A Comparative Study of Variation Operators used for Evolutionary Multi-Objective Optimization

Isolina Alberto[a,*], Carlos A. Coello Coello[b], Pedro M. Mateo[c]

[a] *Depto. Métodos Estadísticos, Escuela de Ingeniería y Arquitectura*
Universidad de Zaragoza María de Luna 3, 50018 Zaragoza, Spain
[b] *CINVESTAV-IPN, Depto. de Computación*
Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07300
[c] *Depto. Métodos Estadísticos, Facultad de Ciencias*
Universidad de Zaragoza Pedro Cerbuna 12, 50009 Zaragoza, Spain

Abstract

In this paper, several variation operators based on Pareto efficiency, extracted from Differential Evolution, Estimation of Distribution Algorithms, Evolution Strategies and Evolutionary Programming, are compared in order to determine whether or not they increase the performance of the non-Pareto based versions. Firstly, we compare the selected variation operators in pairs, each operator with a modification of itself, in which we remove those elements related to the Pareto efficiency. Then, in a second experiment we compare among the selected operators, the variation operators used in the NSGA-II algorithm and the ones presented by the authors, PBCO and RBMO. In all the experiments the variation operators are incorporated in a well-known algorithm usually considered as a reference for making the comparisons, the NSGA-II algorithm. The experiments show that the Pareto based variation operators selected from the literature do not usually present a better behavior than their non-Pareto based versions; and none of them presents a better performance than the one reached by the variation operators defined by the authors, which were entirely built around the Pareto information of the individuals. These facts suggest that more effort should be placed in the design of variation operators devoted to

*Corresponding author
Email addresses: isolina@unizar.es (Isolina Alberto), ccoello@cs.cinvestav.mx (Carlos A. Coello Coello), mateo@unizar.es (Pedro M. Mateo)

multi-objective algorithms in order to achieve superior results to those obtained by means of general variation operators.

Keywords: Evolutionary Computation, Metaheuristics, Multi-objective Optimization.

---

1. Introduction

Evolutionary Algorithms, EAs, can be considered the most adequate methods for solving complex Multi-Objective Optimization Problems (MOOPs). The use of EAs for solving MOOPs was first hinted by Rosenberg in his PhD in the late 1960s [36], but it was not until 1984 that Schaffer implemented the Vector Evaluated Genetic Algorithm (VEGA) for his PhD thesis [37], the first actual implementation of what has been called a Multi-Objective Evolutionary Algorithm (MOEA). Since then, several different MOEAs have appeared. An extensive review on this matter can be obtained from [14, 55, 43]. For a general vision in this field the reader is referred to these classical books: [24, 23, 33, 21] and the survey [7].

In this work, we will study whether the use of variation operators that make use of the information on the quality of the solutions can lead, in general, to more efficient algorithms evaluated in terms of some quality measures of the efficient solution set found. As we have just stated, a significant number of papers have been devoted to the development of MOEAs since the publication of VEGA in 1984. Among all the MOEAs in the literature, NSGA-II [19] is worth highlighting because ten years after its publication it still serves as a basis for new developments and as a reference algorithm to be compared with new ones. It is for this reason that we have considered it as a candidate for developing our work.

Furthermore, we have selected several variation operators from the literature that explicitly use, in their functioning, information about the quality of the solution, in terms of Pareto optimality, to be altered. These selected operators to be incorporated come from the Differential Evolution (DE), Estimation

of Distribution Algorithms (EDA), Evolution Strategies (ES) and Evolutionary Programming (EP) fields as well as from the ones proposed by the authors in [3, 32].

Practically none of the original works where the selected operators were presented [27, 48, 47], try to deeply verify if the operator proposed really represents an improvement with respect to other operators with similar characteristics but without using the information on Pareto efficiency or Pareto optimality. The only exception is [27], who compared their proposals DS, DC and DSC with NSDE [26], a previous version from the same authors that did not use the multi-objective characteristics of the problem in the design of the DE operator. More specifically, the developments made by their authors in the selected works are the following: In [48], the algorithm was compared with SPEA [57] providing means and variances of the generational distance [14, pp. 256-257], between the populations obtained and PF^*, but no detailed statistical analysis of the results was carried out. In [27], as we have previously said, they compared the three versions they proposed with the basic NSDE, and also with NSGA-II, using the measures Generational Distance (GD) and Inverse Generational Distance (IGD),[14, p. 257]. The authors concluded that all the DE, including the basic version, outperformed NSGA-II and there were always some of their proposals that outperformed the basic version. Among the three, the ones that stood out were those that used a component which tried to reinforce the directional spread, DS and DSC. The conclusions of the authors were obtained in view of the evolution plots of the measures GD and IGD in the different problems, and there was no statistical study. Finally, in PDE-PEDA, [47], the authors compared their proposal with a basic DE algorithm, supposedly similar to NSDE, and also with NSGA-II. For a certain set of test problems and with only 10 executions for each problem, they concluded, in view of the values of the means and variances of GD and spread, that the algorithm they proposed outperformed the two others.

As [22] stated, since in the majority of the situations the comparison of the behavior of the algorithms cannot be accomplished by means of theoretical

results, we have to focus our efforts on the analysis of empirical data obtained from the experiments carried out. Only a small set of works use statistical procedures in order to compare results, although their use has grown recently and it is being suggested as a need for many reviewers. Considering all these facts, in this work we propose a more rigorous comparison of the selected operators in order to study the possible increment in the performance of the operators when taking into account the Pareto information. We also consider the comparison among all the selected operators, those defined by the authors in [3, 32] and those from the algorithm NSGA-II.

The paper is organized as follows: In the next section we will briefly present the definitions and notations on Multi-Objective Problems and Evolutionary Algorithms that we will need later. In the third section, the selected variation operators are shown. Later, in Section 4, we display the basic elements needed for the computational experiment carried out in Section 5 for the comparison among the different implementations. Finally, in Section 6, the conclusions and further research are commented on.

2. Multi-Objective Optimization and Evolutionary Algorithms

The aim of Multi-Objective Optimization is to optimize a set of objective functions which may generally be of a conflicting nature. Hence, the term "optimize" means to find a solution satisfying the constraints, which would give reasonable values of all objective functions to the decision maker. More formally, Multi-Objective Optimization Problems, MOOPs, can be defined in the following way:

$$\begin{aligned} \min \quad & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})), \\ s.t. \quad & \boldsymbol{x} = (x_1, \ldots, x_n) \in D \subset \mathbb{R}^n. \end{aligned}$$

Contrary to single objective optimization, in multi-objective optimization it is usually impossible to find one optimal solution. Instead, algorithms for optimizing multi-objective problems try to find a family of points known as the *Pareto optimal set*. These points verify that there is no different feasible

solution which strictly improves one component of the objective function vector without worsening at least one of the remaining ones.

A more formal definition of Pareto optimality or Pareto efficiency is the following:

Definition 1. *If, given a solution* $\boldsymbol{y}$*, there exists another solution* $\boldsymbol{x}$ *such that* $\forall j = 1, \ldots, m$ $f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$ *and* $\exists j \in \{1, \ldots, m\}$ *such that* $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$*, then we will say that solution* $\boldsymbol{x}$ *dominates solution* $\boldsymbol{y}$ *(denoted by* $\boldsymbol{x} \prec \boldsymbol{y}$*), and, obviously, solution* $\boldsymbol{y}$ *will never be sensibly selected as the solution to the problem. If* $f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$*,* $\forall j$*, we will say that solution* $\boldsymbol{x}$ *weakly dominates solution* $\boldsymbol{y}$ *and will be denoted by* $\boldsymbol{x} \preceq \boldsymbol{y}$*.*

Definition 2. *A solution* $\boldsymbol{x} \in D$ *is said to be Pareto optimal or efficient if and only if* $\nexists \boldsymbol{y} \in D$ *such that* $\boldsymbol{y} \prec \boldsymbol{x}$*.*

Definition 3. *The real Pareto optimal set will be denoted with* P^{true}*. The image of* P^{true} *in the objective function space is called Pareto front and it will be denoted by* PF^{true}*.*

As we have stated in Section 1, we have selected NSGA-II in order to develop new implementations by incorporating the variation operators we have chosen. NSGA-II is based on the original version of the Non-dominated Sorting Genetic Algorithm, NSGA, presented in [40]. This second version was proposed with the aim of reducing the computational complexity of the non-dominated sorting, and incorporating the concept of elitism which was supposed to speed up the performance of the algorithm. The main differences between these two versions are the fast non-dominated sorting approach, the estimation of the density of solutions around a particular point of the population and the crowding comparison operator, thus solving the main criticisms NSGA received. The pseudo-code of NSGA-II, extracted from [16, pp. 234-235], is shown in Fig. 1.

The crossover and mutation operators that NSGA-II uses are the Simulated Binary Crossover [17] and the Polynomial Mutation operator [18]. They will be briefly explained in Section 3. For more specific details, readers are encouraged

Step 0: A Random population P^0 of size N is created and evaluated. Then the crowded tournament selection operator is used for obtaining a sample from P^0. The individuals of this sample are recombined and mutated to create an offspring population Q^0 of size N. $t = 0$.

Step 1: Let $R^t = P^t \cup Q^t$. Obtain the domination fronts $\mathcal{F}_i, i = 1, 2, \ldots$ [16, pp. 40-44].

Step 2: Set the new population $P^{t+1} := \emptyset$ and $i = 1$. Until $|P^{t+1}| + |\mathcal{F}_i| < N$ do $P^{t+1} := P^{t+1} \cup \mathcal{F}_i$ and $i := i + 1$.

Step 3: Calculate the crowding distance [16, p. 236] of the individuals in $\mathcal{F}_i$ and include in P^{t+1} the $(N - |P^{t+1}|)$ ones with the greatest value.

Step 4: Create offspring population Q^{t+1} from P^{t+1} by using the crowded tournament selection, crossover and mutation operators. $t := t+1$ and go to Step 1.

Figure 1: Pseudo-code of NSGA-II.

to refer to the original study [19] or the books [16, pp. 233-241] and [14, pp. 91-94].

3. The Variation operators

In this section we briefly present the different variation operators we have considered. We will carry out a short description of them and we will make reference to the original papers in order to obtain a more detailed description.

In order to select the operators, we have taken into consideration the following facts: First of all, they take explicitly into consideration the quality of the solutions, measured in terms of Pareto efficiency; secondly, the algorithms in which the operators were involved in should be, as far as possible, in the framework of the algorithm NSGA-II; and thirdly, the computational requirements of the operators should be comparable. The second requirement means all the operators being compared under similar conditions to the ones that the authors established in their original papers; this fact should guarantee as good performance as the one that their authors claimed. The third element also tries to establish a fair comparison by ensuring that all the algorithms have similar complexities. In the particular case of the EDA algorithms, we found several works

that finally were not included in our study since they use very sophisticated and elaborated mathematical and statistical tools such as Bayesian networks, principal component analysis, splines,Voronoi diagrams, etc. [35, 54, 6, 53, 28] and their computational requirements are superior to the other algorithms considered.

The complexity of one iteration of all the NSGA-II "variations" that we will propose in Section 4.4 will maintain a similar order as the original NSGA-II, i.e., $O(m \cdot N^2)$. This complexity is because of the non-dominated sorting process [16, pp.42-43]. The EDA and PBV versions will include additional terms. EDA has the term $n \cdot N \cdot \log N$, as a consequence of sorting the n components of the solutions in the population of size N and PBV has the term $n \cdot N^2$ due to the calculation of the mean and standard deviation of the distances between solutions. In both cases, it could mean a slight increment due to the factor n instead of m; however, the order of the number of variables and objectives in the majority of the multi-objective optimization problems can be considered equal.

In the following five subsections, we will briefly present the original variation operators used by NSGA-II and the different variation operators we have selected: Two variation operators based on DE, one based on EDAs, one based on Gaussian/Cauchy variation and the ones proposed by the authors.

In the definition of the operators we will use a ranking function, denoted by $r(\cdot)$, which corresponds with the ranking based on Pareto layers proposed in [23, p. 201]. This function takes as ranking the Pareto layer in which the solution is placed, i.e., given an individual $\boldsymbol{x}^{(t)} \in P^t$ belonging to the ith layer, then $r(\boldsymbol{x}^{(t)}) = i$.

3.1. The original operators of NSGA-II

As we mentioned at the end of Section 2, the crossover and mutation operators NSGA-II implements are the Simulated Binary Crossover and the Polynomial Mutation operators.

SBX: Simulated Binary Crossover [17]: This operator simulates the behavior of the single-point crossover operator on binary strings. Given $\boldsymbol{x}^{(1,t)}, \boldsymbol{x}^{(2,t)}$

to be recombined, it generates the ith component, $i = 1, \ldots, n$, of the offsprings as follows:

$$x_i^{(1,t+1)} = 0.5 \cdot \left[(1+\beta_i) \cdot x_i^{(1,t)} + (1-\beta_i) \cdot x_i^{(2,t)}\right],$$
$$x_i^{(2,t+1)} = 0.5 \cdot \left[(1-\beta_i) \cdot x_i^{(1,t)} + (1+\beta_i) \cdot x_i^{(2,t)}\right],$$

where

$$\beta_i = \begin{cases} (2 \cdot u)^{\frac{1}{\eta_c+1}}, & \text{if } u \leq 0.5, \\ \left(\frac{1}{2 \cdot (1-u)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise,} \end{cases}$$

and u is a random number in $[0, 1]$. The parameter η_c determines how well spread the children will be from their parents.

PMO: Polynomial Mutation operator [18]: This mutation operator uses a polynomial distribution in the following way:

$$x_i^{(t+1)} = x_i^{(t)} + \left(x_i^{(U)} - x_i^{(L)}\right) \cdot \delta_i,$$

$i = 1, \ldots, n$ where $x_i^{(U)}$ and $x_i^{(L)}$ are the upper and lower bounds, respectively, for component x_i,

$$\delta_i = \begin{cases} (2 \cdot u)^{1/(\eta_m+1)} - 1, & \text{if } u < 0.5, \\ 1 - [2 \cdot (1-u)]^{1/(\eta_m+1)}, & \text{if } u \geq 0.5, \end{cases}$$

u is a random number in $[0, 1]$ and η_m is the mutation distribution index.

We have considered the value of η_c and η_m equal to 20 as in the original version of NSGA-II in [19].

3.2. The Differential Evolution operators

Differential Evolution (DE) was originally developed for single-objective optimization in continuous search spaces in [41] by Storn and Price and it has shown good convergence properties for single-objective optimization problems. An updated survey about DE is provided in [15]. DE performs variation based on the location of the solutions in the population instead of using a probability density function as other evolutionary methods usually do. With regard to MOOPs, [9] proposes the first attempt at extending DE for multi-objective

problems and [2, 1] are algorithms usually commented on in the literature. In these proposals the information on the efficiency of the solutions is not explicitly used in the description of their variation operators, but to conform the set of individuals which are involved in the reproduction process. On reviewing the literature we can find some papers that use the information on the efficiency of the solutions in the variation process [48, 27, 5, 11]. We have chosen two of them to be included in our study. The first one, MODE, was originally presented in [48], extended later in [49] and applied in an interesting work in the context of the printed circuit board assembly industry in [50]. This algorithm is a slight variation of NSGA-II, in which, apparently, the parent population is always the whole population. They also include an additional parameter, σ_{crowd}, to reduce the fitness assignment of very similar individuals. In our implementation we have intentionally omitted this last parameter in order to ensure that the core of all versions is as similar as possible. The second selected paper is [27]. It proposes three NSGA-II versions that use DE variation operators. We have selected the one their authors named NSDE-DCS, where DCS stands for "Directional Convergence and Spread", because in their study it shows equal to or better performance than the other two. We will now briefly present both operators. The solution quality information used by these two operators consists of selecting the different solutions for building the differential and/or perturbation vectors among those that fullfil certain efficiency relations related with the solutions to be varied.

MODE: This mutation operator consists of two components: the differential and the perturbation vectors. The differential part is the vector defined between the individual $\boldsymbol{x}^{(best,t)}$ and the individual to be mutated $\boldsymbol{x}^{(i,t)}$. Individual $\boldsymbol{x}^{(best,t)}$ is a randomly selected individual from the set of non-dominated individuals of the parent population (i.e., individuals for which $r(\boldsymbol{x}) = 1$) that dominate $\boldsymbol{x}^{(i,t)}$ (if $r(\boldsymbol{x}^{(i,t)}) = 1$, then the differential part is $\boldsymbol{0}$). On the other hand, the perturbation vectors are defined by randomly choosing pairs of individuals from the parent population. Given an

individual $\boldsymbol{x}^{(i,t)}$ to be altered, the new individual is obtained according to:

$$\boldsymbol{x}^{(i,t+1)} = \boldsymbol{x}^{(i,t)} + F \cdot \sum_{k=1}^{K} (\boldsymbol{x}^{(i_a^k,t)} - \boldsymbol{x}^{(i_b^k,t)}) + \mathbf{I}_{\boldsymbol{x}^{(i,t)}} \cdot \gamma \cdot (\boldsymbol{x}^{(best,t)} - \boldsymbol{x}^{(i,t)}), \tag{1}$$

where $\mathbf{I}_{\boldsymbol{x}^{(i,t)}} = 1$ if $r(\boldsymbol{x}^{(i,t)}) \neq 1$ and 0 otherwise, $\gamma \in [0,1]$ represents the greediness of the operator, K is the number of perturbation vector pairs, F is the factor scale of the perturbation, and $\boldsymbol{x}^{(i_a^k,t)}$ and $\boldsymbol{x}^{(i_b^k,t)}$ are randomly selected distinct individuals from the parent population, which may be different for each mutated individual. The values for the parameters were set to $K = 2$, $\gamma = 0.7$ and $F = 0.5$ as the authors proposed in their original paper [48]. After the mutation phase, a crossover operator is applied using the following expression:

$$x_j^{(i,t+1)} = \begin{cases} x_j^{(i,t+1)}, & \text{if } u_j \leq CR \text{ or } j = j_i, \\ x_j^{(i,t)}, & \text{otherwise.} \end{cases} \tag{2}$$

In this expression, u_j is a random uniform value in the interval $[0,1]$ and $CR \in [0,1]$ is the crossover parameter which has been fixed to 0.3, as also proposed in [48]. The j_i value is a randomly chosen index component in $\{1, 2, \ldots, n\}$ that ensures that at least one component of $\boldsymbol{x}^{(i,t)}$ is changed.

DCS: This second operator was introduced in [27]. In order to apply this operator, three individuals have to be selected in the following way: Select $\boldsymbol{x}^{(i_a,t)}$ and $\boldsymbol{x}^{(i_b,t)}$ at random such that they belong to the same Pareto layer, $(r(\boldsymbol{x}^{(i_a,t)}) = r(\boldsymbol{x}^{(i_b,t)}))$. Then, select another solution $\boldsymbol{x}^{(i_c,t)}$ such that it belongs to a higher layer than the solution to be mutated $(r(\boldsymbol{x}^{(i_c,t)}) < r(\boldsymbol{x}^{(i_a,t)}))$. Furthermore, it has to hold that $\boldsymbol{x}^{(i_a,t)} \neq \boldsymbol{x}^{(i_b,t)} \neq \boldsymbol{x}^{(i_c,t)} \neq \boldsymbol{x}^{(i,t)}$.

With the selected individuals, individual $\boldsymbol{x}^{(i,t)}$ is mutated according to the

following equation:

$$\begin{aligned} \boldsymbol{x}^{(i,t+1)} = \boldsymbol{x}^{(i,t)} \quad &+ \quad K \cdot (\boldsymbol{x}^{(i_c,t)} - \boldsymbol{x}^{(i,t)}) \\ &+ \quad F \cdot (\boldsymbol{x}^{(i_a,t)} - \boldsymbol{x}^{(i_b,t)}), \end{aligned} \tag{3}$$

where as before, F and K are scale factors fixed to 0.8 and 0.4, respectively, as the original authors use in [27]. If $r(\boldsymbol{x}^{(i,t)}) = 1$ then $\boldsymbol{x}^{(i_c,t)}$ is not selected and the term $K \cdot (\boldsymbol{x}^{(i_c,t)} - \boldsymbol{x}^{(i,t)})$ is not applied. After the mutation, a crossover operator is applied in the same way as shown in (2).

3.3. The Estimation of Distribution Algorithm operators

The Estimation of Distribution Algorithms, EDAs, originally proposed in [4, 34] for the discrete case and extended for continuous variables in [39], are a relatively new adopted technique derived from evolutionary algorithms, the main difference between them being that a distribution on the space of solutions is evolved instead of a population in the space of solutions; the usual crossover and mutation operators are substituted by the sampling of a distribution previously learned from a set of selected individuals. A complete revision of the field can be obtained from [29, 30]. As in the case of DE, EDAs were originally developed for single objective optimization but later on they were proposed for the multi-objective case [45, 44, 35, 54, 6, 53, 47, 28], etc. A special mention of [28] should be made. Such paper shows an EDA based on Bayesian networks, whose main novelty is that it incorporates into the model the variables as well as the information of the objective functions, i.e., the quality of the solutions. However, in spite of this, we have not considered it due to its high computational requirements. Among the remaining alternatives, we have chosen the one by Wang et al. [47] to be included in our study because of its simplicity and similar computational requirements to those of the other variation operators considered. The PDE-PEDA algorithm proposed in [47] follows the scheme of NSGA-II, with a variation operator based on EDAs. More specifically, it presents an EDA variation operator which is hybridized with a standard DE variation operator. With a certain probability p_r, the EDA variation is used

and with a probability $1-p_r$, DE is adopted instead. Probability p_r is adjusted using the following process:

$$p_r^0 = p_r^{\max},$$
$$p_r^{t+1} = p_r^{\min} + \beta \cdot (p_r^t - p_r^{\min}).$$

The following values are fixed in the paper: $p_r^{\min} = 0.2$, $p_r^{\max} = 0.9$, $\beta = 0.95$.

As in the first DE operator, MODE, the set of parents to be altered is the current population. After obtaining a random uniform value in (0,1), if it is greater than p_r, individual $\boldsymbol{x}^{(i,t)}$ is mutated according to the following DE scheme:

$$\boldsymbol{x}^{(i,t+1)} = \boldsymbol{x}^{(i_a,t)} + F \cdot (\boldsymbol{x}^{(i_b,t)} - \boldsymbol{x}^{(i_c,t)}),$$

where $\boldsymbol{x}^{(i_a,t)} \neq \boldsymbol{x}^{(i_b,t)} \neq \boldsymbol{x}^{(i_c,t)} \neq \boldsymbol{x}^{(i,t)}$. Then, the crossover operation is applied according to (2). In the paper, the values of the parameters were taken as $F = 0.3$ and $CR = 0.3$.

When the random value is less than or equal to p_r, an EDA variation operator is applied. The authors proposed building a histogram as the probabilistic distribution model because it is the most straightforward method for estimating the probabilistic density. In particular, they use a fixed-height histogram (FHH) [46]. For each component x_i, its search space $[x_i^{(L)}, x_i^{(U)}]$ is divided into H bins (subintervals), each bin having the same height and, therefore, each bin containing the same number of individuals. As a consequence, dense regions have narrower bins than the others. The histogram of each component is built using the information of the efficient solutions of the current population (set of parents). Once the histograms are built, and in order to obtain a value for component x_i, first, a bin, $h_s, s = 1, \ldots, H$, is selected with a probability $1/H$ and then a random value is obtained in the bin according to a uniform distribution. In the implementation, we have considered the number of bins, H, equal to 20, as the authors did in [47].

Given that all the other variation operators are not hybridized versions and, in order to obtain a fair comparison among the different variation operators, only the EDA part of this variation operator will be initially taken into consideration

in the comparisons.

3.4. The Gaussian/Cauchy operators

Gaussian and Cauchy mutation operators are variation operators widely used in Evolution Strategies (ES) and Evolutionary Programming (EP), in both the single and the multi-objective cases. Furthermore, in several papers they are used together in order to evolve solutions in EAs [10, 51, 20, 12]. However, unlike the other variation operators, we do not modify any of the algorithms proposed; instead, we design a variation operator based on their ideas. We do this because the algorithms we found which use these operators are not variations of NSGA-II, as happens to the other algorithms. So we propose a variation operator which combines a usual ES crossover and a mutation process which depends on the origin of the parents of the individual. If any of the parents are efficient, then the Gaussian mutation is used (exploitation). Otherwise, we use the Cauchy mutation. The underlying idea is similar to the one proposed in [52, 12], but unlike them we do not apply both mutation operators one after the other. The crossover operator that we use is one of a local type with intermediate recombination for the strategy parameters, $\boldsymbol{\sigma}$, and discrete recombination for variables, $\boldsymbol{x}$, [21, p. 81]. In this case, given two individuals $(\boldsymbol{x}^{(1,t)}, \boldsymbol{\sigma}^{(1,t)}), (\boldsymbol{x}^{(2,t)}, \boldsymbol{\sigma}^{(2,t)})$ to be recombined, the operator generates the ith component, $i = 1, \ldots, n$, of the offspring as follows:

$$\begin{aligned} x_i^{(t+1)} &= (x_i^{(1,t)} + x_i^{(2,t)})/2, && \text{intermediate recombination,} \\ \sigma_i^{(t+1)} &= \sigma_i^{(1,t)} \text{ or } \sigma_i^{(2,t)}, && \text{discrete recombination.} \end{aligned}$$

We then carry out the mutation process whose mechanism is the following:

$$\sigma_i^{(1\_2,t+1)} = \sigma_i^{(t+1)} \cdot e^{(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))},$$

$$x_i^{(1\_2,t+1)} = x_i^{(t+1)} + \begin{cases} \sigma_i^{(1\_2,t+1)} \cdot N_i(0,1), & \text{if } r(\boldsymbol{x}^{(1,t)}) = 1 \text{ or } r(\boldsymbol{x}^{(2,t)}) = 1, \\ \sigma_i^{(1\_2,t+1)} \cdot C_i(0,1), & \text{otherwise,} \end{cases}$$

where $N(0,1)$ and $N_i(0,1)$ stand for a random number drawn from a Gaussian distribution with zero mean and standard deviation 1, $C_i(0,1)$ denotes a random

value from a Cauchy distribution centred at 0 with a scale parameter equal to 1. τ and τ' represent a kind of learning rate and they are fixed to $1/\sqrt{2 \cdot \sqrt{n}}$ and $1/\sqrt{2 \cdot n}$, respectively, according to [38]. Furthermore, a boundary rule is applied to prevent standard deviations from being very close to zero: if $\sigma_i^{(t)} < \epsilon_i \Rightarrow \sigma_i^{(t)} = \epsilon_i$. We have established $\epsilon_i = 0.001,\ i = 1, \ldots, n$.

3.5. Our operators

Finally, we shall comment on the operators we propose as substitutes for the original ones of NSGA-II. They are a crossover operator, named PBCO, and a mutation operator, named RBMO. Both were proposed in [3, 32]. They are devoted to both exploitation and exploration, depending on the quality of the solution. The main idea of the operators is to use good parents (efficient ones) to improve the quality of the offspring (exploitation) and to use not so good parents (non-efficient ones) to explore the whole space (exploration). The outlines of these Pareto-based operators are the following.

RBMO: Given a solution $\boldsymbol{x}^{(t)}$ to be mutated, its components are mutated with a certain mutation probability according to the expression:

$$x_i^{(t+1)} = x_i^{(t)} + 2 \cdot \delta^{(t,r)}(x_i^{(U)} - x_i^{(L)}) \cdot (u - 0.5),$$

$i = 1, \ldots, n$, where u is a random number in $[0, 1]$, $\delta^{(t,r)}$ defines the maximal amplitude of the mutation based on the iteration t and the ranking r of the solution $\boldsymbol{x}^{(t)}$ according to the following expression:

$$\delta^{(t,r)} = \begin{cases} 2 \cdot \epsilon(t) + 10^{-4} \cdot [1 - \epsilon(t)], & \text{if } r_{\max}^t = 1, \\ \epsilon(t) + [1 - \epsilon(t)] \cdot \lambda(r), & \text{if } r_{\max}^t \geq 2, \end{cases}$$

where

$$\epsilon(t) = 0.25 \cdot \left(\frac{t_{\max} - t}{t_{\max}} \right)^2$$

and

$$\lambda(r) = \left(\frac{1 - e^{-\beta \cdot (r-1)}}{1 - e^{-\beta \cdot (r_{\max}^t - 1)}} \right)$$

with $t_{\max}$ the maximum number of iterations, $r_{\max}^t$ the maximal ranking among the elements of the current population and $\beta = 0.1$.

PBCO: This is a uniform crossover operator which has a different capacity of variation depending on the quality of the solution and the distance between the parents. First of all, the mean distance and standard deviation between all pairs of efficient solutions in the current population is calculated, $d^{(eff,t)}$ and $\sigma^{(eff,t)}$, respectively. Given $\boldsymbol{x}^{(1,t)}$ and $\boldsymbol{x}^{(2,t)}$ to be recombined, we define $\Delta_i = |x_i^{(1,t)} - x_i^{(2,t)}|$ and $\xi_i^h = h \cdot \mathbf{I}_{[u_i<0.75]} + (3-h) \cdot \mathbf{I}_{[u_i \geq 0.75]}$, where $\mathbf{I}$ is the indicator function and u_i is a random value in $(0,1)$. Then, the ith component, $i = 1, \ldots, n$, of the offsprings, $\boldsymbol{x}^{(1,t+1)}$ and $\boldsymbol{x}^{(2,t+1)}$, are obtained depending on the quality of the parents:

- If $r(\boldsymbol{x}^{(1,t)}) \neq 1$ and $r(\boldsymbol{x}^{(2,t)}) \neq 1$, the components $x_i^{(h,t+1)},\ h = 1, 2$ are obtained by generating two values in:

$$\left[\frac{(x_i^{(1,t)} + x_i^{(2,t)})}{2} \pm \Delta_i \cdot f^{Neff} \right].$$

- If $r(\boldsymbol{x}^{(1,t)}) = r(\boldsymbol{x}^{(2,t)}) = 1$, the components are randomly obtained in:

$$\left[x_i^{(\xi_i^h,t)} \pm \frac{3}{4} \cdot \Delta_i \cdot f^{(eff,t)} \right],\ h = 1, 2.$$

- If only $r(\boldsymbol{x}^{(1,t)}) = 1$, for instance, then, the components of the first solution are randomly sampled in:

$$\left[x_i^{(\xi_i^1,t)} \pm \frac{3}{4} \cdot \Delta_i \cdot f^{(eff,t)} \right],$$

  and for the second in:

$$\left[x_i^{(\xi_i^2,t)} \pm \frac{3}{4} \cdot \Delta_i \cdot f^{Neff} \right].$$

$f^{(eff,t)}$ and f^{Neff} are correction factors which increase or decrease the amplitude of the intervals in which the components are sampled depending on the distance between the parents in relation with the mean, $d^{(eff,t)}$, and standard deviation, $\sigma^{(eff,t)}$, of the efficient solutions in the current population (see [3] for more details).

4. Methodology

In this section we present the necessary elements used in the experiments we have carried out: The test problem suite, the measure used for carrying out the comparisons, the parameter setting of the algorithms and the decision making process.

4.1. Test problems

In order to guarantee that the proposed algorithms will confront efficient solution sets of different characteristics, we have used a wide set of test problems proposed in [25]. The set of test problems we have selected is wider than those used in the papers from where we have taken the different operators, with the aim of being able to accomplish a more detailed study.

A short description of the test suite used is shown in Table 1, which has been obtained from the work mentioned. In all of these test problems the Pareto optimal set is known.

4.2. Measure for comparing the populations

In order to compare the performance of the different implementations we have selected the Hypervolume Difference (HD). This measure is based on the widely renowned hypervolume indicator or S measure [56], which is the only indicator that holds the properties of a metric and the only one to be strictly Pareto monotonic [58]. Given P^* and PF^* the set of non-dominated solutions resulting from the execution of an optimization algorithm and its image in the objective space, respectively, the hypervolume measure calculates the volume covered by the hypercube formed by the vectors of PF^* (considering that all the objectives are to be minimized) and a reference point (one at least weakly dominated by every member in P^*). To obtain the value of HD, the difference between the hypervolume of a reference set contained in PF^{true} and the hypervolume of PF^* is calculated. [42] provides reference sets for all the problems considered. In the experiments carried out, for each problem and number of iterations, the reference point is placed by obtaining the worst objective function

Table 1: Properties of the test functions. S: Separable; NS: Nonseparable; U: Unimodal; M: Multimodal; D: Deceptive.

Test problem	Objective functions	No. of variables	Total no. of variables	Separability and modality	Geometry
1. S-ZDT1	f_1	1	30	S, U	convex
	f_2	29		S, U	
2. S-ZDT2	f_1	1	30	S, U	concave
	f_2	29		S, U	
3. S-ZDT4	f_1	1	30	S, U	convex
	f_2	29		S, M	
4. R-ZDT4	$f_{1:2}$	10	10	NS, M	convex
5. S-ZDT6	f_1	1	30	S, M	concave
	f_2	29		S, M	
6. OKA2	f_1	1	3	S, U	concave
	f_2	3		NS, M	
7. SYMPART	$f_{1:2}$	30	30	NS, M	concave
8. S-DTLZ2	$f_{1:3}$	10	10	S, U	concave
9. R-DTLZ2	$f_{1:3}$	10	10	NS, M	concave
10. S-DTLZ3	$f_{1:3}$	10	10	S, M	concave
11. WFG1	$f_{1:3}$	24	24	S, U	mixed
12. WFG8	$f_{1:3}$	24	24	NS, U	concave
13. WFG9	$f_{1:3}$	24	24	NS, M, D	concave

value among all the P^* populations obtained and then all data are normalized so as to obtain the value of HD.

4.3. Parameter setting

Since our experiment is based on variations of the classical NSGA-II, to carry out the experiments later shown, we have considered a usual fixed population size equal to 100 individuals as in the original paper of NSGA-II, [19]. For the parameters used in the different variation operators we have considered the values proposed by their authors in the corresponding papers, they are shown in Table 2. The initial values of the variables for all implementations are obtained from a uniform distribution in $(x_i^{(L)}, x_i^{(U)})$ and the initial values for the σ_i parameters of the Gaussian/Cauchy implementations are obtained from a uniform distribution in $(0, (x_i^{(U)} - x_i^{(L)})/\sqrt{12})$. For all the variation operators,

if any of the individuals obtained is not feasible, another one is generated in the same way.

Table 2: Parameters of the algorithms.

NSGA-II	$\eta_c = 20$, $\eta_m = 20$, $pr = 0.9$, $pm = 1/n$
DE operator of MODE	$F = 0.5$, $\gamma = 0.7$, $K = 2$, $CR = 0.3$
DE operator of NSE-DCS	$F = 0.8$, $K = 0.4$, $CR = 0.3$
EDA operator of PDE-PEDA	$H = 20$
Gaussian/Cauchy operator	$\epsilon_i = 0.001$

4.4. The decision making process

In this section we present the decision making process accomplished for deciding which operator presents better results than the others. We will use the names shown in column 2 of Table 3 to identify the variations of the NSGA-II that we have developed when introducing the different operators previously mentioned. Since in the first experiment we test whether the use of the Pareto information improves the performance of the variation operators or not, in the third column we present the name of the version when the Pareto elements are not considered. The way in which the non-Pareto versions of the operators are accomplished is the following:

- **DE1**: In this case, all the solutions are mutated as if they were efficient, i.e., no differential part is considered (DE1-nPb, non-Pareto based, in Table 3).
- **DE2**: In this case, the three solutions are selected at random, i.e., there is no selection based on the Pareto layers (DE2-nPb).
- **EDA**: The histogram of each component is built using the information of solutions of the current population instead of using only the information of the efficient ones (EDA-nPb).
- **GC**: A value from a uniform distribution in (0,1) is drawn. Then, if it is less than 0.5 the Gaussian mutation operator is used; otherwise the

Cauchy mutation operator is (GC-nPb).

It is worth mentioning that, after these modifications, some of the non-Pareto versions become almost random search procedures. Particularly, DE1-nPb and EDA-nPb do not have parent selection for reproduction based on the quality of the solutions and then, they do not use Pareto information in their functioning, except for the final survival selection process. But, surprisingly, one of them has a better performance than its original version, as we will show in the first experiment. There is no version of our Pareto Based Variation (PBV) operators because they are totally built around the Pareto efficiency or Pareto ranking information and if we remove these elements the operators cannot be defined.

Table 3: List of variations of NSGA-II implemented with and without their Pareto elements

Variation operators used	Name of version	Name of modified version
SBX and PMO	NSGA-II	/
PBCO and RBMO	PBV	/
DE operator of MODE	DE1	DE1-nPb
DE operator of NSE-DCS	DE2	DE2-nPb
EDA operator of PDE-PEDA	EDA	EDA-nPb
Gaussian/Cauchy operator	GC	GC-nPb

In all the experiments carried out the aim is to compare the performance of the operators for solving MOOPs. The decision rule for determining which operator is better than the others is a very complicated problem which can be tackled in several ways. One possibility is to perform a statistical test to compare the means or the medians of the measure selected; another is to count the number of times that each operator outperforms the others (with respect to the measure selected) and to determine whether those counts can be considered statistically equivalent or not. Both methodologies present advantages and disadvantages: In the first case, a few of very large or very small values can distort the results, even using a test for the medians; in the second case, we cannot know to what extent one operator is better than the other. In view of this, and taking into account the advantages and disadvantages of those methodologies, we

finally opted for the second alternative, and we calculated the number of times that each algorithm is the best one, with respect to HD. This index intends to summarize the performance of each algorithm with regard to its peers. We then performed statistical tests with these values. However, the results obtained did not differ much from comparing the means or the medians.

With all these, for obtaining the information with which we made the comparisons, the process was the following. For each version of Table 3, once we fixed the problem number ($P_1, P_2, \ldots, P_{13}$) and the number of iterations (100, 1000, 3000), we executed the algorithm 50 times, starting from 50 different initial populations. After the evolution process, we obtained, for each version of Table 3, 50 final populations for each problem and number of iterations. We then calculated the values of measure HD on the final efficient solution sets obtained.

As we have previously mentioned we carried out three experiments. In the first experiment we contrasted the implementations in pairs: the original form with its non-Pareto version. In a similar way as in [31] we calculated the "performance index" as the number of times that each version is better than the other with respect to HD, taking into account that for HD, the smaller, the better. Since trying to understand all this information can be unmanageable, and according to [31], this information is summarized per problem and per number of iterations. In both cases, and with a significance level $\alpha = 0.05$, we set the hypothesis test of a proportion, [8], $H_0 : p = 0.5$ versus $H_1 : p \neq 0.5$, where p is the probability that the original version is better than or equal to the non-Pareto one. If the null hypothesis was not rejected, we can say that both versions are equivalent. As we will show in Section 5, depending on the sample size, the acceptance region of the test will be expressed in terms of the number of times that the original version is equal to or better than the non-Pareto one.

In the second experiment we considered the selected versions, DE1, DE2, EDA and GC together with the original NSGA-II and PBV. In a similar way to the first experiment, for each problem and number of iterations we calculated the performance index of each algorithm as the number of times that each

algorithm presents the best value of the HD measure. After obtaining these values, we carried out the multiple comparison test for proportions, [8], in order to decide if the obtained values of the performance index could all be considered statistically equivalent or not. As it will be shown in Section 5, since the multiple comparisons of the values always rejected the null hypothesis, we compared these values in pairs performing the χ^2 goodness-of-fit test with the multiple-comparison correction of Bonferroni, [8], with a global significance level equal to 0.05.

5. Results of the experiments

In this section, we present the results of the three experiments that we carried out. The first one was devoted to studying whether the use of the information of the quality of the solutions, measured in terms of Pareto efficiency or Pareto layers to which the solutions belong to, as Goldberg suggested in [23, p. 201], really improves the behavior of the same operator from which we have removed its Pareto elements. In the second experiment, we studied the behavior of all the operators we have described in sections 3.1 to 3.5, both included. In the third one we performed two comparisons including the complete version of the EDA operator [47].

5.1. Experiment 1: Pareto based operators versus non-Pareto based operators

In this first experiment, we compared the selected operators in pairs: in their original form and in an equivalent one but without the Pareto information, so that we could study the possible increment in the performance of the operators when taking into account the Pareto information.

As we have mentioned in Section 4.4, the information of the performance index is summarized per number of iterations and per problem. In Table 4, we show the results when we fix the number of iterations and sum up the values of the performance index for all the problems, obtaining only one value per number of iterations; in this case, the sample size is $13 \times 50 = 650$. With a sample size equal to 650 and a significance level $\alpha = 0.05$, the acceptance region of the

hypothesis test of a proportion, expressed in terms of the number of times that the original version is better than or equal to the non-Pareto one, proves to be $[301, 349]$. If we fix the problem number and summarize the results per number of iterations (see Table 5), the sample size is $3\times50=150$, and the acceptance region is $[63, 87]$.

Table 4: Value of the performance index per number of iterations. We show in **boldface** the value of the performance index of the best algorithm.

	Number of iterations		
Algorithm	100	1000	3000
DE1	**449**	**366**	**359**
DE1-nPb	201	284	291
DE2	227	175	167
DE2-nPb	**423**	**475**	**483**
EDA	163	175	202
EDA-nPb	**487**	**475**	**448**
GC	289	266	253
GC -nPb	**361**	**384**	**397**

Table 5: Value of the performance index per problem. We show in **boldface** the value of the performance index of the best algorithm.

	Problem number												
Algorithm	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
DE1	**102**	**98**	**104**	**94**	**150**	75	**95**	69	80	85	81	70	71
DE1-nPb	48	52	46	56	0	75	55	81	70	65	69	80	79
DE2	8	9	47	9	5	59	4	2	**94**	**138**	71	63	60
DE2-nPb	**142**	**141**	**103**	**141**	**145**	**91**	**146**	**148**	56	12	79	87	**90**
EDA	**150**	42	0	10	0	51	0	1	18	0	78	73	**117**
EDA-nPb	0	**108**	**150**	**140**	**150**	**99**	**150**	**149**	**132**	**150**	72	73	33
GC	56	47	63	46	59	73	21	62	73	78	77	79	74
GC -nPb	**94**	**103**	87	**104**	**91**	77	**129**	**88**	77	72	73	71	76

So, for example, if we pay attention to Table 5 and the algorithms DE1 and DE1-nPb, in problem P_9, out of the 150 populations obtained, version DE1 is

80 times better than or equal to DE1-nPb, and version DE1-nPb is 70 times better than DE1. If the number of times that the original version is equal to or better than the non-Pareto one falls in the range $[63, 87]$, we cannot reject the null hypothesis, which means that both versions are equivalent. If we reject the null hypothesis, if the performance index of the original version is greater than or equal to 88, we will say that the original version is the best one; and if the performance index is smaller than or equal to 62, we will say that the non-Pareto one is the best. In the previous example, since $80 \in [63, 87]$ we then say that both versions are equivalent.

Next, we will comment on the results obtained:

DE1 vs DE1-nPB: For this operator, paying attention to the summary per iteration number (Table 4), version DE1 seems to be clearly better. But the superiority decreases when the number of iterations increases, being less accentuated for 1000 and 3000 iterations. If we analyze the results per problem, (Table 5), for those with two objective functions (P_1 to P_7), version DE1 appears to be the best: it outperforms DE1-nPb in 6 problems. For problems with three objective functions (P_8 to P_{13}), both operators show an equivalent behavior. In summary, we can say that DE1 behaves better than DE1-nPb. The tendency of both versions to present a similar performance level when iterations increase makes sense taking into account the base algorithm, NSGA-II, since when more iterations are run, the number of efficient solutions tends to increase and then both implementations tend to be equivalent because of the definition of the modified version DE1-nPb. It also makes sense for the differences to appear only in problems with two objective functions since in problems with three, the number of efficient solutions increases and then, again, both implementations become equivalent. It seems as if the inclusion of the differential part that guides the search towards better solutions, $\gamma \cdot (\boldsymbol{x}^{(best,t)} - \boldsymbol{x}^{(i,t)})$ in Eq. 1, provides a small advantage that disappears when the number of iterations increases, thus the information provided by

the two differential vectors $F \cdot \sum_{k=1}^{K} (\boldsymbol{x}^{(i_a^k,t)} - \boldsymbol{x}^{(i_b^k,t)})$ seems to be enough.

DE2 vs DE2-nPb: As it can be observed in Table 4, version DE2-nPb appears to be better than DE2. This situation is slightly more pronounced when iterations increase. In Table 5, we can see that in problems with two objective functions version DE2-nPb always presents a better behavior. In problems with three objective functions, none of the versions seem to be clearly better than the other; DE2 is the best in problems 9 and 10 and DE2-nPb is the best for problems 8 and 13. In the remaining cases, both versions are equivalent. Therefore, in spite of the promising ideas on which the operator is based, to maintain differentials adapted for generating vectors which point towards the Pareto-optimal set, $K \cdot (\boldsymbol{x}^{(i_c,t)} - \boldsymbol{x}^{(i,t)})$ in Eq. 3, and for generating a diverse set of solutions, $F \cdot (\boldsymbol{x}^{(i_a,t)} - \boldsymbol{x}^{(i_b,t)})$, we can conclude that DE2-nPb is preferred rather than DE2. We believe that the problem is not in the basic ideas, but in the design of the operator. $(\boldsymbol{x}^{(i_c,t)} - \boldsymbol{x}^{(i,t)})$ could be considered as an improvement direction but only if the solutions are close enough. Otherwise, it is a "common" differential vector (in which case, it does not add any information) and therefore it uses less information in its functioning. A similar situation can be found for the spread device when both solutions from the same layer are not close. In this case, the differential vector does not guarantee that the perturbation tends to generate the spread values well. Furthermore, the differential part in charge of guiding towards the Pareto-optimal set is not applied if the solution to be varied is efficient. So, when the number of iterations as well as the number of efficient solutions increases, this part is not used. On the other hand, the modified version always uses two pairs of differential vectors. In our opinion, the operator could be improved if it took into consideration the distance between solutions.

EDA vs EDA-nPb: In Table 4 we can observe that the number of times in which EDA-nPb is better than EDA it is always greater than 349 although this amount slightly decreases with the number of iterations. If we analyze

the results per problem (see Table 5), EDA-nPb shows a better behavior in the majority of the problems regardless of the number of objective functions. Only in 2 out of 13 is EDA the best, in another 2 both versions are equivalent and in the remaining 9 problems EDA-nPb is the best.

GC vs GC-nPb: With regard to the number of iterations, GC-nPb exhibits the best performance which increases with the number of iterations as it can be observed in Table 4. If we analyze the results per problem (see Table 5), in those with two objectives, version GC-nPb appears to be better than or similar to GC. For problems with three objective functions, it can be observed that both versions are equivalent. Therefore, we can select the GC-nPb version as the best one.

In view of these comments and contrary to what could be expected, the Pareto based versions of the operators are not always clearly superior to the non-Pareto based ones. In particular, DE1 is better than DE1-nPb, but they tend to be equivalent when iterations increase. Version DE2-nPb is better than DE2, especially for problems with two objectives. With regard to EDA, EDA-nPb is clearly better than EDA. And finally for GC, we have noticed a slight superiority of GC-nPb.

5.2. Experiment 2: Comparison among Pareto based operators

For the second experiment, we consider the original proposed versions, DE1, DE2, EDA and GC together with the original NSGA-II and PBV. The values of the performance index appear, summarized per problem, in the top half of Table 6; and per number of iterations, in the top half of Table 7. For example, in Table 6, the column that corresponds to 1000 iterations means that out of the 650 executions (50 initial populations $\times$ 13 problems) 56 times the best value of HD was obtained with algorithm DE1, 10 times with algorithm DE2, 117 times with algorithm EDA, and so on. After obtaining these values, we carried out the multiple comparison test for proportions, previously mentioned in Section 4.4, in order to decide whether the obtained values of the performance index can all

be considered statistically equivalent or not. But since the multiple comparison test for proportions always rejected the null hypothesis, we compared these values in pairs using a χ^2 goodness-of-fit test with the multiple-comparison correction of Bonferroni and a global significance level equal to 0.05. The results of the statistical tests are shown by means of the letters "a", "b" and so on, which appear in some rows of the bottom half of the tables. For instance, in Table 6, considering the column that corresponds to 1000 iterations, the performance index values obtained with the algorithms DE1 and NSGA-II can be considered equivalent and so, the letter "a" appears in the bottom half in these algorithms. Also, the values obtained with the algorithms DE2 and GC can be considered equivalent, so the letter "b" points it out in the bottom half of the table. However, the pairs DE1/NSGA and GC/DE2 cannot be considered equivalent, and that is why we have used different letters.

Table 6: Value of the performance index per number of iterations. The letters "a" and "b" show the values of the performance index that can be considered equivalent after applying the statistical tests.

	Number of iterations		
Algorithm	100	1000	3000
DE1	38	56	84
DE2	36	10	3
EDA	123	117	83
GC	6	13	8
NSGA-II	41	49	56
PBV	406	405	416
Best	PBV	PBV	PBV
	EDA	EDA	DE1^a
	NSGA-II^a	DE1^a	EDA^a
	DE1^a	NSGA-II^a	NSGA-II^a
	DE2^a	GC^b	GC^b
Worst	GC	DE2^b	DE2^b

In view of Table 6, in which we present the summary per iterations, we can observe that, in general, the best behavior corresponds to PBV followed by EDA. Then, we can consider as equivalent to the algorithms DE1 and NSGA-II and both are equivalent to EDA when the number of iterations increases. In

Table 7: Value of the performance index per problem. The letters "a", "b" and so on show the values of the performance index that can be considered equivalent after applying the statistical tests

	Problem number						
Algorithm	P_1	P_2	P_3	P_4	P_5	P_6	P_7
DE1	0	0	0	66	57	23	27
DE2	0	0	3	2	0	17	25
EDA	0	0	0	2	0	54	0
GC	0	0	0	0	0	27	0
NSGA-II	0	0	0	12	0	26	98
PBV	150	150	147	68	93	3	0
Best	PBV	PBV	PBV	PBV^a	PBV	EDA	NSGA
	DE1^a	DE1^a	DE2^a	DE1^a	DE1	GC^a	DE1^a
	DE2^a	DE2^a	DE1^a	NSGA^b	DE2^a	NSGA^a	DE2^a
	EDA^a	EDA^a	EDA^a	DE2^{b,c}	EDA^a	DE1^a	EDA^b
	GC^a	GC^a	GC^a	EDA^{b,c}	GC^a	DE2^a	GC^b
Worst	NSGA^a	NSGA^a	NSGA^a	GC^c	NSGA^a	PBV	PBV^b

Algorithm	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	
DE1	0	4	0	0	0	1	
DE2	0	1	0	0	0	1	
EDA	0	0	6	100	23	138	
GC	0	0	0	0	0	0	
NSGA-II	0	10	0	0	0	0	
PBV	150	135	144	50	127	10	
Best	PBV	PBV	PBV	EDA	PBV	EDA	
	DE1^a	NSGA^a	EDA^a	PBV	EDA	PBV^a	
	DE2^a	DE1^{a,b}	DE1^a	DE1^a	DE1^a	DE1^{a,b}	
	EDA^a	DE2^{a,b}	DE2^a	DE2^a	DE2^a	DE2^{a,b}	
	GC^a	EDA^b	GC^a	GC^a	GC^a	GC^b	
Worst	NSGA^a	GC^b	NSGA^a	NSGA^a	NSGA^a	NSGA^b	

the last positions, we find DE2 and GC.

If we look at Table 7, where the information appears summarized per problem, PBV is the version that appears more times as the best one, in 9 out of the 13 problems. Furthermore, when PBV is not the best option, it is the second best in 2 out of 4 remaining problems and it is the worst option in only 2 problems (P_6 and P_7, both problems with two objectives). On the other hand,

EDA appears as the best one in 3 out of 13 problems and as the second best in 6 problems. Among the remaining operators, we cannot establish a clear order, except for GC which presents the worst results, being the one that occupies the two top positions fewer times.

To conclude this part of the study and confirm or refute the superiority of PBV, we compare, in an isolated form, the two algorithms that can be considered as the best: PBV against EDA-nPb. We must mention that we take EDA-nPb instead of EDA since in the first experiment it presented a better behavior than EDA. The procedure was the same as in the first experiment. In Tables 8 and 9, we present the values of the performance index for HD summarized per number of iterations and per problem, respectively. In Table 8, we can observe that PBV always shows a better performance than EDA-nPb and the difference increases with the number of iterations. With respect to Table 9, we can observe that, globally, PBV is better than or equal to EDA-nPb in 10 out 13 problems and worst in only 3, one with 2 objective functions and two with 3. So, we can say that, although EDA-nPb has improved the results of EDA, PBV still presents the best performance.

Table 8: Value of the performance index per number of iterations. We show in **boldface** the value of the performance index of the best algorithm.

	Number of iterations		
Algorithm	100	1000	3000
PBV	**461**	**486**	**540**
EDA-nPb	164	286	110

Table 9: Value of the performance index per problem. We show in **boldface** the value of the performance index of the best algorithm.

	Problem number												
Algorithm	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
PBV	**150**	**150**	**116**	**113**	**150**	49	**150**	**149**	**150**	**102**	48	**127**	33
EDA-nPb	0	0	34	37	0	**101**	0	1	0	48	**102**	23	**117**

Figure 2: Evolution of the mean value of HD for 10 runs of the algorithms for problems 12 (left) and 5 (right).

In addition to the previous comments, we have also studied the convergence speed. We are interested in knowing if any algorithm is able to reach good solutions in the early stages of the run, even if it does not maintain its good behavior later on. We are also interested in the opposite behavior, i.e., an algorithm which starts with bad solutions but outperforms the others, as we increase the number of iterations.

We have solved each problem 10 times with each algorithm, using 4500 iterations. We have calculated the mean value of HD for each intermediate number of iterations 100, 400, 700,..., 4500 and so on, in order to guess how HD evolves. As a first conclusion, we can state that the main decrease of the mean value of HD takes place approximately in the first 500 iterations for all the problems. Furthermore, the operator or operators which present a good HD with a low number of iterations tend to maintain this good behavior during the rest of the run. This happens in problems 1 to 3, 8 to 10, 12 and 13 (see left handside of Fig. 2). In the other 5 problems, one algorithm starts as the best one, but as the number of iterations increases, another algorithm surpasses it, either at the early stages of the execution or later on (see right handside of Fig. 2). Therefore, in general, we cannot say that any of the algorithms presents a higher convergence speed than the others.

5.3. Experiment 3: Comparison between EDA-nPb, EDA-DE and PBV

As we stated before, the EDA version that we have used is not the original one proposed by its authors in [47] but an EDA operator hybridized with a DE operator. In the previous experiments, in order to perform a fair study, we considered only the EDA part, since the other operators did not have these characteristics. Also, in order to remain faithful to the original operator (called EDA-DE), we compare it with respect to EDA and, afterwards, we also compare it with PBV. The values of the parameters of EDA-DE are fixed to $p_r^{max} = 0.9$, $p_r^{min} = 0.2$, $\beta = 0.95$, $F = 0.3$ and $CR = 0.3$ as its authors proposed in [47].

The procedure was the same as in the first experiment. It can be observed in the top half of Tables 10 and 11 that EDA-DE significantly outperforms EDA. It is always better in the summary per iterations and it shows a better performance index in 11 out of 13 of the problems; also, they are equivalent in the remaining two problems. So, the hybridization in this way produces a significant improvement. In view of these results, we wondered whether the improvements would be enough to surpass the performance of the PBV operator or not. So, we compared PBV against EDA-DE. In the bottom half of Tables 10 and 11, we can see the results of this comparison. In the summary per iterations, PBV always shows a higher performance index, although the value slightly decreases with the number of iterations. In the summary per problem, we can see how PBV presents better or equal results in 9 problems, and worse results in only 4 problems. So, although EDA-DE has increased the performance of EDA and EDA-nPb, this is not enough to make it a better option than PBV.

Table 10: Value of the performance index per number of iterations. We show in **boldface** the value of the performance index of the best algorithm.

	Number of iterations		
Algorithm	100	1000	3000
EDA-DE	**521**	**568**	**574**
EDA	129	82	76
PBV	**473**	**397**	**402**
EDA-DE	177	253	248

Table 11: Value of the performance index per problem. We show in **boldface** the value of the performance index of the best algorithm.

	Problem number												
Algorithm	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
EDA-DE	**150**	**148**	**150**	**140**	**150**	**90**	**150**	**150**	**144**	**147**	**88**	76	80
EDA	0	2	0	10	0	60	0	0	6	3	62	74	70
PBV	**112**	**145**	**150**	**104**	77	43	50	**150**	**148**	**124**	17	**138**	14
EDA-DE	38	5	0	46	73	**107**	**100**	0	2	26	**133**	12	**136**

6. Conclusions and further research

In order to show whether or not the use of Pareto based variation operators can improve the performance of Evolutionary Algorithms, in this work we have presented a comparison amongst several implementations of a well-known algorithm, NSGA-II, in which the original variation operators have been replaced by variation operators that take into consideration the multi-objective nature of the problems to be solved. The considered variation operators come from the Differential Evolution, Estimation of Distribution Algorithms, Evolution Strategies and Evolutionary Programming fields, and also include two operators proposed in previous works by the authors. We should also comment that the operators we found in the literature, among which are those we have selected for the paper, generally make minimal use of the information on the quality of the solutions, a fact that presumably should be taken into account to improve the behaviour of those operators.

The results of the experiments show that adopting variation operators which use explicitly the multi-objective character of the problems does not guarantee success, since, in general, the Pareto versions of the operators did not provide us with better results. The only exceptions are the variation operators proposed by the authors, which unlike the others,which are adaptations of similar operators, have been built entirely and from the

beginning around Pareto elements.

So, in our personal opinion, in order to develop more efficient variation operators to be included in Multi-Objective Evolutionary Algorithms, additional research should be done in this area. In the design of the operators, some important elements should be taken into account. The operators should appropriately balance the capacities of exploration and exploitation in function of, for instance, the quality of the solution (based on Pareto ranking or on any other ranking or quality measure of the solution), its age (number of iterations that this solution has been in the successive populations), and so on. The exploratory capacity should be initially higher and also nearly independent of the quality of

the solution, and it should give way to the capacity of exploitation as iterations proceed. Also, when the number of iterations is high, we should stress exploitation when we are working with good solutions and use the not so good solutions for maintaining certain exploratory capacity. It is also important to take into account the proximity of the solutions when, for example, performing crossover. The ideas used in the operator MODE [27] are very interesting and promising. However, since this operator does not take into account the distance between solutions, in our opinion, the operator loses effectiveness. In view of the third experiment, another possible area of research would be the hybridization of the operators, combining operators which allocate both capacities, as for example EDA-DE. This operator, as we have previously mentioned, combines an EDA and a DE. The EDA part presents a higher exploratory capacity and tends to be applied with a higher probability in the early stages of the algorithm. Later on, DE is applied with a higher probability, which means that in the final stages, when the solutions are very close, the capacity of exploitation increases. We are convinced that working on considering some of the above ideas could lead to the development of variation operators that are more efficient for multi-objective optimization than the current ones.

Acknowledgments

This research has been supported by the Universidad de Zaragoza under grant UZ2012-CIE-09 and by the Research Group of Gobierno de Aragón E58.

References

[1] Abbass, H., 2002. The self-adaptive Pareto differential evolution algorithm. Computational Intelligence 1.

[2] Abbass, H., Sarker, R., Newton, C., 2001. PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the Congress on Evolutionary Computation. IEEE, pp. 971–978.

[3] Alberto, I., Mateo, P., Apr. 2011. A crossover operator that uses Pareto optimality in its definition. TOP 19 (1), 67–92.

[4] Baluja, S., Caruana, R., 1995. Removing the genetics from the standard genetic algorithm. In: Prieditis, A., Russel, S. (Eds.), 12th Annual Conference on Machine Learning. Morgan Kaufmann, pp. 38–46.

[5] Batista, L., Guimarães, F., Ramírez, J., 2009. A differential mutation operator for the archive population of multi-objective evolutionary algorithms. In: Proceedings of the Eleventh conference on Congress on Evolutionary Computation. CEC'09. IEEE Press, Piscataway, NJ, USA, pp. 1108–1115.

[6] Bosman, P., Thierens, D., 2006. Multi-objective optimization with the naive MIDEA. Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms, 123–157.

[7] Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Information Sciences 237, 82–117.

[8] Casella, G., Berger, R.L., 2002. Statistical Inference. Duxbury Advanced Series, Pacific Grove, California.

[9] Chang, C., Xu, D., Quek, H., 1999. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. IEE Proceedings - Electric Power Applications 146 (5), 577–583.

[10] Chellapilla, K., 1998. Combining Mutation operators in Evolutionary Programming. IEEE Transactions on Evolutionary Computation 2 (3), 91–96.

[11] Chen, C., Chen, Y., Zhang, Q., May 2009. Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization. In: IEEE Congress on Evolutionary Computation. IEEE, pp. 209–216.

[12] Chen, M., Lu, Y., Aug. 2008. A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization. European Journal of Operational Research 188 (3), 637–651.

[13] Coello, C.A., 2013. EMOO Repository (Online). Available in URL `http://delta.cs.cinvestav. mx/~ccoello/EMOO/`. Accessed September 2013.

[14] Coello, C., Lamont, G., Van Veldhuizen, D., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition. Springer, Berlin.

[15] Das, S., Suganthan, P.N., Feb. 2011. Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation, 15 (1), 27–54.

[16] Deb, K., 2001. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester.

[17] Deb, K., Agrawal, R., 1995. Simulated binary crossover for continuous search space. Complex systems 9 (2), 115–148.

[18] Deb, K., Goyal, M., 1996. A combined genetic adaptive search (GeneAS) for Engineering design. Computer Science and Informatics 26, 30–45.

[19] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

[20] Dong, H., He, J., Huang, H., Hou, W., Jan. 2007. Evolutionary programming using a mixed mutation strategy. Information Sciences 177 (1), 312–327.

[21] Eiben, A., Smith, J., 2007. Introduction to Evolutionary Computing. Springer, Berlin.

[22] García, S., Molina, D., Lozano, M., Herrera, F., 2009. A study on the use of non-parametric test for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. Journal of Heuristics 15, 617–644.

[23] Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, Massachusetts.

[24] Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan, USA.

[25] Huang, V., Qin, A., Deb, K., Zitzler, E., Suganthan, P., Liang, J., Preuss, M., Huband, S., 2007. Problem definitions for performance assessment of multi-objective optimization algorithms. Tech. rep., Nanyang Technological University, Singapore.

[26] Iorio, A., Li, X., 2004. Solving rotated multi-objective optimization problems using differential evolution. In: AI 2004: Advances in Artificial Intelligence. Springer, pp. 861–872.

[27] Iorio, A., Li, X., 2006. Incorporating directional information within a differential evolution algorithm for multi-objective optimization. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, pp. 691–698.

[28] Karshenas, H., Santana, R., Bielza, C., Larrañaga, P., 2011. Multi-objective optimizaction with joint probabilistic modeling of objectives and variables. Lecture Notes in Computer Science 6576, 298-312.

[29] Larrañaga, P., Lozano, J.A. (Eds.) 2002. Estimation of distribution algorithms: A new tool for evolutionary computation. Kluwer Academic Publishers, Boston.

[30] Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (Eds.) 2006. Towards a new evolutionary computation. Advances in estimation of distribution algorithms. Studies in Fuzziness and Soft Computing 192. Springer, Heidelberg.

[31] Martí, L., García, J., Berlanga, A., Coello, C., Molina, J., Mar. 2011. MB-GNG: Addressing drawbacks in multi-objective optimization estimation of distribution algorithms. Operations Research Letters 39 (2), 150–154.

[32] Mateo, P., Alberto, I., Jan. 2012. A mutation operator based on a Pareto ranking for multi-objective evolutionary algorithms. Journal of Heuristics 18 (1), 53–89.

[33] Michalewicz, T., 1996. Genetic Algorithms + Data Structures = Evolution Programs, third edn. Edition. Springer, Berlin.

[34] Mühlenbein, H., Paass, G., 1996. From recombination of genes to the estimation of distributions I. Binary parameters. In: In LNCS 1411:Proceedings 4th Parallel Problem Solving for Nature. Springer, pp. 178–187.

[35] Okabe, T., Jin, Y., Sendoff, B., Olhofer, M., 2004. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In: Congress on Evolutionary Computation, 2004. CEC2004. Vol. 2. IEEE, Portlan, OR, USA, pp. 1594–1601.

[36] Rosenberg, R., 1967. Simulation of genetic populations with biochemical properties. Ph.D. thesis, University of Michigan, Ann Arbor, Michigan, USA.

[37] Schaffer, J., 1984. Multiple Objective Optimization eith Vector Evaluated Genetic Algorithms. Ph.D. thesis, Vanderbilt University, Nasvhille, Tennessee.

[38] Schwefel, H., 1995. Evolution and Optimal Seeking. Wiley, New York.

[39] Sebag, M., Ducoulombier, A., 1998. Extending population-based incremental learning to continuous search spaces. In: Eiben, A., Bäck, T., Schoenauer, M., Schwefel H.P. (Eds.), PPSN V Proceedings of the 5th International Conference on Parallel Problem Solving from Nature. Springer, London, pp. 418–427.

[40] Srinivas, N., Deb, K., 1994. Multi-objective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2 (3), 221–248.

[41] Storn, R., Price, K., 1995. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., TR-95-12 International Computer Science Institute, Berkeley, California.

[42] Suganthan, P., 2007. CEC (2007) Special Session and Competition on Performance Assessment of Multi-Objective Optimization Algorithms. URL http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-07/CEC07.htm

[43] Talbi, E.G., Basseur, M., Nebro, A.J., Alba, E., 2012. Multi-objective optimization using metaheuristics: non-standard algorithms. International Transactions in Operational Research 19 (1/2), 283–305.

[44] Thierens, D., Bosman, P., 2001. Multi-objective mixture-based iterated density estimation evolutionary algorithms. In: Spector, L., Goodman, E., Wu, A. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference GECCO'2001. pp. 663–670.

[45] Thierens, D., Bosman, P., 2001. Multi-objective optimization with iterated density estimation evolutionary algorithms using mixture models. In: Third International Symposium on Adaptative Systems, Evolutionary Computation and Probabilistic Graphical Models. pp. 129–135.

[46] Tsutsui, S., Pelikan, M., Goldberg, D., 2001. Evolutionary algorithm using marginal histogram models in continuous domain. In: Proceedings of the Genetic and Evolutionary Computation Conference Workshop. Citeseer, San Francisco, CA, pp. 230–233.

[47] Wang, X., Hao, M., Cheng, Y., Lei, R., 2009. PDE-PEDA: A New Pareto-Based Multi-objective Optimization Algorithm. Journal of Universal Computer Science 15 (4), 722–741.

[48] Xue, F., Sanderson, A., Graves, R., 2003. Pareto-based multi-objective differential evolution. In: Proceedings of the 2003 Congress on Evolutionary Computation, CEC'03. Vol. 2. pp. 862–869.

[49] Xue, F., Sanderson, A., Graves, R., 2005. Multi-objective differential evolution-algorithm, convergence analysis, and applications. In: Proceedings of the IEEE Congress on Evolutionary Computation. Vol. 1. IEEE, pp. 743–750.

[50] Xue, F., Sanderson, A., Graves, R., 2009. Multiobjective Evolutionary Decision Support for Design-Supplier-Manufacturing Planning. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 39 (2), 309–320.

[51] Yang, S., Shao, D., Luo, Y., Nov. 2005. A novel evolution strategy for multiobjective optimization problem. Applied Mathematics and Computation 170 (2), 850–873.

[52] Yao, X., Liu, Y., Lin, G., Jul. 1999. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3 (2), 82–102.

[53] Zhang, Q., Zhou, A., Jin, Y., 2008. RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. IEEE Transactions on Evolutionary Computation 12 (1), 41–63.

[54] Zhou, A., Zhang, Q., Jin, Y., Tsang, E., Okabe, T., 2005. A model-based evolutionary algorithm for bi-objective optimization. In: The 2005 IEEE Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 2568–2575.

[55] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P.N., Zhang, Q., 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation 1 (1), 32–49.

[56] Zitzler, E., 1998. Multiobjective Optimization Using Evolutionary Algorithms - A comparative case study. Lecture Notes in Computer Science 1498, 292–301.

[57] Zitzler, E., Thiele, L., 1998. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Tech. Rep. 43, Computer Engineering and Communication Networks lab (TIK), Zurich.

[58] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V., 2003. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.