A genetics algorithms for optimizing a function over the integer efficient set

Ali Zaidi[1,*], **Djamal Chaabane** [1], **Larbi Asli**[2], **Lamine Idir** [3], **and Saida Matoub**[3]

[1] *Laboratory of Multiple Criteria Decision and Operations Research (AMCD and RO), Faculty of Mathematics, USTHB University, Algiers, Algeria*
E-mail: ⟨a.zaidi@crlca.dz, dchaabane@usthb.dz⟩

[2] *LaMOS Laboratory, Faculty of Exact Sciences, University of Bejaia, Bejaia, Algeria*
E-mail: ⟨larbi.asli@univ-bejaia.dz⟩

[3] *Centre for Research in Amazigh Language and Culture (CRLCA), Bejaia, Algeria*
E-mail: ⟨idir.lamine@ensc.dz, saida.matoub@univ-bejaia.dz⟩

Abstract. In this paper, we propose an algorithm called Directional Exploration Genetic Algorithm (DEGA) to resolve a function Φ over the efficient set of a multi-objective integer linear programming problem (MOILP). DEGA algorithm belongs to evolutionary algorithms, which operate on the decision space by choosing the fastest improving directions that improve the objectives functions and Φ function. Two variants of this algorithm and a basic version of the genetic algorithm (BVGA) are performed and implemented in Python. Several benchmarks are carried out to evaluate the algorithm's performances and interesting results are obtained and discussed.

Keywords: DEGA Algorithms, efficient set, genetic algorithms, optimization over the efficient set

Received: December 20, 2024; accepted: April 26, 2024; available online: May 27, 2024

DOI: 10.17535/crorr.2024.0007

1. Introduction

Multi-objective optimization is a type of optimization problem, that differs from classical problems, due to the presence of a set of efficient solutions rather than a single optimal solution. This multiplicity of solutions can create a real dilemma for the decision maker, who ultimately has to make his choice among the possible set of 'efficient' solutions.

To avoid this problem, a modeling approach called optimization of a function over the set of efficient solutions was proposed in 1972 by Phillip [15]. Since then, numerous works have been carried out in this field, whether for continuous problems (see [6, 19, 12]) or discrete problems(see [1, 10, 5, 3, 2]).

The problem of optimizing a function over the set of efficient solutions of a multi-objective integer linear problem(MOILP) was first addressed by Abbas in 2006 [1] and since then, several works have followed, such as those of Jesus in 2009 [10], Chaabane in 2010 [5], Brahmi in 2012 [3], Boland in 2017 [2].

This approach has been applied to other problems such as the MOILP-stochastic treated by Mebarek in 2014 [4], the quadratic optimization over a discrete Pareto set of a multi-objective linear fractional program addressed by Moulai in 2021 [14], as well as MOILP in a fuzzy environment by Menni in 2020 [13].

[*]Corresponding author.

Although all these previous methods are deterministic, they have limitations in problem size and CPU time, especially for integer problems. This is why we address in this work the problem of optimizing a function over the set of efficient solutions of a MOILP using a meta-heuristics. We adapt a genetic algorithms for this work for its many advantages, including: Efficiency, they can handle large search spaces and explore a wide diversity of solutions in a relatively short time. Exploration of the search space, they are capable of efficiently exploring the search space to find optimal or near-optimal solutions. Parallelism, they can be easily parallelized to accelerate the search for solutions.

In this article, we present three variants of genetic algorithms. The first is the basic version of the genetic algorithm (BVGA) based on crossover and mutation operators, using well-known methods. The second is based on the exploration of the directions of the influential variables without updating their influence weights, while the third is based on updating the influence coefficients of the variables. This paper is organized as follows: Following the introduction in section 2, we present the preliminaries and basics of optimization over the efficient set and genetic algorithms. Next in section 3, we will propose three algorithms to solve the considered problem. To illustrate our algorithm DEGA-II, we will unfold the steps of a didactic example in subsection 3.4. In section 4, we will present the computations results and its discussions. Finally, in section 5, the paper ended by conclusion.

2. Preliminaries and basic concepts

2.1. Mathematics modeling

A multi-objective integer linear problem (MOILP) embodies a mathematical optimization problem aimed at improving several objectives simultaneously within the confines of linear constraints and integer decision variables. This problem is structured as follows:

$$(\Pi)\begin{cases}\min\ Z_j(x) = c_j^\top x, \text{ j=1,...,p},\\ st.\quad x \in \boldsymbol{D}.\end{cases} \tag{1}$$

Where $\boldsymbol{D} \equiv \{x \in \boldsymbol{Z}_+^n | Ax \leq b\}$ is the feasible set of the problem, with $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $C = (c_j^\top,\ \ j = 1, \ldots, p)$ is a $p \times n$-matrix defining a number $p \geq 2$ of objective functions. We suppose that the feasible set $\boldsymbol{D}$ is not empty and bounded. As the objective functions are usually conflicting, generally there does not exist any feasible solution optimizing all the criteria simultaneously and thus, the concept of an efficient solution is widely used.

We denote by $\boldsymbol{E}$ the set of efficient solutions of (Π) and the problem we want to treat is the problem $(\Pi_{\boldsymbol{E}})$ defined as:

$$(\Pi_{\boldsymbol{E}})\begin{cases}\min\ \Phi(x) = \phi^\top x,\\ st.\quad x \in \boldsymbol{E}.\end{cases} \tag{2}$$

where ϕ denotes n dimensional vector.

Definition 1. *A point $\overline{x} \in \boldsymbol{D}$ is considered an efficient solution if and only if there is no $x \in \boldsymbol{D}$ such that $Z_j(x) \geq Z_j(\overline{x})$ for all $j \in \{1, 2, \ldots, p\}$ and $Z_j(x) > Z_j(\overline{x})$ for at least one j [17].*

2.2. Genetic Algorithm

The concept of Genetic Algorithm (GA) was developed by Holland in the 1960-1975 [8]. GA is inspired by the principles of natural selection and genetics. It is widely used in various fields, including computer science, engineering, biology and artificial intelligence to solve complex problems. GA operate by evolving a population of potential solutions over multiple generations.

Solutions are represented as chromosomes, typically composed of genes that encode specific parameters or characteristics of the considered problem. During each generation, individuals are selected based on their fitness, and genetic operations, such as crossover and mutation are applied to create new offspring. Over time, GA seeks to improve solutions iteratively. The strength of GA lies in their ability to explore large solution spaces and find solutions in complex non-linear or poorly understood problems. It is a versatile tool for tackling optimization problems. Schaffer [16] proposed the first multi-objective GA, known as vector-evaluated GA (VEGA). Several other algorithms were also developed after that, the most significant of these algorithms are MOGA , NPGA , WBGA , RWGA , NSGA , SPEA , SPEA2 , PAES , PESA-II , NSGAII , MEA , Micro-GA , RDGA and DMOEA .

The GA is composed of three essential steps : *Initial population,* is composed of a set of chromosomes generated. *Mating*: This step is composed of three operations: The first is the selection step (tels que split–based selection (SBS)[9]) which consists of choosing a subset of chromosomes based on the evaluation of a specific function, so-called fitness function to reproduce the next generation. The second called crossover, is the process of combining two chromosomes to create new chromosomes of the offspring. the thread the mutation, is the process of randomly selecting genes within an individual's chromosome and applying small changes to their values.

Survival: is the process of assembling the survival chromosomes of the previous population and the offspring created.

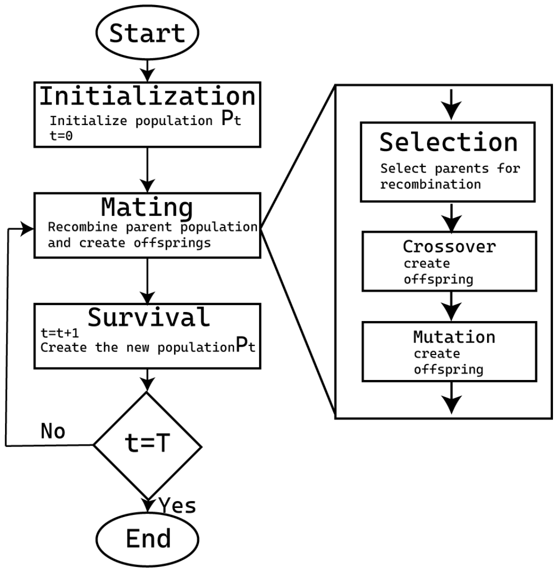


Figure 1: *Genetic algorithm diagram.*

3. Algorithms description

To solve the problem considered above, we adapt and introduce three variants of a genetic algorithm. The first one is called the basic version of the genetic algorithm (BVGA) and the two others called the Directional Exploration Genetic Algorithm DEGA-I and DEGA-II . They have some common parameters, such as the initial population P^0, the population size n_p and the iteration number T.

3.1. BVGA algorithm

BVGA is a general version of a genetic algorithm that uses the basic operators. In selecting the latter, we have drawn on the analysis carried out in Sourabh Katoch's article published in 2021[11]. Given that our problem is in integers, we've opted for an integer encoding (value encoding) of our individuals. For the selection operator, we have chosen Rank Selection and Uniform Crossover for the crossover operator. For the mutation operator, we have preferred Simple-Inversion Mutation. It starts by fixing different parameters such as the number of solutions to select for the crossing θ, the number of genes to cross λ and initializing the iterations parameter $t = 0$. We start the process by generating the individuals of the initial population P^0. Then, we choose the set of solutions to be crossed and mutated, as well as the type of crossover to be performed. For the mutation process, the variable will be selected randomly. After the crossover and mutation steps have been performed, we select the n_p best solutions from P^0 that optimize the Φ function. Then, we use these solutions to create a new population P^t (The solutions of P^t are feasible and not-dominated). We will continue the process until t equals T. Then among the non-dominated solutions of P^T, we choose the one with the best fitness.

```
BVGA algorithm
 Input:
 $\downarrow A_{(m\times n)}$: constraints matrix;
 $\downarrow b_{(m\times 1)}$: RHS vector;
 $\downarrow \phi_{(1\times n)}$: compromise criterion vector;
 $\downarrow C_{(p\times n)}$: objectives matrix;
 $\downarrow \theta$: number of solutions to select for the crossing;
 $\downarrow \lambda$: the number of genes to crosses;
 $\downarrow n_p$: size population to be generated ;
 $\downarrow T$: the maximum number of iterations;
 Output:
 $\uparrow x^*$: optimal solution of the problem $\Pi_E$;
 $\uparrow \Phi(x^*)$: optimal value of function $\Phi$;
 Initialization:
 $t \longleftarrow 0$;
  begin
   -Generate the initial population $P^0$.
    While $t \leq T$ do
      -Select $\theta$ solutions from $P^t$ to cross;
      -Crossing $\theta$ solutions ;
      -Mutate the solutions;
      -Create the population $P^{t+1}$.
      -$t \longleftarrow t+1$;
    end while
    -Elimination of dominated solutions in $P^T$;
   end.
```

3.2. DEGA-I algorithm

DEGA-I is a genetic algorithm based on the Pareto approach. It is also inspired by Schaffar's principle [16], which was used in the VEGA algorithm to separate the Φ function from the multi-objective problem (MOILP). A subset of size α of a population P^t focuses on the optimization of the Φ function, while the remaining population focuses on the optimization of the MOILP problem. The two parts are then re-concatenate at each iteration. Firstly we specify the parameters needed, such as the number of variables to explore by classification of their influence weights γ. The set of solutions $S = \emptyset$ and the iteration index $t = 0$. After that, we proceed to the calculation of the influence weights of each coefficient variables by the following formulas:

$$\rho_{Z(x_j)} = \frac{1}{p}\sum_{i=1}^{p} \frac{c_{ij}}{\sum_{k=1}^{n} |c_{ik}|} \text{ and } \rho_{\Phi(x_j)} = \frac{\phi_j}{\sum_{k=1}^{n} |\phi_k|}.$$

$\rho_{Z(x_j)}$ and $\rho_{\Phi(x_j)}$ are composed of two parts, the absolute value which gives the weight of influence and the sign which gives the direction of improvement objectives. The influence weights will be ranked in decreasing order of their absolute values and selecting the first subset of variables of size γ to be explored. Then the iterative process is started until the number of iterations t equals T. In each iteration, we will construct a new population (that will be explain next). This population will be composed of a set of solutions that satisfy the MOILP's constraints. We will apply a filtering process to remove any dominated solutions for the new population P^t of size s_t. If $s_t > n_p$, we keep only the n_p solutions that produce the best Φ function results for the newly created population. Else if $s_t < n_p$, we generate what's missing as we did with the initial population. The algorithm stops when t equals T.

```
DEGA-I algorithm
Input:
$\downarrow A_{(m\times n)}$: matrix of constraints;
$\downarrow b_{(m\times 1)}$: second member vector;
$\downarrow \phi_{(1\times n)}$: compromise criterion vector;
$\downarrow C_{(p\times n)}$: matrix of criteria;
$\downarrow \gamma$: the number of variables to explore;
$\downarrow \alpha$: selection parameter;
$\downarrow n_p$: the size of the population to be generated;
$\downarrow T$: the maximum number of iterations;
Output:
$\uparrow x^*$: optimal solution of the problem $\Pi_E$;
$\uparrow \Psi(x^*) : \uparrow \Phi(x^*)$: optimal value of criterion $\Phi$;
Initialization:
$t \longleftarrow 0$; $S \longleftarrow \emptyset$;
-Calculate the influence weights of the coefficients of the variables $\rho_{Z(x_i)}$ and $\rho_{\Phi(x_i)}$.
-Generate the initial population $P^0$ which verifies the constraints and is not dominated.
-Rank the population solutions $P^0$ in increasing order according to the value of $\Phi(x_i)$.
Select the variables to be explored for $\Phi$ and $Z$.
-$s_t \longleftarrow n_p$;
  While $t \leq T$ do
    -$t \longleftarrow t+1$;
    -Create the next population $P^t$.
    -$s_t \longleftarrow$ size of $P^t$;
    If $s_t > n_p$
      -Rank the population solutions $P^t$ in an increasing order according to the value of $\Phi(x_i)$;
      -$P^t$ keeps only the first $n_p$ solutions;
    else if $s_t < n_p$
      -$rst \longleftarrow n_p - s_t$;
      -$S \longleftarrow$ Generate $rst$ solutions;
      -$P^t \longleftarrow P^t \cup S$;
    end if
  end while
  -Elimination of dominated solutions in $P^t$;
 end.
```

3.2.1. Major steps of the algorithm

Building new solutions for the new population:
To create the new population P^t, we divide our population P^{t-1} into two distinct subsets: the first one, with size α contains the solutions that have the best values of the function Φ, while the second comprises the remaining solutions. Each solution of P^{t-1} will produce γ new solutions by exploring the last selected directions individually (to avoid redundancy) and to do this we increase by $+1$ each selected decision variable that has a $\rho_{\Phi(x_j)} < 0$ or $\rho_{Z(x_j)} < 0$, and decrease by -1 for those that have a $\rho_{\Phi(x_j)} \geq 0$ or $\rho_{Z(x_j)} \geq 0$.

Check the feasibility constraints of the new solutions:
This step aims to ensure that every solution of the new population verifies the constraints.
Check the non-dominance of new solutions:
This is the key step of the algorithm, it replaces the efficiency test used in the different deterministic algorithms quoted in the literature, we use the Pareto dominance, where $Z(x_i) < Z(x_j)$ means that $Z(x_i)$ dominates $Z(x_j)$.

3.3. Algorithm DEGA-II

The DEGA-II algorithm is similar to the DEGA-I algorithm, the only difference consists of including an additional step of updating the influence weights of the coefficients variables at each iteration. Indeed, to avoid being trapped in directions that do not improve our solutions. It is necessary to allow the system to weaken the influence weights of the bad directions. To achieve this, we use formulas that update the absolute values of these influence weights $\mid \rho_{Z(x_j)}(t+1) \mid = \mid \rho_{Z(x_j)}(t) \mid \times (\tau_j)^\nu$ and $\mid \rho_{\Phi(x_j)}(t+1) \mid = \mid \rho_{\Phi(x_j)}(t) \mid \times (\tilde{\tau_j})^\nu$. Where $0 \leq (\tau_j = \dfrac{\eta_j}{\gamma}) \leq 1$, $0 \leq (\tilde{\tau_j} = \dfrac{\tilde{\eta_j}}{\gamma}) \leq 1$. η_j and $\tilde{\eta_j}$ are the number of non-dominated solutions which verify the constraints created by exploring the direction of variable x_i and $\nu > 0$ is the update speed which is related to the number of variables that remain to be explored. In the case $\eta_j = 0$ or $\tilde{\eta_j} = 0$, we include their coefficients in a set named Ω. This set represents a set of directions to avoid or not to explore. At the end, we re-sort coefficients weights to select the new directions for the next generation.

```
DEGA-II algorithm
Input:
$\downarrow A_{(m\times n)}$: matrix of constraints;
$\downarrow b_{(m\times 1)}$: second member vector;
$\downarrow \phi_{(1\times n)}$: compromise criterion vector;
$\downarrow C_{(p\times n)}$: matrix of criteria;
$\downarrow \gamma$: the number of variables to explore;
$\downarrow \alpha$: selection parameter;
$\downarrow n_p$: the size of the population to be generated;
$\downarrow T$: the maximum number of iterations;
Output:
$\uparrow x^*$: optimal solution of the problem $\Pi_E$;
$\uparrow \Psi(x^*) : \uparrow \Phi(x^*)$: optimal value of criterion $\Phi$;
Initialization:
$t \longleftarrow 0$; $S \longleftarrow \emptyset$; $\Omega_Z = \emptyset$; ‘$\Omega_\Phi = \emptyset$;
  begin
  -Calculate the influence weights of the coefficients variables $\rho_{Z(x_i)}$ and $\rho_{\Phi(x_i)}$.
  -Generate the initial population $P^0$ which verifies the constraints and is not dominated.
  -Rank the solutions of population $P^0$ in increasing order according to the value of $\Phi(x_i)$.
  -Select the variables to be explored for $\Phi$ and $Z$.
  While $t \leq T$ and $(Card(\Omega_Z) \leq n)$ or $(Card(\Omega_\Phi) \leq n)$ do
    $t \longleftarrow t+1$;
    -Create the population $P^t$;
    -Update and reclassify influence weights $\rho_{Z(x_i)}$ and $\rho_{\Phi(x_i)}$;
    -$s_t \longleftarrow$ size of $P^t$
    If $s_t > n_p$
      $\theta \longleftarrow \theta - \varepsilon$;
      -Rank the solutions of population $P^t$ in an increasing order according to the value of $\Phi(x_i)$;
      -$P^t$ keeps only the first $n_p$ solutions;
    else if $s_t < n_p$
      -$rst \longleftarrow n_p - s_t$;
      -$S \longleftarrow$ Generate $rst$ solutions;
      -$P^t \longleftarrow P^t \cup S$;
    end if
  end while
  -Elimination of dominated solutions in $P^t$;
  end.
```

3.4. Numerical example

To illustrate our algorithm DEGA-II consider the following example (Π) and its graphical representation (see Figure 2):

$$(\Pi)\begin{cases}\min\ Z_1(x) = -3x_1 + x_2\\ \min\ Z_2(x) = 2x_1 - 3x_2\\ s.t.\quad (\boldsymbol{D})\begin{cases} x_1 + 2x_2 & \le 8\\ x_1 & \le 5\\ x_2 & \le 7\\ x_1 + x_2 & \le 10\\ x_1, x_2 & \in \mathbb{N}_+\end{cases}\end{cases} \tag{3}$$

Let be the compromise problem :

$$(\Pi_E)\begin{cases}\min\ \Phi(x) = x_1 + 3x_2\\ s.t.\quad (x_1, x_2) \in \boldsymbol{E}\end{cases} \tag{4}$$

Initialization:

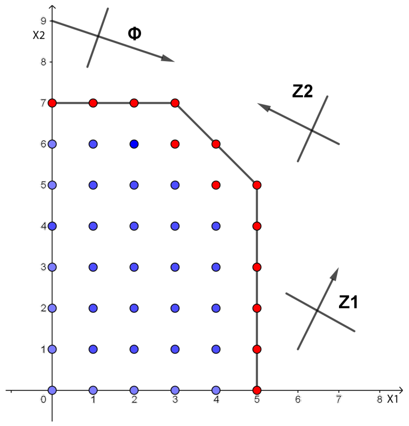


Figure 2: *Feasible set of solutions.*

$n_p = 6$, $\gamma = 1$, $\alpha = 3$, $\nu = 1$, $T = 10$, $t = 0$, $S = \emptyset$, $\Omega_Z = \emptyset$ and $\Omega_\Phi = \emptyset$.

We calculate the influence weights of the coefficients of the variables :

$$\rho_{Z(x_1)} = -\frac{23}{20},\ \rho_{Z(x_2)} = -\frac{7}{20},\ \rho_{\Phi(x_1)} = \frac{1}{4},\ \rho_{\Phi(x_2)} = \frac{3}{4}.$$

We have $|\rho_{Z(x_2)}| < |\rho_{Z(x_1)}|$ and $|\rho_{\Phi(x_2)}| > |\rho_{\Phi(x_1)}|$, thus we explore the direction of the variable x_2 for the function Φ and x_1 for the objectives.

Generate the initial population : $P^0 = \{(0,7),(3,7),(3,6),(1,1),(5,2),(4,6)\}$.
Rank the solutions according to their value of function Φ. Solutions 4 and 5 were identified as those that optimize Φ, while the other solutions optimize $Z_1(x)$ and $Z_2(x)$.
Iteration 1: $t = 0$ (Represented by Fig.3)

Build the new population $\acute{P}^0 = \{(1,7),(4,7),(4,6),(1,0),(5,1),(5,6)\}$, check its feasibility. $P^1 \longleftarrow P^0 \cup \acute{P}^0$. After checking its non-dominance, $P^1 = \{(0,7),(3,6),(4,6),(5,2),(1,7),(5,1)\}$.

Update the influence weights of the coefficients of the variables : $\rho_{Z(x_1)} = -\frac{23}{40}$ and $\rho_{\Phi(x_2)} = \frac{3}{8}$.

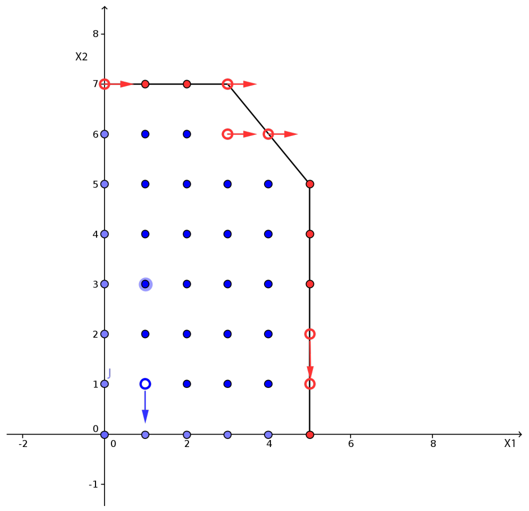


Figure 3: *Iteration 1.*

We still have $|\rho_{Z(x_2)}| < |\rho_{Z(x_1)}|$ and $|\rho_{\Phi(x_2)}| > |\rho_{\Phi(x_1)}|$, thus we will explore the same directions.
Iteration 2: $t = 1$ (Represented by Fig.4)

Build the new population $\acute{P}^1 = \{(1,7),(4,6),(5,6),(2,7),(5,1),(5,0)\}$, Check its feasibility. $P^2 \longleftarrow P^1 \cup \acute{P}^1$. After checking its non-dominance, $P^2 = \{(0,7),(3,6),(4,6),(5,2),(5,1),(5,0)\}$.

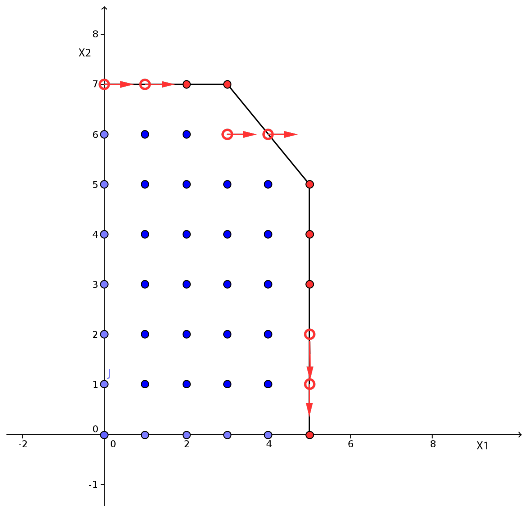


Figure 4: *Iteration 2.*

Update the influence weights of the coefficients of the variables : $\rho_{Z(x_1)} = -\frac{69}{320}$ and $\rho_{\Phi(x_2)} = \frac{3}{16}$.

$|\rho_{Z(x_2)}| > |\rho_{Z(x_1)}|$ and $|\rho_{\Phi(x_2)}| < |\rho_{\Phi(x_1)}|$, thus we change the exploration directions for the two directions variables. $\Omega_Z = \{1\}$, $\Omega_\Phi = \{2\}$.

Iteration 3: $t = 2$
Build the new population $\acute{P}^2 = \{(0,8),(3,7),(5,3),(4,7),(4,1),(4,0)\}$, check its feasibility.

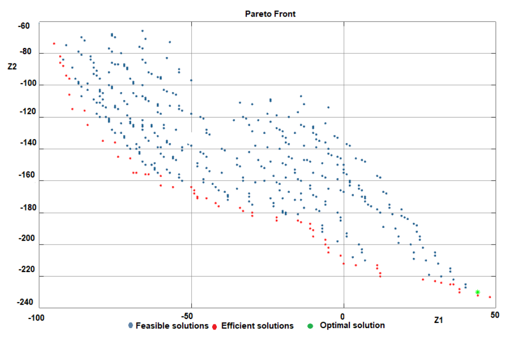
$P^3 \longleftarrow P^2 \cup \acute{P}^2$. After checking its non-dominance, $P^3 = \{(0,7),(3,6),(5,2),(5,1),(5,0),(5,3)\}$. Update the influence weights of the coefficient variables $\rho_{Z(x_2)} = -\frac{56}{320}$ and $\rho_{\Phi(x_1)} = 0$.

$|\rho_{Z(x_2)}| < |\rho_{Z(x_1)}|$ and $|\rho_{\Phi(x_2)}| > |\rho_{\Phi(x_1)}|$, thus we change the exploration directions for the two directions variables, $\Omega_Z = \{1,2\}$, $\Omega_\Phi = \{2,1\}$. The $Card(\Omega_Z) = 2$ and $Card(\Omega_\Phi) = 2$, then the algorithm stops.

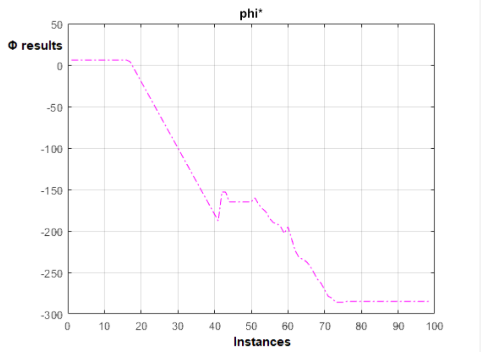
The optimal solution of Π_E is $x^* = (5,0)$ with : $\Phi(x^*) = 5$, $Z_1(x^*) = -15$ and $Z_2(x^*) = 10$.

3.5. Graphical bi-objective example

In this subsection, we present the graphic results of an implementing instance of MOILP ($p = 2$, $n = 10$, $m = 8$) using the algorithm DEGA-II.



(a) *Distribution of feasible solutions and the Pareto front.*



(b) *Behavior of the Φ solution during iterations of the DEGA-II algorithm.*

Figure 5: *Graphical bi-objective example.*

In Figure 5a, we graphically represent the distribution of feasible solutions to our problem (shown in blue), highlighting the Pareto front (marked in red), while the solution to problem $\Pi_{\boldsymbol{E}}$, optimizing function Φ, is marked in green. This figure highlights the complexity of the problem and the presence of thousands of feasible solutions. However, in Figure 5b, we can see the evolution of the Φ-function solution over the iterations. From this figure, we can see that our DEGA-II algorithm manages to stabilize and reach the optimal solution from the 75th iteration onwards.

4. Computational study

4.1. Performances measurements

The performance of an algorithm of resolution is one of the important steps in the resolution process, in our case, we chose three performance metrics: spacing metric (SM), hole relative size (HRS) for spacing studies and convergence metric (RP) for progression and convergence results [18].

Spacing metrics :The Spacing metric (SM) allows us to measure the uniformity of the spread of the points of the solutions set in the (Z^1, Z^2) plane. It's defined as :

$SM = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\bar{d} - d_i)^2}$. Where : $d_i = \min_{i=1}(|Z_i^1(\overrightarrow{x}) - Z_j^1(\overrightarrow{x})| + |Z_i^2(\overrightarrow{x}) - Z_j^2(\overrightarrow{x})|)$, $i,j = 1,\dots,N$ and $i \neq j$.

$\overline{d}$ is the mean value of all the d_i, and N is the number of elements in the solutions set.

HRS metric : The use of an SM metric (that calculates an average error relative to an optimal spacing) may obscure significant gaps in the results. To address this issue, a new metric called HRS (hole relative size) was introduced in [18]. The HRS metric enables us to measure the magnitude of the largest gap in the points distribution on the trade-off surface. Its definition is as follows: $HRS = \frac{\max d_i}{\overline{d}}$. Where $\overline{d}$ and d_i are equivalent to those defined in the spacing metric.
Convergence metric: The metric used to evaluate the improvement of the objective function Φ is as follows:
$RP = \ln \sqrt{\frac{\Phi_{\min}(0)}{\Phi_{\min}(T)}}$. The term $\Phi_{\min}(T)$ in the metric refers to the optimal value of the function Φ achieved at iteration T.

4.2. Computation results and analyses

We implemented our algorithms in Python and ran it on machine characteristics: Intel i7 2.0 GHz, 8 GB of RAM. Since there do not exist any benchmarks in the literature for our problem, we proceed to randomly generate 100 problems to test the performances of our algorithms as follows:

The coefficients of each matrix vary: for A, $a_{ij} \in \{1, 2, \ldots, 29, 30\}$. For C, $c_{ij} \in \{-20, -19, \ldots, 19, 20\}$. Φ, $\phi_j \in \{-20, -19, \ldots, 19, 20\}$, and the vector b, $b_i \in \{50, 51, \ldots, 149, 150\}$. The number of variables n, varies from 15 to 5000, the constraints m, from 10 to 4000 and the number of objective functions p is 4, 10, 30 or 100 in 5 different classes. For each class, 5 instances are solved. those examples are available in: (see link: https://sites.google.com/view/ali-zaidi/home).

We initialize the parameters according to the number of variables and objectives, noting that, when there is a small number of variables the influence of their coefficients is significant and on the other hand, the influence becomes minimal.
The initial parameters are: $\alpha = 0.7$, $\gamma = 2$, $\delta = 1$, $n_p = 10$, $\nu = 10$ and T varies from 50 till 15000.

The results have been summarized in Tables 1, which display the value of function Φ, the CPU time (in seconds) and the performance metrics: SM, HRS and RP.

To see more clearly the results from the above Table 1, we have graphically represented each metric.
Figure 6a: illustrates the variation results of function Φ for each instance of BVGA, DEGA-I, DEGA-II and we observe that the results of BVGA are significantly worse than the other methods, indicating that it gets stuck in a local minimum. On the other hand, it is clear that DEGA-II performs better than DEGA-I in all cases, especially when the size of variables is important.

Figure 6b: displays the CPU time, where we observe that DEGA-II shows lower CPU times than DEGA-I.

Figure 7a, 7b and 8 show the performance metrics of our algorithms. The SM diversity metric in Fig 7a reveals that the distances between the Pareto front points of the DEGA-I and DEGA-II methods are homogeneously distributed, unaffected by problems size. On the other hand for BVGA, this distribution is uncertain and depends on problem size.

The HRS(see Figure 7b) index indicates that DEGA-II points are evenly distributed across the entire Pareto front, while DEGA-I solutions show a slight concentration towards a particular objective. On the other hand, the RP(see Figure 8) convergence index shows that the DEGA-II method converges significantly better than the other methods.

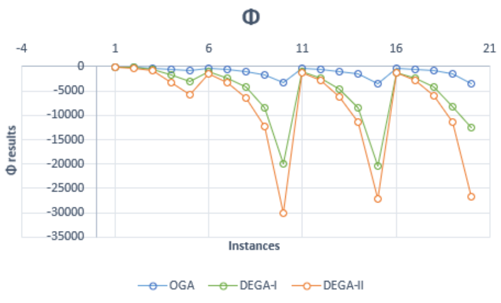
P=4															
	BVGA					DEGA-I					DEGA-II				
$m \times n$	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HR	RP
10×15	-73.4	15.3	8.7	2.74	0.04	-112.2	49.49	10.68	3.31	0.188	-109	13.77	7.58	3.6	0.168
50×80	-122.4	93.14	27	2.39	0.114	-188.4	82.39	32.41	6.39	0.256	-378.2	20.71	1.88	3.39	0.62
100×150	-263.6	134.51	39.16	2.56	0.038	-552	64.158	53.96	13.9	0.43	-884	28.66	8.005	3.11	0.638
300×500	-449	269.68	73.86	2.3	0.86	-1642.2	507.5	79.86	6.85	0.64	-3341.6	124.3	8.6	4.35	1.05
800×1000	-841.6	146.2	89.26	2.2	0.04	-3036.4	437.95	118.95	11.04	0.62	-5766.2	193.96	7.9	3.28	0.96

P=10															
	BVGA					DEGA-I					DEGA-II				
$m \times n$	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HR	RP
100×200	-263.2	12.702	76.27	1.6	0.046	-1056.4	5.57	15.5	2.84	0.72	-1367.4	98.062	12.74	2.3	0.85
300×500	-555	30.98	131.15	1.7	0.016	-2386	155.34	15.19	2.65	0.748	-3278.2	234.78	10.78	1.91	0.92
800×1000	-940.4	49.82	175.5	1.58	0.012	-4263.6	323.92	54.39	16.3	0.78	-6350.4	365.35	13.23	2.078	0.98
1800×2000	-1729.8	269.9	239.23	1.6	0.002	-8535.8	1640.1	19.85	2.86	0.8	-12283.6	1293.56	13.136	2.234	0.98
4000×5000	-3235.2	3447.15	404.15	1.64	0	-20002.2	17350.2	42.03	4.07	0.92	-3017.8	11945.94	12.5	2.05	0.92

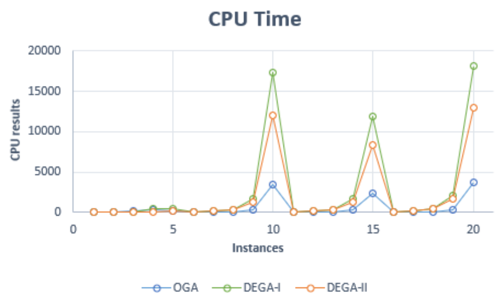
P=30															
	BVGA					DEGA-I					DEGA-II				
$m \times n$	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HR	RP
100×2006.3	-271.8	12.278	197.09	1.272	0.028	-988.4	58.496	16.45	1.576	0.704	-1252	96.57	30.504	1.908	0.762
300×500	-498	23.49	258.03	1.38	0.02	-2264.8	152.47	21.34	1.64	0.76	-2916.6	181.58	33.89	2.17	0.87
800×1000	-1090.4	51.7	368.84	1.298	0.006	-4578.2	374.46	29.02	2.304	0.716	-6208.6	394.6	25.22	1.74	0.87
1800×2000	-1487.25	272.32	-519	1.29	0.005	-8456	1681.65	39.79	2.86	0.87	-11430.9	1314.94	39.22	2.76	1.02
4000×5000	-3414	2384.2	836.02	1.3	0.21	-20306.4	11895.4	48.34	2.99	0.9	-27248.4	8307.8	22.22	1.61	1.04

P=100															
	BVGA					DEGA-I					DEGA-II				
$m \times n$	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HRS	RP	ϕ^*	CPU	SM	HR	RP
100×200	-3226	9.532	328.05	1.14	0.02	-1152.4	57.5	75.92	1.8	0.63	-1211.8	62.56	86.92	1.69	0.66
300×500	-541	25.86	510.6	1.22	0.02	-2300	162.28	64.46	1.51	0.75	-2924	177.43	114.99	2.02	0.99
800×1000	-818.6	60.79	734.41	1.136	0.008	-4208.6	433.314	69.008	1.772	0.84	-5977	405.82	75.05	1.75	1.016
1800×2000	-1465.6	371	1016.54	1.13	0.04	-8335.4	2082.49	71.26	1.57	0.87	-11331.2	1715.73	89.7	1.91	1.024
4000×5000	-3465.6	3764.89	1611.32	1.26	0.01	-12587.8	18171.5	124.95	2.08	0.9	-26690.8	12912.46	97.15	1.79	1.024

Table 1: *Computation results.*

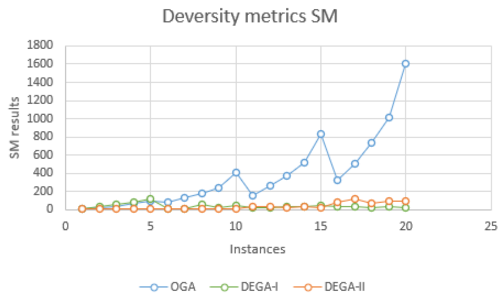


(a) *Φ results for examples.*

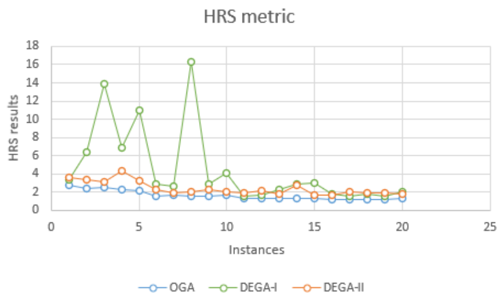


(b) *CPU time for examples.*

Figure 6: *Φ results and CPU time for examples.*



(a) *Diversity metric for examples.*



(b) *Hole relative size for examples.*

Figure 7: *Diversity metric and Hole relative size for examples.*

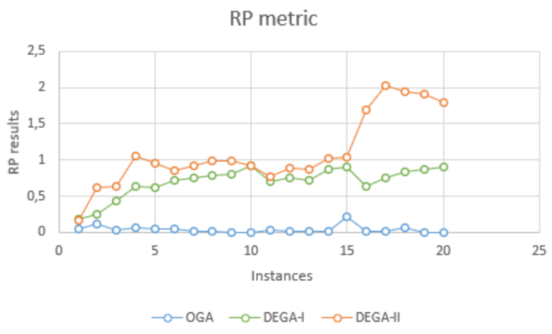


Figure 8: *Convergence metrics for examples.*

5. Conclusion

In this work, we studied the problem of optimizing the function over the efficient set of a multi-objective integer linear programming problem (MOILP). For this, we have adapted and proposed three variants of algorithms inspired by genetic algorithms: the first one called BVGA adapted to our problem and is based on the basic architecture of Genetic Algorithm, where the two last algorithms so-called DEGA-I and DEGA-II are obtained by removing the crossover operator in genetic algorithm and adapting the mutation operator. For DEGA-I the explored directions are determined from the first iteration and do not change; however, the explored directions of DEGA-II change at each iteration.

A numerical study was carried out by implementing those algorithms on the machine using Python. Examples of different sizes are designed and tested, and a comparison between those algorithms considering Φ value, CPU Time, SM, HRS and RP are performed. The results show that the DEGA-II obtained the best results nearly for all metrics.
From this work, several interesting ideas can be investigated, like adapting DEGA-II to resolve other kind of multi-objective problems (TSP, KP, ...) and applying it to other multi-objective non-linear problems.

References

[1] Abbas, M. and Chaabane, D. (2006). Optimizing a linear function over an integer efficient set. European Journal of Operational Research, 174(2), 1140–1161. doi: 10.1016/j.ejor.2005.02.072

[2] Boland, N., Charkhgard, H. and Savelsbergh, M. (2017). A New Method for Optimizing a Linear Function over the Efficient Set of a Multiobjective Integer Program. European Journal of Operational Research, 260(3), 904–919. doi: 10.1016/j.ejor.2016.02.037

[3] Chaabane, D., Brahmi, B. and Remdani, Z. (2012). The augmented weighted Tchebychev norm for optimizing alinear function over an integer efficient set of a multicriteria linear program. International Transactions in Operational Research, 19(4), 531–545. doi: 10.1111/j.1475-3995.2012.00851.x

[4] Chaabane, D. and Mebrek, F. (2014). Optimization of a linear function over the setof stochastic efficient solutions. Computational Management Science. 11, 157–178. doi: 10.1007/s10287-012-0155-1

[5] Chaabane, D. and Pirlot, M. (2010). A method for optimizing over the efficient set. Journal of Industrial and Management Optimization, 6(4), 811–823. doi: 10.3934/jimo.2010.6.811

[6] Ecker, J. G. and Song, H. G. (1994). Optimizing a Linear Function over an Efficient Set. Journal of Optimization Theory and Applications, 83(3), 541–563. doi: 10.1007/BF02207641

[7] Fonseca, C.M. and Fleming, P.J. (1993). Multiobjective genetic algorithms. Genetic Algorithms for Control Systems Engineering, IEEE, 83–90. Retrieved from: eden.dei.uc.pt

[8] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

[9] Hussain, A. and Cheema, S. A. (2020). A new selection operator for genetic algorithms that balances between premature convergence and population diversity. Croatian Operational Research Review, 11(1), 107–119. doi: 10.17535/crorr.2020.0009

[10] Jorge, M. J. (2009). An algorithm for optimizing a linear function over an integer efficient set. European Journal of Operational Research, 195(1), 98—103. doi: 10.1016/j.ejor.2008.02.005

[11] Katoch, S., Chauhan, S. and Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80, 8091-–8126. doi: 10.1007/s11042-020-10139-6

[12] Liu, Z. and Ehrgott, M. (2018). Primal and Dual Algorithms for Optimization over the Efficient Set. A Journal of Mathematical Programming and Operations Research, 67(10), 1–26. doi: 10.1080/02331934.2018.1484922

[13] Menni, A. and Chaabane, D. (2020), A possibilistic optimization over an integer efficient set within a fuzzy environment. RAIRO Operations Research, 54(5), 1437–1452. doi: 10.1051/ro/2019077

[14] Moulai, M. and Mekhilef, A. (2021), Quadratic optimization over a discrete pareto set of a multi-objective linear fractional program. A Journal of Mathematical Programming and Operations Research, 70(7), 1425–1442. doi: 10.1080/02331934.2020.1730834

[15] Philip, J. (1972). Algorithms for the vector maximization problem. Mathematical Programing, 2, 207–229. doi: 10.1007/BF01584543

[16] Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Grefenstette, J.J., et al. (Eds.) Genetic Algorithms and Their Applications, Proceedings of the 1st International Conference on Genetic Algorithms, 93–100, Lawrence Erlbaum, Mahwah.

[17] Teghem, J. and Kunsch, P. (1986). A survey of techniques for finding efficient solutions to multi-objective integer linear programming. Asia Pacific Journal of Operations Research, 3(2), 95–108.

[18] Van Veldhuizen, D. A. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations. Graduate School of Engineering. Air Force Institute of Technology, Wright Patterson AFB, Ohio, PhD thesis.

[19] Yamamoto, Y. (2004). Optimization over the Efficient Set: Overview. Journal of Global Optimization, 22, 285–317. doi: 10.1023/A:1013875600711