Multi-Objective Compact Differential Evolution

Jesus Moises Osorio Velazquez
CINVESTAV-IPN, Av. IPN 2508,
San Pedro Zacatenco, Gustavo A. Madero,
México D.F. 07360, MEXICO.
e-mail: moises.osorio@wkoder.com

Carlos A. Coello Coello
CINVESTAV-IPN, Av. IPN 2508,
San Pedro Zacatenco, Gustavo A. Madero,
México D.F. 07360, MEXICO.
e-mail: ccoello@cs.cinvestav.mx

Alfredo Arias-Montaño
IPN-ESIME, Av. Ticoman 600,
San José Ticoman, Gustavo A. Madero,
México D.F. 07340, MEXICO.
e-mail: aarias@ipn.mx

***Abstract*—A wide range of problems in engineering require the simultaneous optimization of several objectives. Given the nature of such problems, it is often the case that the optimization process needs to take place from a device with very limited resources. Compact algorithms are a suitable alternative for being implemented in devices with limited computing resources, but so far, they have been used only to solve single-objective optimization problems. Here, we present a multi-objective compact algorithm based on differential evolution. The proposed algorithm obtains competitive results (and even better in some cases) than state-of-the-art multi-objective evolutionary algorithms while using less memory resources because of its statistical representation of the population.**

I. Introduction

In many areas of engineering, the optimization of two or more objectives is frequently required. Such problems, called multi-objective problems (MOPs) have not one, but a set of optimal solutions. Several mathematical programming methods are currently available for solving MOPs [1]. Nevertheless, such methods have some limitations and usually require additional information about the problem to be solved. Such limitations have motivated the use of metaheuristics, from which multi-objective evolutionary algorithms (MOEAs) have become a very popular choice.

In the real world, there are many applications that involve solving optimization problems without using too many computer resources, maybe because of cost or space reasons. These situations usually arise in robotics and control problems. For these types of problems, state-of-the-art evolutionary algorithms (EAs) can normally not be used because of the high amount of computer resources that they usually require. Compact evolutionary algorithms (CEAs) have been developed as a viable choice for these types of problems. A compact EA belongs to the class of estimation of distribution algorithms (EDAs) [2] but adopts the principles used in traditional EAs. EDAs only save a statistical representation of the population. In this way, CEAs require less memory than their traditional counterparts. Although there exist CEAs that are suitable for being implemented in devices with limited computing resources, few attempts have been made to extend their application to the multi-objective optimization case (see [3], [4]). In these cases, binary code representation and PBIL strategy [5] are used. To the best of the authors' knowledge, no CEAs currently exist for solving multi-objective optimization problems using real-coded representation. In this paper, we propose the first multi-objective compact differential evolution algorithm (mocDE). This algorithm is designed to use less memory resources by adopting a statistical representation of the main population instead of the whole population, adopted by traditional MOEAs. The proposed approach was also designed to produce competitive results with respect to state-of-the-art MOEAs.

The remainder of this paper is organized as follows: In Section II we present a brief review of some previous work related to the design of CEAs. Then, in Section III, we present our proposed mocDE. Section IV describes the methodology used for benchmarking our proposed mocDE, using standard test MOPs and performance measures usually adopted in the multi-objective optimization literature. Section V is devoted to present a comparison of results of our proposed approach with respect to several state-of-the-art MOEAs. Finally, in Section VI we present our conclusions and some possible paths for future work.

II. Compact Evolutionary Algorithms

In this section, we describe the main characteristics of several CEAs found in the literature.

A. Compact Genetic Algorithm

The *Compact Genetic Algorithm* (cGA) was the first CEA ever proposed [6]. cGA simulates a traditional genetic algorithm (GA) with binary encoding and consists of a probability vector v of length n which is initialized with n values equal to 0.5, determining the probability that a gene has of being either 0 or 1. At each iteration, it takes two samples from v, which serve as individuals, and it calculates their fitness to compete against each other. The winner affects v according to the virtual population size p. If the winner has a 1 in gene i and the loser has a 0, the probability of v_i increases $1/p$. Otherwise, v_i decreases $1/p$. In case both individuals have the same value, v is not modified. The resulting vector v represents the final solution.

With a population of p individuals, cGA requires only $n \log_2(p+1)$ memory bits. The selection process used by the cGA focuses on favoring the best genes, instead of the best individuals as in a traditional GA. Therefore, generating individuals from this vector can be seen as a shortcut to the final goal of crossover. Two different cGA variants were proposed [7]: *Persistent Elitist Compact Genetic Algorithm* (pe-cGA) and *Non-Persistent Elitist Compact Genetic Algorithm* (ne-cGA). Both variants, showed a better performance than cGA after incorporating elitism to the optimization process. Both the pe-cGA and the ne-cGA make use of an *elite* individual, in addition to the probability vector v. At each iteration, these

algorithms only generate one more individual which competes against the *elite* one. The main difference between these two variants resides in the moment a new individual becomes the elite individual. The pe-cGA only replaces the elite individual when the new individual is better. In the ne-cGA, the elite individual is replaced when the new individual is better or when it has not been replaced after η iterations. Note that the pe-cGA is equivalent to the ne-cGA when $\eta = \infty$.

B. Real-Valued Compact Genetic Algorithm

The *Real-Valued Compact Genetic Algorithm* (rcGA) [8] is another variant of the cGA that takes the compact feature from the binary domain to the real domain. In rcGA, the probability vector v is not a vector of bits but a matrix of dimension $n \times 2$:

$$V = [\vec{\mu}, \vec{\sigma}] \tag{1}$$

where $\vec{\mu}$ and $\vec{\sigma}$ are vectors of length n. Each position i of $\vec{\mu}$ and $\vec{\sigma}$ represent the mean and standard deviation, respectively, of the decision variable x_i in the problem to optimize. Such vectors represent a Gaussian probability distribution function (PDF) truncated to the interval [-1, 1] with a height that is normalized to an area of 1. At the beginning, the values $\vec{\mu}$ are initialized with 0 and $\vec{\sigma}$ with α, where α is a constant ($\alpha = 10$) so that the PDF works as a uniform distribution that converges to a Gaussian distribution. As in the pe-cGA and the ne-cGA, an elite individual is generated to guide the optimization process, and in each iteration a new individual is generated, by means of v, to compete against the elite. Because the rcGA deals with real values instead of binary values, the update rules of v are different:

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i) \tag{2}$$

$$(\sigma_i^{t+1})^2 = (\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2) \tag{3}$$

Both persistent and non-persistent variants of the rcGA work as the pe-cGA and the ne-cGA, and even show the same features regarding solution quality and convergence speed. The authors of the rcGA also show that using one variant or another is problem dependent.

Sampling mechanism: To generate the variable i of a new individual x by means of the Gaussian PDF, the rcGA uses the next formula:

$$PDF(x_i) = \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}\sqrt{\frac{2}{\pi}}}{\sigma_i\left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} \tag{4}$$

where erf is the error function [9], and μ_i and σ_i are the mean and standard deviation of variable i, respectively. From the PDF, its cumulative distribution function (CDF) is calculated by Chebyshev polynomials, using the method described in [10]. Given the co-domain of the CDF is $[0, 1]$, to obtain a sample s_i, a random number $r_i \in [0, 1)$ taken from an uniform distribution is needed. Then, this random number is evaluated by the inverse function of the CDF, i.e., $s_i = CDF^{-1}(r_i)$. The resulting sample is in the interval $[-1, 1]$ and to return it back to the original interval $[a, b]$ of the variable, the formula $x_i = a + \frac{b-a}{2}(s_i + 1)$ is applied.

C. Compact Differential Evolution

Compact Differential Evolution (cDE) [11] was introduced after modifying the rcGA to use the principles of differential evolution (DE) instead of those of GAs. Since cDE uses the same sampling mechanism of the rcGA, both algorithms are very similar. Considering the most common implementation of DE, DE/rand/1/bin, at each iteration three individuals x_r, x_s and x_t are sampled by means of v, and are used to create a new individual x'_{off}. This new individual is recombined with the elite individual to finally generate individual x_{off}, which will compete against the elite individual. cDE and the rcGA only differ in their crossover and mutation mechanisms. cDE also has two variants: with persistent elitism (pe-cDE) and with non-persistent elitism (ne-cDE). Both variants work as in the rcGA.

cDE has shown a very good performance in a wide range of problems, and has been compared even against traditional differential evolution. Some reasons for its success, described in more detail in [11], are:

- The cDE generates new individuals strictly following the DE principles, and not only by means of v, increasing its exploratory power.
- Unlike the cGA and the rcGA, the cDE allows to directly implement the survival scheme of DE, avoiding degradation.
- Given that cDE is based on a probabilistic structure, it is unnecessary to rely on the control parameters randomness because such randomness is introduced directly by the cDE in the solutions generated by means of v.
- Although it does not aim to outperform DE, the cDE is an excellent light version of DE because it is capable of achieving a similar performance while reducing the use of $2 \cdot p$ individuals to only 4.

III. Our Proposed approach

In this section, we describe the *Multi-Objective Compact Differential Evolution (mocDE)* approach, which is proposed in two versions: with persistent elitism and with non-persistent elitism. Besides solving unconstrained multi-objective optimization problems, this approach:

- Handles the convergence speed because of its two variants: with persistent elitism and with non-persistent elitism.
- Saves memory resources by using a statistical representation of the main population.
- Produces a good distribution of solutions by using an archive based on Chebyshev aggregation functions.
- Has a great exploratory power due to the use of differential evolution principles.

A. Population representation

As cDE, mocDE operates on a statistical representation of the main population and not on a set of individuals. This allows the use of only five vectors of length n (number of decision variables) instead of p (population size) vectors, achieving a linear order memory savings. The statistical representation of the population, v, is defined by vectors μ and σ (both of length n) that represent the mean and the standard deviation of each decision variable, respectively. The values of μ_i and σ_i define the Gaussian PDF of variable x_i, truncated in the interval $[-1, 1]$ with normalized height, as follows:

$$PDF(x_i) = \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \sqrt{\frac{2}{\pi}}}{\sigma_i \left(erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} \quad (5)$$

From the PDF, mocDE obtains the inverted CDF that allows to take a sample s_i by means of a uniform random number. The general form of CDF^{-1} is:

$$CDF^{-1}(\mu_i, \sigma_i) = \mu_i - \sqrt{2}\sigma_i \times erf^{-1}\left((x - C)\left(erf\left(\frac{\mu_i - 1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i + 1}{\sqrt{2}\sigma_i}\right)\right)\right) \quad (6)$$

where erf is the error function, and C is the integration constant obtained when calculating the CDF:

$$C = -CDF(-1) = -\frac{erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)}{erf\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right) - erf\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)} \quad (7)$$

Figure 1 shows the normalization of two truncated Gaussian curves to obtain their PDF, whilst Figure 2 shows a PDF and its CDF.

During the optimization process, mocDE maintains an elite solution and, in each iteration, it generates a new solution to be compared to the elite solution. Then, the winning solution is added to the population, i.e., it affects the statistical representation of the population.

B. Elitism

Ahn and Ramakrishna [7] show the benefits of elitism in the optimization process when using a compact evolutionary algorithm. Based on that work, mocDE implements the same elitism classes: persistent and non-persistent. This originates two variants: *Persistent Elitist Multi-Objective Compact Differential Evolution* (pe-mocDE) and *Non-Persistent Elitist Multi-Objective Compact Differential Evolution* (ne-mocDE).

Both variants use an extra individual called elite during the optimization process to compare it against the new generated solution and determine if such solution improves the population or not. If the elite solution is better than the new one, it affects the statistical representation of the population, else the new solution affects the population and replaces the elite solution. Such substitution is only done when:

- The new solution dominates the elite solution.
- Both solutions are non-dominated but the new solution can be added to the archive (see subsection III-C).
- η iterations have passed since the last time the elite solution was replaced (only in ne-mocDE).

The main difference between both variants is only the last case of substitution. This adds an extra parameter to ne-mocDE called η, which should be no greater than the population size, i.e., $\eta < p$ according to [7].

C. Archive of non-dominated solutions

To keep a good distribution in the solutions, mocDE implements an external archive that works as the one of MOEA/D. Such archive uses Chebyshev aggregation functions, where weight vectors $\lambda_i, i = 1, 2, ..., p$ are defined by the user and represent the desired distribution. This archive helps to determine whether a solution improves the distribution and it must be declared as the winner when affecting the population. The Chebyshev scaling method has the powerful feature of being able to generate all the Pareto optimal solutions [1]. The Chebyshev method is applied as follows:

$$g_i(x) = \max_{1 \le j \le k} \lambda_{i,j}(f_j(x) - z_j) \quad (8)$$

where z is a reference vector and λ is a set of vectors that define the desired distribution. The reference vector z is the ideal vector found at the current time, i.e., $z_i = \min f_i(x)$ for all x generated during the optimization process. The maximum archive size should be defined by the user and represents the number of solutions to be returned as a result.

Algorithm 1 shows how a new solution x is added to archive A using the reference vector z, returning *true* if it was successfully added or *false* otherwise.

Algorithm 1 *archive* function in mocDE

Require: $n > 0 \wedge k > 1 \wedge p > 0 \wedge x \in \mathbb{R}^n \wedge z \in \mathbb{R}^k \wedge f : \mathbb{R}^n \to \mathbb{R}^k \wedge A \in \mathbb{R}^{p \times n} \wedge \lambda \in \mathbb{R}^{p \times k}$;
Ensure: If solution x was added to archive A
  for $i = 1 \to k$ **do** {Update ideal vector}
    if $f_i(x) < z_i$ **then**
      $z_i \leftarrow f_i(x)$
    end if
  end for
  added $\leftarrow$ **FALSE**
  for $i = 1 \to p$ **do** {Update archive}
    $f_1 \leftarrow \max_{1 \le j \le k} \lambda_{i,j} |f_j(x) - z_j|$
    $f_2 \leftarrow \max_{1 \le j \le k} \lambda_{i,j} |f_j(A_i) - z_j|$
    if $f_1 < f_2$ **then** {x is better than A_i}
      added $\leftarrow$ **TRUE**
      $A_i \leftarrow x$
    end if
  end for
  return added

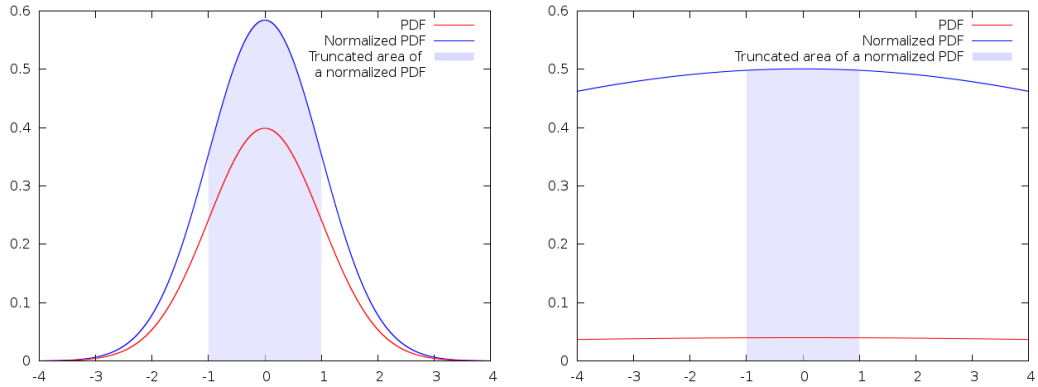
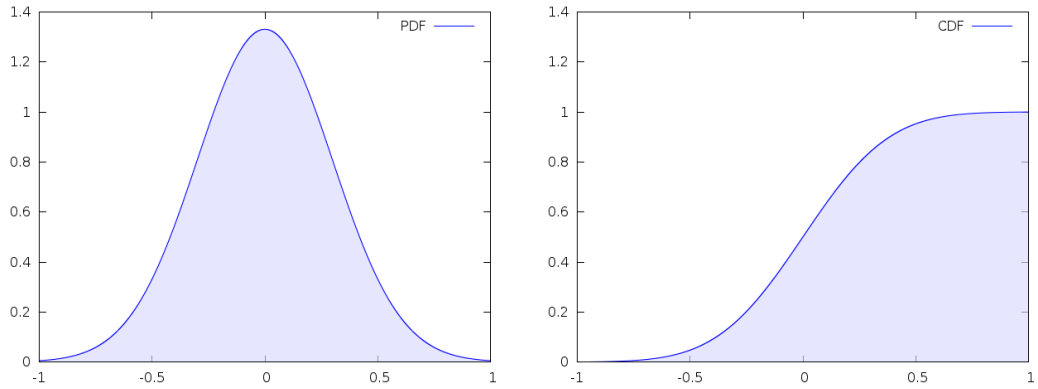


Fig. 1. Normalization of truncated gaussian curves in interval $[-1, 1]$, with $\mu = 0$ and (a) $\sigma = 1$ and (b) $\sigma = 10$.



(a) (b)

Fig. 2. Relation of a (a) PDF (with $\mu = 0$ and $\sigma = 0.3$) and its (b) CDF.

D. MocDE Algorithm

Our proposed approach works as follows. At the beginning, the value of μ is set to 0 and σ is set to α, where α is a constant ($\alpha = 10$) that allows the PDF to work as a uniform distribution that converges to a Gaussian distribution. It also creates an elite individual e that guides the optimization process, which is added as the first solution to the archive and initializes the ideal vector z, which works as a reference vector in the Chebyshev method. At each iteration, mocDE uses the standard implementation of differential evolution $DE/rand - to - best/1/bin$, but it can be replaced by any other DE variant. It generates three individuals x_r, x_s and x_t by means of v, to produce a new individual, x'_{off}. The new individual is recombined with the elite individual to generate individual, x_{off}, which will compete against the elite itself. To decide if x_{off} is better than e, it is checked if $f(x_{off}) \prec f(e)$. In case both solutions are non-dominated, x_{off} is the winner if it can be added to the archive, verifying that there exists some $i \in \{1, .., p\}$ such that $g_i(A_i) > g_i(x_{off})$ (see equation (8)). The proposed approach is shown in Algorithm 2, where α is usually equal to 10.

IV. Performance Assessment

To evaluate the performance of our proposed mocDE, the Zitzler-Deb-Thiele (ZDT) [12] and the Deb-Thiele-Laumanns-Zitzler (DTLZ) [13] test suites were used as benchmarks. To test the quality of the final solutions for each algorithm the following quality indicators have been used: Hypervolume (I_H) [14], Inverted generational distance (I_{IGD}) [15], Binary multiplicative ϵ ($I_{\epsilon*}$) [16], Binary additive ϵ ($I_{\epsilon+}$) [16], and Two set Coverage (I_C) [14]. The first two are unary and the rest are binary indicators.

Algorithm	Parameter	Value
mocDE	Variant	pe-mocDE
	Population size p	100
	Differential variation F	1.0
	Crossover probability C	0.1
NSGA-II	Population size p	100
	Mutation rate m	0.01
	Crossover probability C	0.9
MOEA/D	Population size p	100
	Niche size	100 (2D), 150 (3D)
	Update limit	10 (2D), 15 (3D)
	Differential variation F	1.0
	Crossover probability C	0.5
	Mutation rate m	1.0 / n
	Selection probability	0.9
PAES	Population size p	100
	Mutation rate m	1.0 / n
	Bisection	5
	Distribution index	20

TABLE I. Parameters for mocDE and state-of-the-art MOEAs.

The different MOEAs and their corresponding parameters adopted in our comparative study are shown in Table I.

All results were obtained by running the algorithms for

Algorithm 2 Multi-Objective Compact Differential Evolution (mocDE)

Require: $n > 0 \wedge k > 1 \wedge p > 0 \wedge C_r > 0 \wedge F > 0 \wedge \eta > 0 \wedge f : \mathbb{R}^n \rightarrow \mathbb{R}^k \wedge \lambda \in \mathbb{R}^{p \times k}$
Ensure: Set of solution vectors A
  $t \leftarrow 0$ {Current iteration}
  $\theta \leftarrow 0$ {Survived iterations by the elite solution}
  for $i = 1 \rightarrow n$ **do** {Initialize $v = \{\mu, \sigma\}$}
    $\mu_i \leftarrow 0$ {Mean}
    $\sigma_i \leftarrow \alpha$ {Standard deviation}
  end for
  Generate elite individual e, by means of v
  Initialize archive of non-dominated solutions $A \leftarrow \{e\}^p, z \leftarrow f(e)$
  while stopping criteria is not met **do**
    Generate three individuals x_r, x_s and x_t, by means of V
    $x'_{off} \leftarrow x_t + F \cdot (x_r - x_s) + F \cdot (e - x_t)$ {Apply mutation}
    for $i = 1 \rightarrow n$ **do** {Do crossover}
      $r \leftarrow rand(0, 1)$ {Generate an uniform random number}
      if $r < C_r$ **then**
        $x_{off_i} \leftarrow e_i$
      else
        $x_{off_i} \leftarrow x'_{off_i}$
      endif
    end for
    if $f(x_{off}) \prec f(e) \vee \theta \geq \eta$ **then** {Comparison of $\theta \geq \eta$ is skipped in the persistent variant}
      $w \leftarrow x_{off}$ {Winner}
      $l \leftarrow e$ {Loser}
      $e \leftarrow x_{off}$
      $\theta \leftarrow 0$
      archive(A, z, x_{off}) {Add to the archive}
    else
      if $f(e) \not\prec f(x_{off}) \wedge archive(A, z, x_{off})$ **then** {If non-dominated and can be added to the archive}
        $w \leftarrow x_{off}$ {Winner}
        $l \leftarrow e$ {Loser}
        $e \leftarrow x_{off}$
        $\theta \leftarrow 0$
      else
        $w \leftarrow e$ {Winner}
        $l \leftarrow x_{off}$ {Loser}
        $\theta \leftarrow \theta + 1$
      endif
    endif
    for $i = 1 \rightarrow n$ **do** {Update population representation}
      $\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i)$
      $\sigma_i^{t+1} = \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2)}$
    end for
    $t \leftarrow t + 1$
  endwhile
  return A

the test problems after 30 independent runs, performing a maximum of 20,000 evaluations.

V. Results

Because of the results obtained when comparing the different variants of mocDE, pe-mocDE was chosen as the best mocDE variant (this was the outcome of a comparison against ne-moCDE not included here due to space limitations) and will represent mocDE in this section. Next, we compare mocDE (pe-mocDE) to PAES, MOEA/D and NSGA-II.

In problems ZDT1 and ZDT2, mocDE and NSGA-II show the best results according to unary quality indicators but binary indicators determine that mocDE is better than NSGA-II. Figure V shows the Pareto front approximations of all algorithms for problem ZDT1.

In ZDT3, NSGA-II presents the best results followed by mocDE, according to the unary indicators and additive ϵ (I_ϵ^+). Nevertheless, the two set coverage indicator (I_C) and multiplicative ϵ (I_ϵ^*) show mocDE as the winner. In ZDT4, the best results are from MOEA/D followed by PAES and mocDE, which showed competitive and more consistent results than PAES. In ZDT6, mocDE shows the best results but with some instability. For problems DTLZ1 and DTLZ3, both mocDE and PAES obtained the best results with PAES having a better graphical outcome. NSGA-II showed highly competitive results according to the unary indicators, but mocDE presented better results regarding the binary indicators. In problems DTLZ4 and DTLZ7, NSGA-II was better according to the unary indicators but the binary indicators showed that mocDE had a better performance. In DTLZ5, NSGA-II was the best algorithm, followed by mocDE which was better than MOEA/D and PAES. In DTLZ6, mocDE obtained the best results according to all the indicators. Figure V shows the Pareto front approximations obtained for DTLZ6.

Tables II, III, IV, and V show the complete results of mocDE, PAES, MOEA/D and NSGA-II for all the ZDT and DTLZ problems. The best results are shown in **boldface**.

VI. Conclusions and future work

In this paper, a MOEA based on the principles of *Compact Differential Evolution (cDE)* [11] was proposed. mocDE uses only one new solution in each iteration, allowing it to apply cDE concepts as in mono-objective problems. Nevertheless, mocDE has a non-dominated solution archive whose goal is to maintain the best set of solutions found and a replacement criteria for such solutions, helping in approximating the real Pareto front. The results obtained with our pe-mocDE indicate that it is very important to rely on elite individuals which can reach good search regions, and manage to avoid local optima due to the inherent randomness of the sampling process. Our proposed mocDE was found to be highly competitive or superior to several state-of-the-art MOEAs in 10 out of 12 problems and was very close to the best algorithm in the other two problems. This is remarkable if we consider that our proposed approach was designed to produce important memory savings so that it's suitable for being implemented in hardware. Our proposed approach also produced good distributions of solutions, although it was not better than NSGA-II in this regard. Therefore, it is necessary to investigate alternative

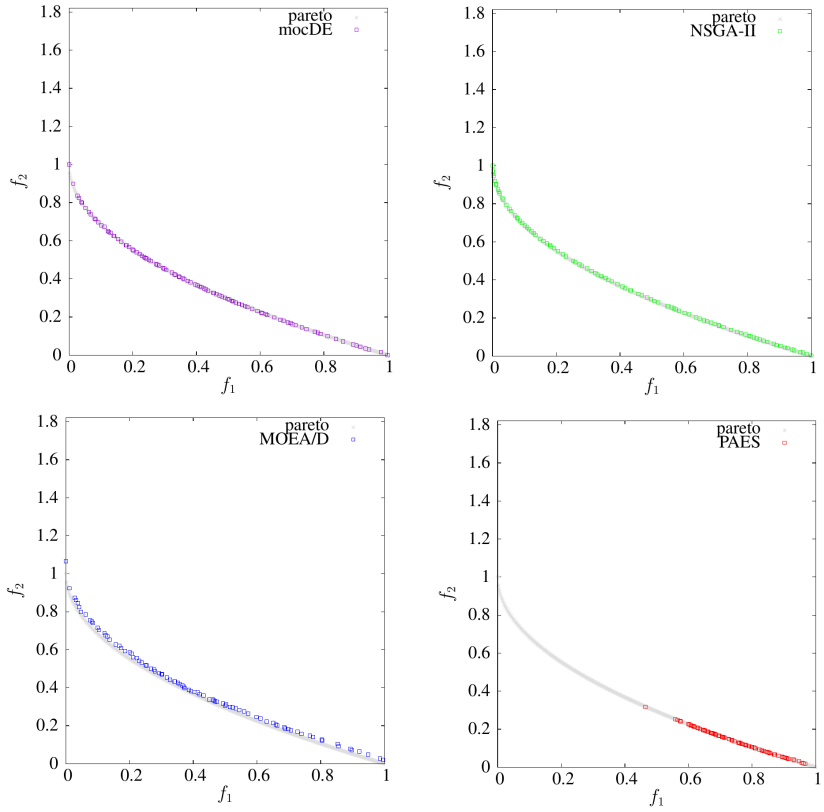


Fig. 3. Outcome of all algorithms when solving problem ZDT1.

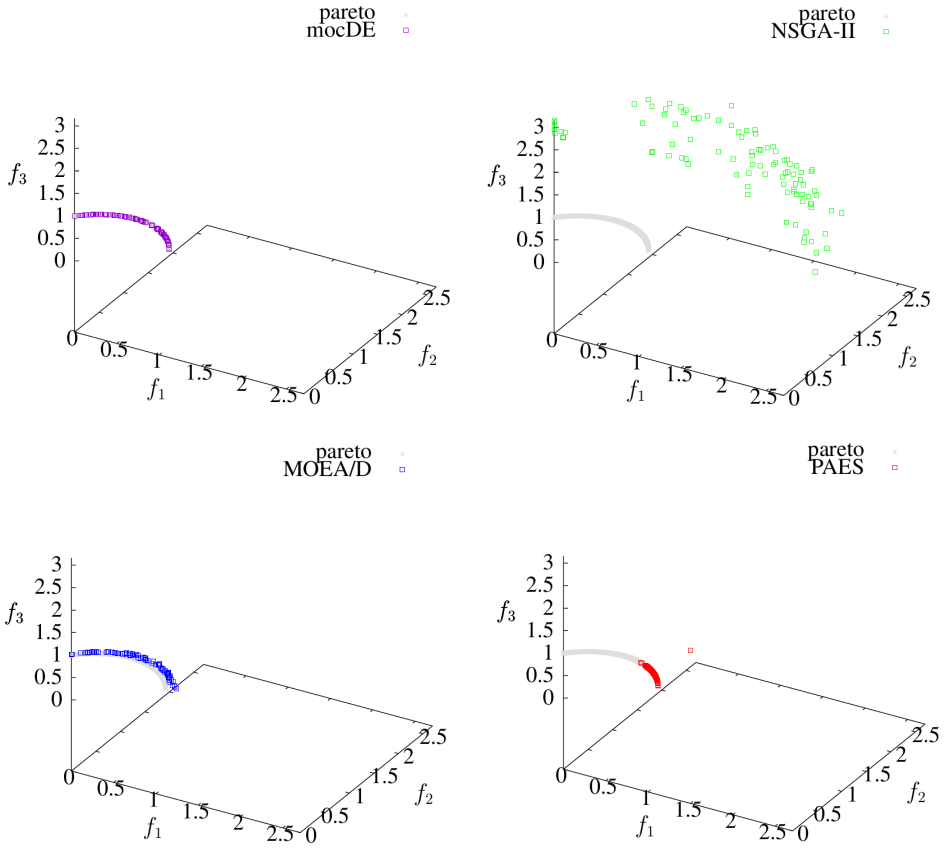


Fig. 4. Outcome of all algorithms when solving problem DTLZ6.

MOP	Indicator	mocDE		NSGA-II		MOEA/D		PAES	
		μ	σ	μ	σ	μ	σ	μ	σ
ZDT1	I_{IGD}	**0.0003**	**0.0000**	**0.0003**	**0.0000**	0.0008	0.0002	0.0004	0.0013
	I_H	**0.9988**	**0.0001**	0.9983	**0.0001**	0.9938	0.0016	0.8372	0.1714
ZDT2	I_{IGD}	**0.0003**	**0.0000**	**0.0003**	**0.0000**	0.0006	0.0002	0.0139	0.0070
	I_H	**0.9957**	**0.0002**	0.9926	0.0007	0.9787	0.0065	0.6757	0.0599
ZDT3	I_{IGD}	0.0021	0.0002	**0.0006**	**0.0000**	0.0068	0.0021	0.0310	0.0116
	I_H	0.9962	**0.0004**	**0.9975**	**0.0004**	0.9345	0.0197	0.7606	0.1506
ZDT4	I_{IGD}	0.0161	0.0077	0.0229	0.0121	**0.0068**	0.0080	0.0139	**0.0043**
	I_H	0.9921	**0.0037**	0.9893	0.0056	**0.9966**	0.0039	0.8968	0.1293
ZDT6	I_{IGD}	**0.0001**	**0.0000**	0.0011	0.0002	0.0002	0.0001	0.0022	0.0014
	I_H	**0.9985**	0.0018	0.9906	0.0014	0.9973	**0.0011**	0.5277	0.2614
DTLZ1	I_{IGD}	0.0164	0.0066	0.0839	0.0445	0.1371	0.1793	**0.0062**	**0.0035**
	I_H	**1.0000**	**0.0000**	0.9998	0.0004	0.9953	0.0122	0.9968	0.0015

TABLE II. VALUES OF THE UNARY PERFORMANCE MEASURES FOR ALL THE MOEAS COMPARED IN THE ZDT TEST PROBLEMS.

MOP	Indicator	mocDE		NSGA-II		MOEA/D		PAES	
		μ	σ	μ	σ	μ	σ	μ	σ
DTLZ2	I_{IGD}	**0.0008**	**0.0000**	**0.0008**	**0.0000**	**0.0008**	**0.0000**	0.0052	0.0008
	I_H	**0.8956**	0.0032	0.8768	0.0098	0.8781	**0.0025**	0.3937	0.0652
DTLZ3	I_{IGD}	0.0321	0.0152	0.1765	0.0665	0.2794	0.3707	**0.0120**	**0.0061**
	I_H	**1.0000**	**0.0000**	0.9998	0.0002	0.9963	0.0093	0.9958	0.0017
DTLZ4	I_{IGD}	0.0035	0.0007	**0.0012**	**0.0000**	0.0029	0.0024	0.0143	0.0019
	I_H	0.8454	0.0255	**0.8900**	**0.0085**	0.8598	0.0687	0.2006	0.1358
DTLZ5	I_{IGD}	**0.0001**	**0.0000**	**0.0001**	**0.0000**	**0.0001**	**0.0000**	0.0028	0.0012
	I_H	0.9830	0.0008	**0.9974**	**0.0006**	0.9809	0.0019	0.6649	0.1216
DTLZ6	I_{IGD}	**0.0001**	**0.0000**	0.0174	0.0010	0.0006	0.0005	0.0039	0.0016
	I_H	**0.9995**	**0.0000**	0.7912	0.0188	0.9953	0.0043	0.8222	0.0290
DTLZ7	I_{IGD}	0.0021	**0.0001**	**0.0009**	**0.0001**	0.0036	0.0027	0.0089	0.0024
	I_H	**0.9646**	**0.0015**	0.9624	0.0059	0.8988	0.0229	0.6423	0.2512

TABLE III. VALUES OF THE UNARY PERFORMANCE MEASURES FOR ALL THE MOEAS COMPARED IN THE DTLZ TEST PROBLEMS.

archiving methods that allow us to improve the distribution of the solutions obtained. It would also be interesting to test our proposed approach in additional multi-objective test instances such as the WFG and CEC 2009 benchmarks, as well as to compare our proposed approach against newer variants of MOEAs, reported in recent CEC competitions. Finally, it would be interesting to analyze the CPU usage of our proposed mocDE to determine if it could be used in real-time applications.

ACKNOWLEDGMENT

The first author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN. The second author acknowledges support from CONACyT project no. 221551. The third author acknowledges support from IPN-ESIME through the COTEBAL, SIP/EDI and SIBE programs.

REFERENCES

[1] K. Miettinen, *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, Boston, 1999.

[2] P. Larrañaga and J. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[3] S. I. V. Peña, S. B. Rionda, and A. H. Aguirre, "Multiobjective shape optimization using estimation distribution algorithms and correlated information," in *Evolutionary Multi-Criterion Optimization*. Springer, 2005, pp. 664–676.

[4] S. Bureerat and K. Sriworamas, "Population-based incremental learning for multiobjective optimisation," in *Soft Computing in Industrial Applications*. Springer, 2007, pp. 223–232.

[5] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," School of Computer Science, Carnegie Mellon University, Pittsburg, Pennsylvania 15213, Tech. Rep. CMU-CS-94-163, 1994.

[6] G. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," in *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, 1998, pp. 523–528.

[7] C. W. Ahn and R. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367 – 385, August 2003.

[8] E. Mininno, F. Cupertino, and D. Naso, "Real-valued compact genetic algorithms for embedded microcontroller optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 203–219, 2008.

[9] W. Gautschi, "Error function and Fresnel integrals," in *Handbook of Mathematical Functions*, M. Abramowitz and I. Stegun, Eds. New York: Dover, 1965, ch. 7, pp. 295–329.

[10] W. J. Cody, Jr., "Rational Chebyshev approximations for the error function," *Mathematics of Computation*, vol. 23, no. 107, pp. 631–637, Jul. 1969. [Online]. Available: http://www.jstor.org/stable/2004390

[11] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32 –54, February 2011.

[12] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, pp. 173–195, 2000.

[13] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. USA: Springer, 2005, pp. 105–145.

[14] E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study," in *Parallel Problem Solving from Nature V*, A. E. Eiben, Ed. Amsterdam: Springer-Verlag, September 1998, pp. 292–301.

[15] D. A. V. Veldhuizen and D. A. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Evolutionary Computation, Tech. Rep., 1999.

[16] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 117–132, 2002.

MOP	Ind.	Alg A	Alg B			
			mocDE	NSGA-II	MOEA/D	PAES
ZDT1	I_C	mocDE	– – –	**0.3072**	**0.9536**	**0.0357**
		NSGA-II	0.0076	– – –	**0.9061**	0.0179
		MOEA/D	0.0004	0.0003	– – –	0.0047
		PAES	0.0127	**0.2043**	**0.4496**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0121**	**0.0030**	**0.0106**
		NSGA-II	0.0127	– – –	**0.0036**	**0.0115**
		MOEA/D	0.0235	0.0223	– – –	**0.0197**
		PAES	0.3677	0.3661	0.3528	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0933**	**1.0155**	**1.0562**
		NSGA-II	1939.8397	– – –	**3.7745**	**2.5625**
		MOEA/D	19415.3127	18.0051	– – –	**16.8749**
		PAES	352979.4390	69357.5846	60707.0336	– – –
ZDT2	I_C	mocDE	– – –	**0.3780**	**0.8545**	**0.0291**
		NSGA-II	0.0046	– – –	**0.7233**	0.0177
		MOEA/D	0.0001	0.0459	– – –	0.0091
		PAES	0.0069	**0.2018**	**0.3836**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0091**	**0.0040**	**0.0093**
		NSGA-II	0.0125	– – –	**0.0059**	**0.0113**
		MOEA/D	0.0409	0.0357	– – –	**0.0185**
		PAES	0.5870	0.5851	0.5722	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0144**	**1.0054**	**1.0146**
		NSGA-II	5045.5631	– – –	**1.7980**	**1.1681**
		MOEA/D	39864.9492	12.8372	– – –	**2.5276**
		PAES	585684.8224	259213.3343	249077.7582	– – –
ZDT3	I_C	mocDE	– – –	**0.1542**	**0.9779**	0.0439
		NSGA-II	0.0997	– – –	**0.9935**	0.0344
		MOEA/D	0.0004	0.0000	– – –	0.0059
		PAES	**0.0564**	**0.1551**	**0.4510**	– – –
	$I_{\epsilon}+$	mocDE	– – –	0.0218	**0.0011**	**0.0114**
		NSGA-II	**0.0070**	– – –	**0.0000**	**0.0065**
		MOEA/D	0.1442	0.1424	– – –	**0.1296**
		PAES	0.5956	0.5942	0.5224	– – –
	$I_{\epsilon}*$	mocDE	– – –	**291.8517**	**1.0010**	**141.5607**
		NSGA-II	2212.7811	– – –	**1.0000**	**322.0125**
		MOEA/D	110170.1276	108249.1763	– – –	**27228.4280**
		PAES	460754.0149	459013.5366	377671.7812	– – –
ZDT4	I_C	mocDE	– – –	**0.6562**	0.1417	0.0146
		NSGA-II	0.3078	– – –	0.1058	0.0135
		MOEA/D	**0.8143**	**0.8510**	– – –	0.0241
		PAES	**0.7710**	**0.7489**	**0.5783**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0525**	0.1750	0.1989
		NSGA-II	0.1615	– – –	0.2684	0.2702
		MOEA/D	**0.0454**	**0.0322**	– – –	**0.0906**
		PAES	**0.1719**	**0.1507**	0.2267	– – –
	$I_{\epsilon}*$	mocDE	– – –	2.8921	**8.7886**	**2.1943**
		NSGA-II	**2.1174**	– – –	**12.8377**	**2.8963**
		MOEA/D	44.5183	47.3170	– – –	**1.8115**
		PAES	96014.7412	102119.2899	13659.7303	– – –
ZDT6	I_C	mocDE	– – –	**0.9846**	**0.8412**	**0.0077**
		NSGA-II	0.0008	– – –	0.0004	0.0099
		MOEA/D	0.0008	**0.9849**	– – –	0.0068
		PAES	0.0001	**0.4649**	**0.4043**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0021**	**0.0123**	**0.0083**
		NSGA-II	0.0826	– – –	0.0740	**0.0827**
		MOEA/D	0.0170	**0.0002**	– – –	**0.0156**
		PAES	0.4066	0.4043	0.4062	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0052**	**1.0209**	41.9239
		NSGA-II	1234.2914	– – –	135.2997	50818.6292
		MOEA/D	129.3453	**1.0004**	– – –	5325.9426
		PAES	**2.6926**	**2.4380**	**2.4569**	– – –
DTLZ1	I_C	mocDE	– – –	**0.9904**	**0.6557**	0.0264
		NSGA-II	0.0017	– – –	0.3874	0.0003
		MOEA/D	0.2392	**0.4418**	– – –	0.0710
		PAES	**0.4356**	**0.6258**	**0.5366**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0086**	**0.2152**	0.5467
		NSGA-II	2.7682	– – –	**1.7941**	2.8995
		MOEA/D	4.3779	3.7240	– – –	4.4261
		PAES	**0.3736**	**0.3630**	**0.3730**	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0114**	**1.7860**	**4.1740**
		NSGA-II	349976.4876	– – –	16571.9186	**82.7071**
		MOEA/D	3788.8168	**689.2635**	– – –	**89.0216**
		PAES	361903.4032	252385.3753	67254.1126	– – –

TABLE IV. VALUES OF THE BINARY PERFORMANCE MEASURES FOR ALL THE MOEAS COMPARED IN THE ZDT TEST PROBLEMS.

MOP	Indicator	Alg A	Alg B			
			mocDE	NSGA-II	MOEA/D	PAES
DTLZ2	I_C	mocDE	– – –	**0.0222**	**0.2418**	**0.0061**
		NSGA-II	0.0044	– – –	**0.0360**	0.0027
		MOEA/D	0.0068	0.0046	– – –	0.0018
		PAES	0.0032	**0.0059**	**0.0160**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.1014**	**0.0437**	**0.0903**
		NSGA-II	0.1800	– – –	0.1787	**0.0933**
		MOEA/D	0.0979	**0.1100**	– – –	**0.1036**
		PAES	0.7331	0.7056	0.7331	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.1735**	**1.0862**	**1.1523**
		NSGA-II	144748.4553	– – –	37431.2771	**9.8529**
		MOEA/D	67.0391	**15.4861**	– – –	**1.9911**
		PAES	674812.1680	252530.0816	324288.2939	– – –
DTLZ3	I_C	mocDE	– – –	**0.9996**	**0.6180**	0.0420
		NSGA-II	0.0000	– – –	**0.4042**	0.0009
		MOEA/D	0.2357	0.3929	– – –	0.1050
		PAES	**0.3546**	**0.5249**	**0.4590**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.0012**	**0.5999**	1.4145
		NSGA-II	8.1380	– – –	**4.9019**	8.4637
		MOEA/D	12.6178	11.1733	– – –	12.6794
		PAES	**0.9886**	**0.9761**	**0.9877**	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0003**	**1.4620**	**2.9935**
		NSGA-II	416845.4902	– – –	7462.1463	**507.9522**
		MOEA/D	5891.1202	**1662.3479**	– – –	**182.9857**
		PAES	921766.6866	632961.9617	132743.2069	– – –
DTLZ4	I_C	mocDE	– – –	**0.0188**	**0.1080**	0.0172
		NSGA-II	0.0065	– – –	**0.0229**	**0.0068**
		MOEA/D	0.0111	0.0100	– – –	0.0026
		PAES	**0.0614**	0.0038	**0.0384**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.1662**	**0.1425**	**0.0133**
		NSGA-II	0.1849	– – –	0.1846	**0.1174**
		MOEA/D	0.1586	**0.1559**	– – –	**0.0114**
		PAES	0.9638	0.9521	0.9623	– – –
	$I_{\epsilon}*$	mocDE	– – –	**4.5330**	**27.1076**	**26.4156**
		NSGA-II	135583.1489	– – –	93819.9909	**93625.9635**
		MOEA/D	18451.1499	**896.8074**	– – –	**4.9309**
		PAES	894503.4408	452850.1570	490909.7774	– – –
DTLZ5	I_C	mocDE	– – –	0.0020	**0.3023**	0.0214
		NSGA-II	**0.0782**	– – –	**0.1903**	**0.0430**
		MOEA/D	0.0377	0.0011	– – –	0.0143
		PAES	**0.0542**	0.0043	**0.1177**	– – –
	$I_{\epsilon}+$	mocDE	– – –	0.0262	**0.0046**	**0.0216**
		NSGA-II	**0.0078**	– – –	**0.0075**	**0.0081**
		MOEA/D	0.0095	0.0265	– – –	**0.0220**
		PAES	0.5499	0.5499	0.5499	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.0444**	**1.0077**	**1.0372**
		NSGA-II	80.7762	– – –	57.5902	**1.0133**
		MOEA/D	6.5056	**1.8791**	– – –	**1.0453**
		PAES	549916.7838	79694.1601	356232.9212	– – –
DTLZ6	I_C	mocDE	– – –	**0.9983**	**0.9451**	**0.0104**
		NSGA-II	0.0000	– – –	0.0000	0.0002
		MOEA/D	0.0000	**0.9937**	– – –	0.0058
		PAES	0.0012	**0.7497**	**0.3875**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**-0.0000**	**0.0037**	**0.0165**
		NSGA-II	1.3309	– – –	1.3205	1.2812
		MOEA/D	0.0798	**0.0062**	– – –	**0.0612**
		PAES	0.6345	**0.6345**	0.6345	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.6239**	**4.2058**	**1.5164**
		NSGA-II	171.1133	– – –	97.0971	**23.2563**
		MOEA/D	9.3099	**1.1650**	– – –	**1.9309**
		PAES	572759.7626	68363.3602	317401.6423	– – –
DTLZ7	I_C	mocDE	– – –	**0.1312**	**0.5262**	**0.0483**
		NSGA-II	0.0020	– – –	**0.2680**	0.0422
		MOEA/D	0.0008	0.0186	– – –	0.0156
		PAES	0.0048	**0.0794**	**0.1131**	– – –
	$I_{\epsilon}+$	mocDE	– – –	**0.1542**	**0.0730**	**0.1259**
		NSGA-II	0.1736	– – –	**0.0940**	**0.0905**
		MOEA/D	0.7488	0.7118	– – –	**0.3311**
		PAES	1.9071	1.8674	1.5094	– – –
	$I_{\epsilon}*$	mocDE	– – –	**1.1983**	**3.9230**	**1.1368**
		NSGA-II	37462.4683	– – –	1851.1962	**19.1559**
		MOEA/D	309.8735	**27.6673**	– – –	**4.8843**
		PAES	283791.0577	114238.3751	97017.9213	– – –

TABLE V. VALUES OF THE BINARY PERFORMANCE MEASURES FOR ALL THE MOEAS COMPARED IN THE DTLZ TEST PROBLEMS.