ no longer contains efficient solutions. (see proof of ***Corollary 5*** from equation (12)).

Moreover, if $M_p = \emptyset$, the entire starting domain U has been swept (every partition was definitely explored or fathomed).

Finaly, While U is a compact constrained set (intersection of the two compact sets S and V), U contains a finite number of integer solutions, which means that ***CIMOES*** converges in a finite number of iterations. ■

4 Implementation and numerical experimentation

4.1 Parallel processing

We observed during ***CIMOES*** execution on most generated instances, that CPU utilization ratio never conquered forty percent (40%) of its computing capacity. Therefore, we thought about parallelizing some ***CIMOES***'s blocks execution, what would certainly improve CPU exploitation, besides procedure treating level.

Indeed, the execution time of a sequential procedure parallelized on an N-core processor, is theoretically divided by N. *"An efficient parallel architecture is expensive, it must be used well."*

A block of sequential instructions (tasks) is parallelizable on a multiprocessor, if it's separable into inter-independent subtasks, or even inter-dependent subtasks if dependencies are manageable. Managing dependency involves information exchange between **processes** (subtasks), therefore exchanging information mechanisms must be put in place to particularly manage intervention on data and their sharing. In practice, depending on security needs (data security), memory can be distributed on processes so the access to data is restricted; however, exceptional inter-Processes communications can be developed. Otherwise, memory is shared between processes then, the access to data must be synchronized by input/output mutual exclusion.

Furthermore, Parallelism can be static by initializing a parallelized structure, or dynamic during program execution. In a static parallelism case, with a distributed memory, several system of processes-animation and information exchange are conceivable:

- Point-to-point transfer (Data transfer by Pipes).

- Broadcast (Data diffusion by Queues).

- Gather & scatter system.

In the last model, "Gather" operation permits collecting information from all processes in execution, while "Scatter" operation guarantees distributing collected information uniformly over all processes.

Our approach can be discretized into three distinct stages:

1. **Generating** sub-domains on the branch tree and stocking created nodes. This is done in the fifth (05) and sixth (06) steps of ***CIMOES***.

2. Then comes the optimization of those generated sub-domains by the simplex, or the **Evaluation**, realized in the third (03) ***CIMOES***'s step.

3. Finally, proceeding by intermediates sounding after pairwise comparisons, we eliminate unpromising sub-domains. Next to each generating step, "*node fathoming*" procedure execution insures the **Fathoming** process.

For an efficient and deep exercise of the third process, executed on a reference node, **Fathoming** process will require many information about others tree nodes. On the other hand, above-mentioned **Generating** and **Evaluating** processes, are less demanding in terms of required information about the others nodes. Moreover, the sequential execution of ***CIMOES*** does not offer enough visibility in terms of information about stocked and not yet explored nodes.

For these reasons, a parallelization of the first two processes would be rational and beneficial for diversifying research on exploration area, but only if it is controlled by creating exchanging windows between parallel **processes** in order to swap fathoming information.

Parallelized procedure is initialized under the single program multiple data model, the starting domain is subdivided into N sub-domains thus, N parallel **processes** are executed: **CIMOES** $\left(Data_i : \ i = \overline{1, N}\right)$. Where N represents the number of existing cores on used processor. Obviously, to control research diversifying on each local exploration area, *"node fathoming"* procedure is locally executed on each **Process** of the N Parallels.

After a carefully chosen number of iterations, Data stocked on M_{p_i} $(i = \overline{1, N})$ are consolidated to a merged M_p, also, after collecting and treating fathoming information, consolidated Data on M_p are checked according to updated fathoming information. Then, Data on M_p are randomly (which involve more diversification on research area) redistributed on N new equivalents Data-subdivisions (or N clusters). Finally, N new parallel **processes** are executed and updated fathoming information are reloaded for all **processes**.

We can easily distinguish in our animation, the **"gather"** and **"scatter"** mechanisms of the above-mentioned distributed memory animation system. Information are not directly exchanged between **processes** but **gathered** and consolidated; then **scattered** on N new processes. Clearly two different types of information are retained:

(*a*) M_{p_i} $(i = \overline{1, N})$, each M_{p_i} contains stocked nodes, corresponding to the **process** i. N clusters can be consolidated in the **gathering** step.

(*b*) Fathoming information, mainly formed by potentially efficient solutions sets relative to each **process** i, and ideal points relating to apiece cluster's nodes.

We expected that, applying alike algorithmic optimization, would accelerate feasible domain sweeping, as so as ***CIMOES*** convergence, by ensuring a balance between intensification and diversification on research area. In the experimentation, we will indexe parallel processing version by adding the suffix ***prll***.

4.2 Instances generation

CIMOES approach and parallel variant, have been implemented under the programming language $PYTHON$ 3.4; below we display results relating to developed programs execution on randomly generated MONLCIP instances (with quadratic functions), using a $PC\ Intel\ (R)\ Core\ (TM)\ CPU\ i7-4800MQ\ 2.70\ GHz$ and 16 $GB\ RAM$. Note that all procedures have been coded and no optimized solver has been used.

As mentioned before, without losing ***CIMOES*** generality on MONLCIP, r linear criteria $(r \in \{3, 5\})$ are generated, with integer factors randomly generated from an uncorrelated and uniform distribution in the interval $[-30, 30]$. The number of variables n is taken in $\{20, 30, 60, 120\}$. Also, p_1 linear constraints of form $Ax \leq b$, are generated, the A elements

are randomly generated integers, uncorrelated and uniformly distributed on $[10, 30]$. p_1 is equal to $\lceil n/2 \rceil$.

After linear domain V, we construct nonlinear one S, where $p_2 \in \{1, 3\}$ is the number of nonlinear constraints, generated in the quadratic form $x^t Q x + bx - d \leq 0$, each factor of the quadratic form is obtained as follows: initially, a triangular matrix is generated randomly with elements in $[0, 2]$, then multiplied by its transposed to form Q matrices, Q coefficients are in $[0, (2^2 \times n)]$. By following this Process, the generated matrices Q are positive semi-definite which guarantees the convexity of the nonlinear domain. The factors of the vector b are generated randomly with no correlation in the interval $[30, 60]$.

On the other hand, second hand side vectors are engendered as flows:

(a) For V constraints, second hand side values, are randomly chosen on intervals:

$$\begin{cases} b_j \in [80, 100] \ \forall j = \overline{1, p_1} & if \ \ n < 80 \\ b_j \in [100, 120] \ \forall j = \overline{1, p_1} & otherwise \end{cases}$$

(b) Each S constraint second hand side, is engendered as follow:

$$d = ((n \times 2^2) \times 3) - (n \times 2)$$

As reported before, the S-generating process, guarantees its convexity; however, while n increases, nonlinear constraints terms increase further (certainly majored by $(n \times 2^2)$). In fact, that is why relative second hand side is a function of n.

Domain-generating manner guarantees a primary condition: nonlinearity of domain U or non-violation of the $U \subsetneq V$ constriction; what will be expressly reported in next tables on $Linz\_CP$ column. *"Random numbers shouldn't be generated with a method chosen at random, some theory should be used"* ***Donald Knuth***

4.3 Numerical experimentation and outcomes analysis

For each class (n, p_1, p_2, r), a battery of 10 instances is solved, the set of efficient solutions is fully determined. Below, tables summarizing experimentation results.

Note that for both tables, columns Eff, $Linz\_CP$ and $CPU(s)$, reports the arithmetical average and the $[max, min]$ of respectively $card(Eff)$, created *linearization Cut-Plane* number and CPU time, realized on ten (10) instances solved of each problem-class (n, p_1, p_2, r).

While, each Sv column represents the number of instances solved in less than 4 *hours* (otherwise, the processing is stopped after the 14400 *seconds* limit).

Table 1: Results for instances with three criteria ($r = 3$).

$r = 3$			CIMOES			CIMOES_Prll		
n	p_1 p_2	Eff	$Linz\_CP$	$CPU(s)$	Sv	$Linz\_CP$	$CPU(s)$	Sv
20	10 1	31,9 [43,17]	703,2 [965,600]	33,9 [62,19]	10	868,7 [1112,692]	45,6 [54,39]	10
20	10 3	24,7 [31,16]	796,0 [944,618]	77,5 [128,37]	10	980,1 [1194,812]	87,0 [129,45]	10
30	15 1	46,8 [105,23]	2003,1 [4347,160]	629,3 [2292,99]	10	2319,1 [4790,177]	375,7 [1104,81]	10
30	15 3	68,4 [119,27]	2067,4 [4035,940]	495,4 [1654,177]	10	2301,7 [4287,831]	297,0 [771,149]	10
60	30 1	64,7 [145,26]	2127,5 [4602,124]	5677,0 [8383,431]	10	1863,7 [4801,439]	2043,3 [3806,293]	10
60	30 3	52,1 [81,39]	3268,0 [7868,1040]	5721,3 [7795,3172]	10	3671,6 [6564,1586]	1817,2 [2524,1147]	10
120	60 1	50,0 [67,34]	600,5 [654,547]	10831,0 [14357,7305]	2	1454,3 [4147,600]	6904,1 [9218,1276]	10
120	60 3	31,4 [46,20]	2372,2 [4802,807]	10300,3 [12763,7239]	3	2334,2 [5379,815]	5104,7 [7874,1504]	10

At first sight in both tables, the execution time $CPU(s)$ of the two approach is strongly correlated (positively) to n variation; unlike p_2, which shows only a slight (negative) influence on $CPU(s)$. We explain the last point by the fact that, feasible domain is more reduced each time we insert a new constraint, which also has an impact on the efficient solutions number $|Eff|$.

Regarding processing, and in synchronous mode, it was not always possible to execute 10 instances for each class (n, m, p_1, p_2), given that processing time exceeded the imposed limit of 4 *hours*. Indeed, for $r = 3$ and $r = 5$ respectively, only 81% and 74% of the instances were solved by the synchronous approach.

Adopted parallelism subdivides the initial domain U into eight sub-domains (since our microprocessor contains only $8 - cores$), thus, eight MONLCIP sub-problems are solved simultaneously. As a result, $Linz\_Cp$ is on average 20% greater than that in synchronous mode. In return, the $CPU(s)$ is on average reduced by **50%** thanks to the parallelization; however, when considering only solved instances to calculate this **Gap-ratio**, it was approximately **70%**. Obviously, $CPU(s)$ **Gap-ratio** would be **greater** then **70%**, if we suspended the 4 *hours* processing-limit.

Table 2: Results for instances with five criteria ($r = 5$).

$r = 5$			CIMOES			CIMOES_Prll		
n	p_1 p_2	Eff	$Linz\_CP$	$CPU(s)$	Sv	$Linz\_CP$	$CPU(s)$	Sv
20	10 1	90,4 [178,61]	1068,3 [2062,634]	138,2 [445,49]	10	1292,2 [2307,724]	119,5 [289,51]	10
20	10 3	75,5 [115,46]	1072,8 [1346,558]	141,8 [252,42]	10	1195,3 [1550,802]	137,1 [211,56]	10
30	15 1	230,6 [408,62]	2245,1 [6396,360]	1567,6 [5316,191]	10	2445,1 [6603,637]	618,5 [2024,145]	10
30	15 3	207,3 [324,105]	2377,6 [5183,620]	1066,6 [2582,247]	10	2813,1 [5109,804]	487,2 [922,164]	10
60	30 1	208,9 [275,69]	1787,1 [3320,602]	9338,4 [14062,1163]	8	1974,3 [3757,767]	3302,8 [5193,547]	10
60	30 3	187,3 [285,132]	1025,4 [1291,745]	5551,7 [8690,1380]	8	1325,5 [1964,807]	3226,0 [8625,664]	10
120	60 1	143,0 [385,66]	1080,9 [1081,1081]	12676,7 [12677,12677]	1	1396,9 [1654,861]	7828,9 [14208,2584]	10
120	60 3	127,6 [195,79]	785,1 [802,768]	12124,7 [12994,11256]	2	1127,8 [1402,891]	8386,2 [14379,2820]	10

Finally, a strong sensitivity of $|Eff|$ to the variation of r is recorded; moreover, $|Eff|$ increased three more times in the second table compared to the first. We observe the same trend for the $CPU(s)$. Although as advanced, some instances resolution by the synchronous approach was interrupted, this biased some ratios. On the other hand, we did not find any significant correlation between $Linz\_Cp$ and r.

5 Conclusion

We exposed through this work, an exact general approach dedicated to solve MONLCIP, a kin of problems principally characterized by its hardness to solve, since it contains both nonlinear (convex) functions, discrete variables restriction as well as multiple-objective functions to optimize. The strength of the engineered approach is that, initial problem is relaxed to a generic linear one; accordingly, all linear-optimization tools and method are suitable. Effectively like seen hereinbefore, the simplex method was used to optimize objectives on a polyhedron instead of nonlinear domain. Beyond cutting planes uses in linearization process, efficient cut described in equation (4) truncates some non-efficient parts of the feasible domain. Also, throughout the multiobjective branch and bound

process, a fathoming step based on locally ideal points was used to avoid exploring non-promising tree-nodes.

Furthermore, we proved that our approach converges in a finite number of iterations, and generates the entire efficient solutions set for the problem.

We developed a parallel-processing variant, based on the "*single program multiple data*" model, with a distributed memory managed by the "*gather & scatter*" mechanisms. Clearly, this last approach performed more than expected; indeed, comparing to synchronized approach outcomes, more than **70%** of $CPU(s)$ **Gap** was registered for many instances, as reported in our experimentation-results analysis. Finally, one more advantage resides in the fact that, through some appropriate modifications, our method can be extended to deal with mixed variables, which will certainly reduce the difficulty engendered by the integrity of all problem variables.

References

[1] Moncef Abbas, Mohammed-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*, 4(3), 2012.

[2] Harold P. Benson. An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. *Journal of Global Optimization*, 13(1):1–24, Jan 1998.

[3] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. Algorithms and Software for Convex Mixed Integer Nonlinear Programs. October 2009. Technical Report #1664, Computer Sciences Department, University of Wisconsin-Madison, 2009.

[4] Valentina Cacchiani and Claudia D'Ambrosio. A branch-and-bound based heuristic algorithm for convex multi-objective minlps. *European Journal of Operational Research*, 260(3):920–933, 2017.

[5] Mohamed-El-Amine Chergui, Mustapha Moulai, and Fatma Zohra Ouaïl. Solving the multiple objective integer linear programming problem. volume 14, pages 69–76, 01 2008.

[6] Olivier de Weck. Multiobjective optimization: History and promise. *Invited Keynote Paper, GL2-2, the Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems*, 2, 01 2004.

[7] Marco A. Duran and Ignacio E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, Oct 1986.

[8] Matthias Ehrgott, Lizhen Shao, and Anita Schöbel. An approximation algorithm for convex multi-objective programming problems. *Journal of Global Optimization*, 50(3):397–416, Jul 2011.

[9] J. Ch. Fiorot. Algorithmes de programmation convexe par linéarisation en format constant. *RAIRO - Analyse numérique*, 11(3):245–253, 1977.

[10] Renu Gupta and Rita Malhotra. Multi-criteria integer linear programming problem. *Cahiers du Centre d'Études de Recherche Opérationnelle*, 34(1):51–68, 1992.

[11] P. Huard. Tour d'horizon : programmation non linéaire. *Revue Française d'Informatique et de Recherche Opérationnelle R.I.R.O*, 5(R1):3–48, 1971.

[12] J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

[13] Gokhan Kirlik and Serpil Sayin. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232:479–488, 02 2014.

[14] Kathrin Klamroth and Tind Jørgen. Constrained optimization using multiple objective programming. *Journal of Global Optimization*, 37(3):325–355, Mar 2007.

[15] Dieter Klein and Edward L. Hannan. An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, 9:378–385, 1982.

[16] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, Apr 2004.

[17] Fatma Zohra Ouaïl and Mohamed El-Amine Chergui. A branch-and-cut technique to solve multiobjective integer quadratic programming problems. *Annals of Operations Research*, 267(1):431–446, Aug 2018.

[18] Melih Özlen and Meral Azizoğlu. Multi-objective integer programming: a general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199(1):25–35, 2009.

[19] I. Quesada and I.E. Grossmann. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & Chemical Engineering*, 16(10):937 – 947, 1992.

[20] Lizhen Shao and Matthias Ehrgott. Approximately solving multiobjective linear programmes in objective space and an application in radiotherapy treatment planning. *Mathematical Methods of Operations Research*, 68(2):257–276, Oct 2008.

[21] John Sylva and Alejandro Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158(1):46–55, 2004.

[22] Tapio Westerlund and Frank Pettersson. An extended cutting plane method for solving convex minlp problems. *Computers & Chemical Engineering*, 19:131–136, 1995.