Two Decomposition-based Modern Metaheuristic Algorithms for Multi-objective Optimization – A Comparative Study

Miguel A. Medina[1], Swagatam Das[2], Carlos A. Coello Coello[3], Juan M. Ramirez[4]
[1,3,4]Centro de Investigación y de Estudios Avanzados del IPN.[1,4]Unidad Guadalajara., [3]Unidad Zacatenco.
[1,4]Guadalajara, Jalisco., [3]Mexico, D.F., MEXICO
[2]Electronics and Communications Sciences Unit, Indian Statistical Institute, Kolkata 700 108, India
[1,4]{mmedina, jramirez}@gdl.cinvestav.mx, [3]ccoello@cs.cinvestav.mx

***Abstract*—This article presents the multi-objective variants of two popular metaheuritics of the current interest namely, the artificial bee colony algorithm, and the teaching-learning-based optimization algorithm to solve the real-parameter, bound constrained multi-objective optimization problems. The multi-objective variants are based on a decomposition approach, where a multi-objective optimization problem is decomposed into a number of scalar optimization sub-problems and these are optimized simultaneously. The proposed algorithms are tested on seven unconstrained benchmarks proposed for the special session and competition on multi-objective optimizers held under IEEE CEC (Congress on Evolutionary Computation) 2009 and also on five classical bi-objective test instances. The proposed approaches are compared with two other decomposition-based multi-objective evolutionary algorithms which are representative of the state-of-the-art in the area. Our results indicate that the proposed approaches are highly competitive in most of the test instances with respect to the algorithms in comparison.**

Keywords—Multi-objective optimization; artificial bee colony; teaching-learning algorithm; decomposition approach

I. INTRODUCTION

In many practical applications, there are several (possibly conflicting) objectives that need to be optimized simultaneously. These applications can be modeled as multi-objective optimization problems (MOP). In this kind of problems since the objectives are in conflict with one another, there no longer exists a single optimal solution, but rather a whole set of possible solutions of equivalent quality. A Pareto-optimal solution to an MOP is a candidate for the best trade-offs among the objectives. The Pareto set/front is the set of all Pareto-optimal solutions in the decision/objective space.

The past few years have witnessed significant progress in the development of evolutionary algorithms for dealing with multi-objective optimization problems, comprehensive surveys can be found in [1, 2]. The major advantage of these multi-objective evolutionary algorithms (MOEAs) over other methods is that they work with a population of potential solutions and therefore they can produce a set of Pareto-optimal solutions to approximate the Pareto front in a single run. In recent years several meta-heuristics such as: particle swarm optimization (PSO) [3] which imitates the collective intelligence emerging from the group behavior of natural creatures; artificial bee colony (ABC) [4] which works on the foraging behavior of a honey bee, teaching-learning-based optimization (TLBO) [5] which works on the philosophy of teaching and learning, etc. have been applied to solve the multi-objective optimization problems.

The majority of the existing MOEAs aim at producing a number of Pareto-optimal solutions as diverse as possible to approximate the whole Pareto front. Therefore, these methods need some other techniques for ranking solutions (e.g., crowding distance, fitness sharing, niching). Among these methods, non-dominated sorting genetic algorithm II (NSGA-II) [6], and strength Pareto evolutionary algorithm 2 (SPEA2) [7] have received much attention in the real world applications. However, it is shown that these methods cannot always provide good results, especially when the multi-objective optimization problem is complicated [8, 9].

Recently, a new MOEA framework, multi-objective evolutionary algorithm based on decomposition (MOEA/D) [8], has been proposed. This framework decomposes an MOP into several single-objective optimization sub-problems with neighborhood relations. In this way, a set of approximate solutions to the Pareto front is achieved by minimizing each sub-problem instead of using Pareto ranking. This has given rise to a new generation of multi-objective evolutionary algorithms.

In this paper, we present a modified ABC and a TLBO algorithm in the MOEA/D framework. The performance of our proposed approaches is compared with two MOEAs/D, which are representative of the state-of-the-art in the area: MOEA/D-DE [9], and MOEA/D-DRA [10]. The algorithms are tested over seven unconstrained MOP taken from the IEEE Congress on Evolutionary Computation 2009 [11] and five widely used bi-objective ZDT test instances [12]. The comparison is made in terms of the inverted generational distance (IGD) [13]. Our comparison results indicated that our proposed approaches are highly competitive and suggest that can be appear as very promising meta-heuristics candidates in the domain of multi-objective optimization problems.

Organization of the remaining paper is in order. Section II provides the basic background of the multi-objective

optimization problem. In section III, the general framework of the proposed approaches is summarized. Section IV presents and discusses the results of our comparative study. Finally, the paper is concluded in Section V.

II. SCIENTIFIC BACKGROUND

A. Multi-objective optimization

A multi-objective optimization problem may be formulated as follows,

$$\begin{aligned} &\min \quad F(x) = \{f_1(x), \ldots, f_m(x)\} \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{1}$$

where x is the vector of decision variables and Ω is the feasible region within the decision space. F: $\Omega \rightarrow R^m$ is defined as the vector of m objective functions

In multi-objective optimization, the goal is to find the best possible trade off among the objectives since, frequently, one objective can be improved only at the expense of worsening another. To describe the concept of optimality for problem (1) the following definitions are provided [14]:

Definition 1. Let $x, y \in \Omega$, such that $x \neq y$, we say that x dominates y (detonated by $x \prec y$) if and only if $f_i(x) \leq f_i(y)$ for all $i = 1, ..., m$.

Definition 2. Let $x^* \in \Omega$, we say that x^* is a Pareto optimal solution, if there is no other solution $y \in \Omega$ such that $y \prec x^*$.

Definition 3. The Pareto Optimal Set (*PS*) is defined by $PS = \{x \in \Omega \,|\, x \text{ is Pareto Optimal Solution}\}$, while its image $PF = \{F(x) \,|\, x \in PS\}$ is called the Pareto Optimal Front.

B. Decomposition of a multi-objective optimization problem

It is well-known that a Pareto-optimal solution to a multi-objective optimization problem, under certain conditions, could be an optimal solution of a single objective optimization problem in which the objective is an aggregation function of all individual objectives. Therefore, approximation of the Pareto front can be decomposed into a number of single objective optimization sub-problems. This is the basic idea behind many traditional mathematical programing methods for approximating the Pareto front. Several methods for constructing aggregation functions can be found in the literature (see for example [15]). These methods use a weighted vector to define a scalar function. In this way and under certain assumptions (e.g., the minimum is unique, the weighting coefficients are positive, etc.) a Pareto optimal solution is achieved by minimizing such function. Among these methods, probably the two most widely used are the *Tchebycheff* and the *Weighted Sum* approaches.

In this paper, the weighted *Tchebycheff* approach is used to decompose the multi-objective optimization problems. Under this scheme, the scalar optimization problem can be stated as [15]:

$$\begin{aligned} &\text{Minimize} \quad g\left(x \,|\, w, z^*\right) = \max_{i \in \{1,..,m\}} \left\{ w_i \left| f_i(x) - z_i^* \right| \right\} \\ &\text{Subject to} \quad x \in \Omega \end{aligned} \tag{2}$$

where $w = (w_1,..,w_m)$ is a weighting vector and $w_i \geq 0$ for all $i = 1,..., m$. $\Sigma\, w_i = 0$ and $z^* = (z^*_1, \ldots, z^*_m)$ represent the reference point, i. e., $z^* = \min\{f_i(x) \,|\, x \in \Omega\}$, $i = 1, \ldots, m$.

For each Pareto-optimal solution x^* there exists a weighting vector w such that x^* is the optimal solution of (2), and each optimal solution is a Pareto-optimal solution for (1). Therefore, it is possible to obtain different Pareto optimal solutions using different weighting vectors w.

C. Modified Artificial Bee Colony

The first framework of the Artificial Bee Colony (ABC) was introduced by Karaboga in 2005 as a new swarm intelligent technique inspired by the foraging behavior of a honey bee swarm [16]. The colony of artificial bees consists of three groups of bees: employed bees, onlooker bees, and scout bees. In the algorithm, the position of a food source represents a possible solution to the optimization problem, and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. Each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of food sources existing around the hive (number of solutions in the population). The employed bee whose food source has been abandoned becomes a scout.

Akay and Karaboga [17] proposed some modifications to the standard ABC algorithm in order to improve the convergence rate. The main steps of the modified ABC algorithm proposed by Akay and Karaboga can be summarized in the following way:

1) *Initial food sources*: At the first step, the algorithm generates a randomly distributed initial population (food sources position) within the range of the boundaries of the parameters,

$$x_{i,j} = x_j^{\min} + rand(0,1) \cdot (x_j^{\max} - x_j^{\min}) \tag{3}$$

where $i = 1,..., SN$, $j = 1,.., D$. SN is the number of food sources (potential solutions) and D is the number of optimization parameters. In addition, counters which store the numbers of *trials* of solutions are reset to 0 in this phase.

2) *Employed Phase:* An employed bee produces a modification on the position of the food source (solution) for finding a new food source (new solution), and then evaluates its quality (fitness value) of the new food source. Finding a new food source is defined by [17],

$$v_{i,j} = \begin{cases} x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) & \text{if } R_{i,j} < MR \\ x_{i,j} & otherwise \end{cases} \tag{4}$$

By means of this modification, for each parameter $x_{i,j}$, an uniformly distributed random number ($R_{i,j}$) within the range [0,1] is produced, then parameter $x_{i,j}$ is modified by (4) where MR is the modification rate, j is a random integer within the range [1,D] and $k \in \{1,...,SN\}$ is randomly chosen index that

has to be different from i. $\phi_{i,j}$ is a random number between [-1,1]. If a parameter value produced by this operation exceeds its predetermined boundaries, the parameter can be set to an acceptable value. After producing a new solution (v_i) within the boundaries, a fitness value for a minimization problem is assigned to the solution (v_i) given by,

$$fitness_i = \begin{cases} 1/(1+f_i) & \text{if } f_i \geq 0 \\ 1+abs(f_i) & \text{if } f_i < 0 \end{cases} \quad (5)$$

Where f_i is the cost value of the solution v_i. A greedy selection is applied between x_i and v_i; therefore the better one is selected depending on its fitness values. If the new source at v_i is equal or better than the old source x_i in terms of quality, the employed bee memorizes the new position and forgets the old one. Otherwise the previous position is kept in memory. If x_i cannot be improved, its counter of the number of *trials* is incremented by 1; otherwise, the counter is reset to 0.

3) *Calculate probability:* After all employed bees complete their searches; an onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source site with a probability related to its nectar amount. This probabilistic selection depends on the fitness values of the solutions in the population and is calculated by the following expression,

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad (6)$$

In this probabilistic selection scheme, the fitness value of the solution i evaluated is proportional to the nectar amount of the food source in the position i. In other words, as the nectar amount of food sources (the fitness of solutions) increases, the number of onlookers visiting them increases, too. In this way, the employed bees exchange their information with the onlookers.

4) *Onlooker Phase:* In this phase a random number (r_i) within the range [0,1] is generated for each source. If the probability value p_i associated with that source is greater than r_i then the onlooker bee produce a modification on the position of this food source by using (4). As in case of the employed phase, after the source is evaluated, greedy selection is applied and the onlooker bee either memorizes the new position by forgetting the old one or keeps the old one. If solution x_j cannot be improved, its counter of the number of trials is incremented by 1; otherwise, the counter is reset to 0.

5) *Scout Phase:* In a cycle, after all employed bees and onlooker bees complete their searches, the algorithm checks to see if there is any exhausted source to be abandoned. In order to decide if a source is to be abandoned, the counters which have been updated during search are used. If the value of the counter is greater than the control parameter, known as the "*limit*", then the source associated with this counter is abandoned. The food source abandoned by its bee is replaced with a new food source discovered by the scout. This is simulated by randomly produce a new position by using (3) and replacing it with the abandoned one. It is assumed that only one source can be abandoned in each cycle. If more than one counter exceeds the "*limit*" value, one of the maximum ones might be chosen.

D. Teaching-Learning based optimization algorithm

The original teaching learning based optimization (TLBO) was proposed by Rao *et al*. [18] to obtain global solutions for continuous non-linear functions. In this algorithm, the design variables are analogous to different subjects offered to learners. The learners' grade is analogous to the 'fitness' as in any other evolutionary algorithm, and the teacher is considered as the best solution obtained so far. Hence, the TLBO's performance is based on two main phases: the teacher phase, which involves learning from the teacher, and the learner phase, which involves learning through the interaction among learner [18].

1) Teacher Phase: In this phase, each class consists of a number of learnears with diffetents grades; the learner with the best grade is selected as the teacher. The teacher tries to help learners to get good marks or grades. Therefore, a teacher increases the mean of the class according to his/her capability.

Let M_i be the mean of the class and $M_{best,i}$ be the best solution so far and therefore the teacher in the *i-th* iteration. Hence, $M_{best,i}$ will try to move the mean M_i towards its own level. Thus, the new mean will be $M_{best,i}$, designated as $M_{new,i}$. The solutions are updated according to the difference between the mean of the class (M_i) and the new mean ($M_{new,i}$) by [18],

$$x_{new,i} = x_{old,i} + r_i \left(M_{new,i} - T_F M_i \right) \quad (7)$$

where T_F is a teaching factor that determines the mean value to be changed; r_i is a random number within [0, 1]. The value of T_F can be either 1 or 2, which is decided randomly with equal probability as T_F = round [1 + rand (0, 1)]. The new solutions (x_{new}) are accepted if they give better function values. The algorithm uses the best solution of the iteration to change the existing solution, thereby increasing the convergence rate.

2) Learning phase: A learner interacts randomly with other learners through group discussions, presentations, formal communications, etc. [18]. Thus, each learner can acquire new knowledge if the others have more knowledge than him/her. Learners' modification is expressed as follows,

$$\begin{aligned} &\textit{for } i = 1 \text{ to number of learners} \\ &\quad \textit{Randomly} \text{ select one learner } x_j, \text{ such that } x_i \neq x_j \\ &\quad \textit{if } f(x_i) < f(x_j) \\ &\qquad x_{new,i} = x_i + r_i(x_i - x_j) \\ &\quad \textit{else} \\ &\qquad x_{new,i} = x_i + r_i(x_j - x_i) \\ &\quad \textit{end if} \\ &\textit{end for} \end{aligned} \quad (8)$$

The new solutions (x_{new}) are accepted if they give better function values. The new solutions (x_{new}) update the initial learners and the teaching-learning process continues until the termination criterion is achieved.

III. MULTI-OBJECTIVE ALGORITHMS BASED ON DECOMPOSITION

A. Muti-objective modified Artificial Bee Colony

The proposed Multi-objective modified Artificial Bee Colony Algorithm based on Decomposition (MOABC/D) utilizes the *Tchebycheff* approach to decompose the multi-objective problem into N scalar optimization sub-problems by choosing N weight vectors. The proposed MOABC/D solves these sub-problems simultaneously by evolving a population of solutions that mimics the intelligent behavior of a honey bee swarm in a similar way as is described in the modified ABC [17]. Neighborhood relations among these sub-problems are defined by computing the minimum Euclidean distances between the weighted vectors. The neighborhood of each sub-problem represents an artificial colony and the group of bees: employed, onlooker and scout bees are responsible to solve each sub-problem by using mainly the information from its neighboring sub-problems.

In the proposed algorithm, each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source has been abandoned by its bee becomes a scout. The position of a food source represent a potential solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution.

The main steps of the proposed MOABC/D can be summarized as follows,

1) Employed Phase: In the phase, the neighborhood of the *j-th* sub-problem becomes an artificial colony. This colony can be expressed as,

$$C_j = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{Tn,1} & x_{Tn,2} & \cdots & x_{Tn,D} \end{bmatrix} \quad (9)$$

where the subscript T_n is the size of the neighborhood, and D is the number of design variables. In this phase, an employed bee produces a modification on the position of the food source (solution) for finding a new food source (new solution), and then evaluates its quality (fitness value) of the new food source. In MOABC/D, a new food source (v_i) is generated by (4). If a parameter value produced by this operation exceeds its predetermined boundaries, the parameter is set to an acceptable value.

After producing a new solution (v_i) within the boundaries, a fitness value for a minimization problem is assigned to the solution (v_i) by mean (5). A greedy selection is applied after the new source is evaluated. If solution x_j cannot be improved, its counter of the number of *trials* is incremented by 1, otherwise, the counter is reset to 0.

2) Calculate probability: In this phase, an onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source site with a probability related to its nectar amount. This probabilistic selection is calculated by (6).

3) Onlooker Phase: In this phase a random number (r_i) within the range [0,1] is generated for each source. If the probability value p_i associated with that source is greater than *ri* then the onlooker bee produce a modification on the position of this food source by using,

$$v_{j,d} = \begin{cases} x_{j,d} + \phi_{j,d}(x_{i,d} - x_{k,d}) & \text{if } R_{j,d} < MR \\ x_{j,d} & otherwise \end{cases} \quad (10)$$

where index j corresponds to the current index of *j-th* sub-problem, x_i is the food source which its probability value (p_i) is greater than (r_i); the index d is a random integer within the range [1, D]. ($\phi_{j,d}$) is a random number between [-1,1], ($R_{j,d}$) is an uniformly distributed random number within the range [0,1] and the MR is the modification rate.

As in case of the employed phase, after the new source is evaluated, greedy selection is applied. If solution x_j cannot be improved, its counter of the number of *trials* is incremented by 1, otherwise, the counter is reset to 0.

4) Scout Phase: In a cycle, after all employed bees and onlooker bees complete their searches, the algorithm checks to see if there is any exhausted source to be abandoned. In order to decide if a source is to be abandoned, the counters which have been updated during search are used. If the value of the counter is greater than the control parameter, known as the "*limit*", then the source associated with this counter is abandoned. The food source abandoned by its bee is replaced with a new food source discovered by the scout. This is simulated by randomly produce a new position by using (3) and replacing it with the abandoned one. If more than one counter exceeds the "limit" value, one of the maximum ones might be chosen.

Summarizing, the proposed MOABC/D algorithm can be described in pseudo-code format in the following way:

Step1) Initialization

Step 1.1. Generate a well-distributed set of N weighting vectors $w^j = (w_1^j,\ldots,w_m^j)$, $j = 1,..N.$, and find the neighborhood of each sub-problems: $B(j) = \{w^j,...,w^j\}$ for j =1,..,N.

Step 1.2. Generated the initial population $\{x_1,..x_N\}$ according to (3) and evaluated its fitness. Set *trial* (j) = 0.

Step 1.3. Initialize the reference point z^*.

Step 2) For j = 1 to N **do**

Step 2.1. Determine the colony according to:

$$C = \begin{cases} B(j) & if\ rand < \delta \\ \{1,...,N\} & \text{otherwise} \end{cases}$$

where *rand* is a random number within [0,1] and δ the probability to select the neighborhood as the colony.

Step 2.2. Employed phase.

Step 2.3. Update the reference point z^*.

Step 2.4. **Update (s_r) solutions**. where (s_r) is the maximal number of solutions replaced by each new solution obtained.

Step 3) **For** j = 1 to N **do**

Step 3.1. Determine the colony according to:

$$C = \begin{cases} B(j) & if\ rand < \delta \\ \{1,...,N\} & \text{otherwise} \end{cases}$$

Step 3.2. Onlooker Phase

Step 3.3. Update the reference point z^*.

Step 3.4. Update (s_r) solutions.

Step 4) Scout Phase

Step 5) Stop Criterion: **If** the stop condition is satisfied, **then** stop MOABC/D and the output becomes: $\{x_1, .., x_N\}$ and $\{F(x_1),...,F(x_N)\}$. Otherwise go to **Step 2)**.

B. Multi-objective Teaching-Learning Algorithm

The proposed Multi-Objective Teaching Learning Algorithm based on Decomposition (MOTLA/D) utilizes the *Tchebycheff* approach to decompose the multi-objective problem into N scalar optimization sub-problems. The proposed approach solves these sub-problems simultaneously. Neighborhood relations among these sub-problems are defined by computing the minimum Euclidean distances between the weighted vectors. The neighborhood of each sub-problem represents a group of learners or a class, responsible to solve each sub-problem.

The main steps of the proposed MOTLA/D can be summarized as follows:

1) Teacher phase: In this pahse, for the *j-th* sub-problem, the size of the neighborhood becomes the number of learners in the class. This class can be expressed as,

$$C_j = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{Tn,1} & x_{Tn,2} & \cdots & x_{Tn,D} \end{bmatrix} \quad (11)$$

where the subscript T_n is the size of the neighborhood, and D is the number of design variables. Within the teacher phase, the mean of the class (M) for each design variable is calculated,

$$M = [m_1, m_2, ..., m_D] \quad (12)$$

The teacher (M_{new}) for the *j-th* sub-problem represents the best learner of the class C_j. Thus, the teacher is determined by

$$M_{new} = \{x_j \,|\, \min_{x_j \in Tn} g(x_j \,|\, w^j, z^*)\} \quad (13)$$

The solutions are updated according to the difference between the mean of the class (M) and the new mean (M_{new}) by,

$$x_{new,i} = x_{j,i} + r_i \left(M_{new,i} - T_F M_i \right) \quad (14)$$

where index j corresponds to the current index of *j-th* sub-problem, r_i is a random number within the range [0,1]. T_F is the teaching factor which value can be either 1 or 2, which is decided randomly with equal probability. The new solution (x_{new}) is accepted if it gives a better function value.

2) Learner Phase: In the phase for the *j-th* sub-problem, two learners x_i, and x_k are selected randomly such that $i \neq k \neq j$. A new solution (x_{new}) is generated as follows,

$$\begin{aligned} &if\ f(x_k) < f(x_i) \\ &\quad x_{new} = x_j + r_i(x_k - x_i) \\ &else \\ &\quad x_{new} = x_j + r_i(x_i - x_k) \\ &end \end{aligned} \quad (15)$$

Additionally, a polynomial mutation operator is applied to maintain solutions' diversity. The new solution (x_{new}) is accepted if it gives a better function value. If a parameter value produced in the teacher o learner phase exceeds its predetermined boundaries, the parameter is set to an acceptable value.

The proposed MOTLA/D may be summarized as follows,

***Step1)* Initialization**

Step 1.1. Generate a well-distributed set of N weighting vectors $w^j = (w_1^j,\ldots,w_m^j)$, $j = 1,..N$. and find the neighborhood of each sub-problems: $B(j) = \{w^j,...,w^j\}$ for $j = 1,..,N$.

Step 1.2. Generated the initial population $\{x_1,..x_N\}$ according to (3) and evaluated its fitness.

Step 1.3. Initialize the reference point z^*.

***Step 2)* For** $j = 1$ to N **do**

Step 2.1. Determine the class according to:

$$C = \begin{cases} B(j) & if\ rand < \delta \\ \{1,...,N\} & \text{otherwise} \end{cases}$$

Where *rand* is a random number within [0,1] and δ the probability to select the neighborhood as the class.

Step 2.2. Teacher phase

Step 2.3. Update the reference point z^*.

Step 2.4. **Update (s_r) solutions**. where (s_r) is the maximal number of solutions replaced by each new solution obtained.

***Step 3)* For** $j = 1$ to N **do**

Step 3.1. Determine the class according to:

$$C = \begin{cases} B(j) & if\ rand < \delta \\ \{1,...,N\} & \text{otherwise} \end{cases}$$

Step 3.2. Leaner Phase

Step 3.3. Update the reference point z^*.

Step 3.4. Update (s_r) solutions.

***Step 4)* Stop Criterion**: **If** the stop condition is satisfied, **then** stop MOTLA/D and the output becomes: $\{x_1, .., x_N\}$ and $\{F(x_1),...,F(x_N)\}$. Otherwise, go to **Step 2)**.

IV. EXPERIMENTAL RESULTS

In order to assess the performance our proposed algorithms, the MOABC/D and MOTLA/D have been compared with respect to two multi-objective evolutionary algorithms based on decomposition, which are representatives of the state-of-the-art in this area: MOEA/D-DE [9] and MOEA/D-DRA [10]. For this comparison we have used a set of well-known test instances. The quality of solutions obtained has been measured using the inverted generational distance (IGD) metric.

A. Test Problems

We have tested our proposed algorithms over two groups of well-known test functions. The first group consists on seven

unconstrained bi-objective functions (UF) taken from the CEC 2009 [11] and the second group consists of the bi-objective test suite of Zitzler-Deb-Thiele (ZDT) [12]. We used 30 decision variables for the UF, ZDT1, ZDT2 and ZDT3 test functions. ZDT4 and ZDT6 were tested using 10 decision variables.

B. Inverted Generational Distance

The concept of the generational distance was introduced by Van Veldhuizen and Lamont [13] as a way of estimating how far the Pareto-optimal solutions obtained by the algorithm are from those in the Pareto front of the problem. In this work, we implement an inverted general distance metric [11] in which we use as a reference the Pareto front, and we compare all its elements with respect to the Pareto-optimal solutions produced by the algorithms. This metric is described as follows,

Let P^* be a set of points uniformly distributed on the Pareto front and A be the approximation obtained by the algorithm. IGD represents the average distance from P^* to A defined as,

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \quad (16)$$

where $d(v, A)$ is the minimum Euclidean distance between v and the points in A. If the points in the set P^* can appropriately represent the Pareto front, IGD can be measure both the diversity and convergence of set A. The smaller is the value of this metric, better is the performance of the algorithm. A value of IGD equal to zero implies that all obtained solutions lies on the Pareto Front and they cover all the extension of the Pareto front.

C. Experimental Settings

For the UF instances, ten independent runs were performed with each algorithm whilst for the ZDT instances twenty independent runs were performed by each algorithm. The parameters used in each algorithm are summarized in table I, where S_{pop} represents the population size (600 for the UF instances and 150 for ZDT functions). N_{gen} represents the number of generations. It is worth mentioning that the stop condition of each algorithm is the number of function evaluations (300,000 function evaluations for UF instances and 30,000 for ZDT instances). S_r, is the number of solutions which are replaced in the neighborhood. Tn defines the neighbor size. Cr is the crossover rate. For MOABC/D the Cr is equal to its MR parameter. F is the scaling factor used in the MOEA/D [9]. η_m, is the mutation index. For the algorithms using the mutation operator, the mutation rate (P_m=1/n), was considered. δ is the probability of select solutions from the neighborhood. For MOEA/D-DRA, π_s and Δ_r, represent the percentage selection and decay rate for the utility, respectively; *limit* is the algorithm parameter of the MOABC/D. In table I, the number in parentheses indicate the values of the parameters used by each algorithm in the ZDT test instances. The value of the parameters used by MOEA/D-DRA in UF instances were taken from [10], and the control parameter of the remaining algorithms are selected through experiment.

For each test instance, the algorithms were evaluated using the IGD metric previously indicated. The results are summarized in table II and table III. Each of these tables presents the *average* and the standard deviation (in parentheses) of IGD metric for each test instance. The best results for each test instance are displayed in **boldface.**

Figures 1-12, show the final Pareto front obtained by the algorithms on the test instances. These plots show the final set of Pareto-optimal solutions found by each algorithm and correspond to the run with the lowest IGD value in each test instance.

TABLE I. PARAMETERS FOR MOABC/D, MOTLA/D AND MOEAS/D

Parameter	MOABC/D	MOTLA/D	MOEA/D-DE	MOEA/D-DRA
	UF - (ZDT)	UF - (ZDT)	UF - (ZDT)	UF - (ZDT)
S_{pop}	600 - (150)	600 - (150)	600 - (150)	600 - (150)
N_{gen}	250 - (100)	250 - (100)	500 - (200)	500 - (200)
Tn	60 - (30)	60 - (30)	60 - (30)	60 - (30)
S_r	2 - (2)	6 - (2)	6 - (2)	6 - (2)
δ	0.9 - (0.9)	0.9 - (0.9)	0.9 - (0.9)	0.9 - (0.9)
Cr	0.4 - (0.8)	-	1 - (0.5)	1 - (0.5)
limit	15 - (15)	-	-	-
η_m	-	20 - (20)	20 - (20)	20 - (20)
F	-	-	0.5 - (0.5)	0.5 - (0.5)
π_s	-	-	-	5
Δ_r	-	-	-	0.95

D. Discussion of results

As shown in table II and table III, our proposed algorithms outperformed both MOEA/D-DE and MOEA/D-DRA in most of the test problems with respect to the IGD metric.

It can be observed in Table II, that in terms of IGD metric, our proposed algorithms outperformed the MOEA/D-DE in all UF test instance and only in some test instances are better than MOEA/D-DRA. This table indicates that MOABC/D outperformed MOEA/D-DRA in UF4, UF5 and UF6 test instances meanwhile MOTLA/D outperformed MOAE/D-DRA only in UF3 test function.

As noticed in table III, the MOABC/D outperformed both MOEA/D-DE and MOEA/D-DRA in all ZDT test problems. This indicates that our algorithm achieved better convergence than these algorithms. Meanwhile MOTLA/D outperformed MOEAD/D-DRA in all test instances and outperformed MOEA/D-DE in most ZDT instances with exception of ZDT6. For ZDT4, the IGD values obtained by both MOEAs/D are very large compare with respect to the IGD values obtained by our proposed algorithms. This poor performance can be clearly seen in figure 11.

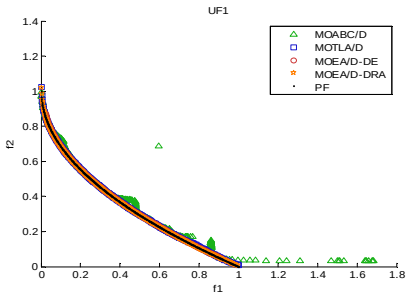

Figure 1. Approximated Pareto front obtained by MOEAs/D in UF1.

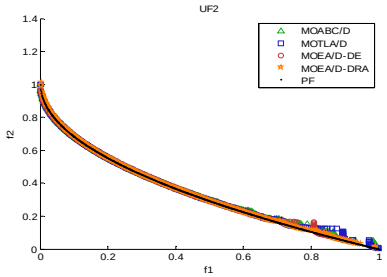

Figure 2. Approximated Pareto front obtained by MOEAs/D in UF2

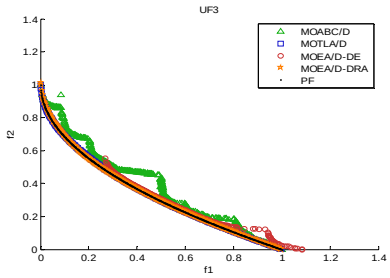

Figure 3. Approximated Pareto front obtained by MOEAs/D in UF3

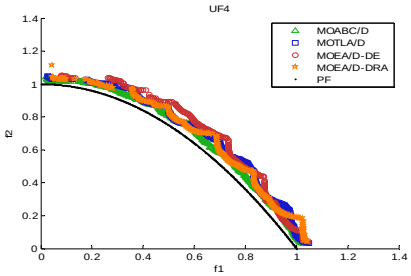

Figure 4. Approximated Pareto front obtained by MOEAs/D in UF4

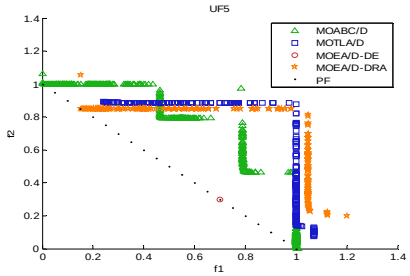

Figure 5. Approximated Pareto front obtained by MOEAs/D in UF5

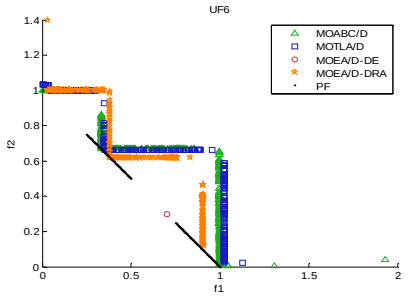

Figure 6. Approximated Pareto front obtained by MOEAs/D in UF6

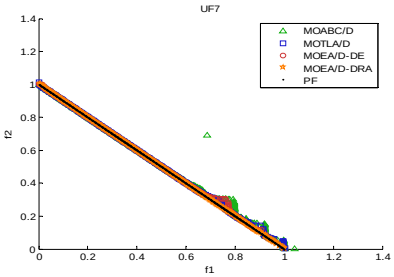

Figure 7. Approximated Pareto front obtained by MOEAs/D in UF7

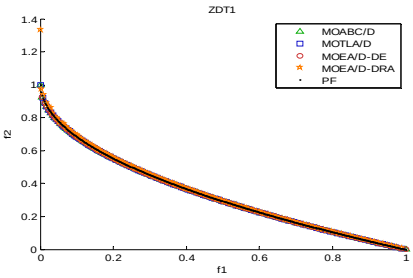

Figure 8. Approximated Pareto front obtained by MOEAs/D in ZDT1

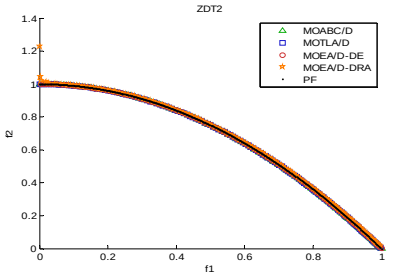

Figure 9. Approximated Pareto front obtained by MOEAs/D in ZDT2

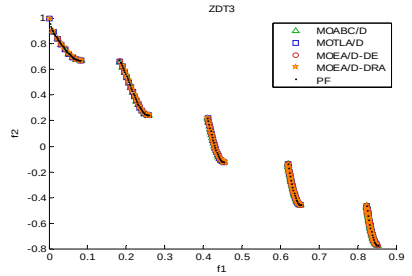

Figure 10. Approximated Pareto front obtained by MOEAs/D in ZDT3

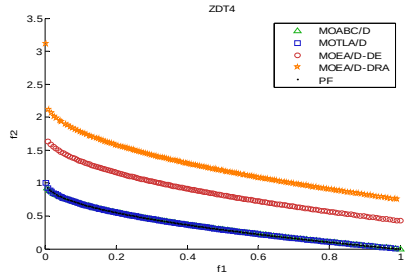

Figure 11. Approximated Pareto front obtained by MOEAs/D in ZDT4

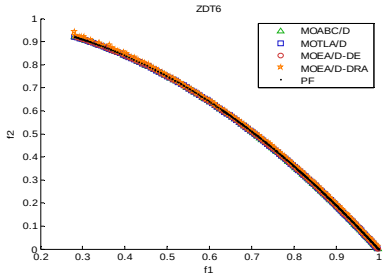

Figure 12. Approximated Pareto front obtained by MOEAs/D in ZDT6

TABLE II. RESULTS OF INVERTED GENERATIONAL METRIC

The IGD statistics based on 10 independent runs				
Test Instances	**MOABC/D**	**MOTLA/D**	**MOEA/D-DE**	**MOEA/D-DRA**
	Average (Std. Dev.)	***Average (Std. Dev.)***	***Average (Std. Dev.)***	***Average (Std. Dev.)***
UF1	0.023153 (0.005069)	0.004098 (0.001017)	0.057012 (0.052169)	**0.002588** (0.000695)
UF2	0.013172 (0.003303)	0.010337 (0.002558)	0.035477 (0.020442)	**0.005767** (0.002693)
UF3	0.069995 (0.019425)	**0.006579** (0.003987)	0.246116 (0.085832)	0.008762 (0.007035)
UF4	**0.041484** (0.001177)	0.065032 (0.005339)	0.079533 (0.005295)	0.059114 (0.005599)
UF5	**0.22254** (0.052683)	0.366491 (0.070979)	0.562263 (0.078414)	0.336049 (0.089490)
UF6	**0.147803** (0.233392)	0.49662 (0.169310)	0.595379 (0.173685)	0.366361 (0.294446)
UF7	0.012865 (0.003923)	0.004901 (0.000651)	0.408011 0.244096)	**0.002198** (0.000432)

TABLE III. RESULTS OF INVERTED GENERATIONAL METRIC

The IGD statistics based on 20 independent runs				
Test Instances	**MOABC/D**	**MOTLA/D**	**MOEA/D-DE**	**MOEA/D-DRA**
	Average (Std. Dev.)	***Average (Std. Dev.)***	***Average (Std. Dev.)***	***Average (Std. Dev.)***
ZDT1	**0.0026495** (5.59E-05)	0.0026875 (8.48E-05)	0.00288 (4E-05)	0.0043255 (4.93E-03)
ZDT2	0.0025575 (2.33E-05)	**0.0025395** (2.91E-05)	0.0026435 (6.08E-05)	0.0055125 (6.5E-03)
ZDT3	0.0069805 (5.9E-05)	**0.0065425** (1.10E-03)	0.0074185 (0.000172)	0.0079755 (4.83E-03)
ZDT4	0.002506 (1.81E-05)	**0.0023675** (1.83E-05)	1.783268 (0.75269)	1.6235585 (0.90677)
ZDT6	**0.001265** (6.06E-06)	0.002635 (1.46E-05)	0.001267 (4.44E-06)	0.008002 (1.72E-02)

V. CONCLUSIONS

In this paper, we presented two multi-objective optimization methods based on decomposition. The first method based on intelligent behavior of honey bees and the second one based on the teaching-learning process. Our proposed algorithms were able to outperform two MOEAs/D, which are representative of the state-of-the-art in the area, in most of the test instances.

The experimental results presented show that our proposed methods are highly competitive with respect to the algorithms in comparison and suggest that can be appear as very promising meta-heuristics candidates in the domain of multi-objective optimization problems.

REFERENCES

[1] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuiszen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed. New York: Springer, 2007.

[2] A. Zhou, B-Y. Qu, H. Li, S-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: a survey of the state-of-the-art", *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 32-49, Mar 2011.

[3] C. A. Coello Coello, G. T. Pulido and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 256--279, June 2004.

[4] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, and K. Ziarati, "A multiobjective artificial bee colony for optimizing multi-objective problems," International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. V5-277 - V5-281, August 2010.

[5] T. Niknam, F. Golestaneh, and M. S. Sadeghi, "θ-Multiobjective Teaching-Learning-Based optimization for Dynamic Economic Emission Dispatch," IEEE System Journal, vol. 6, no. 2, June 2012.

[6] K. Deb, S. Agrawal, A. Pratab, and T. Merayivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, April 2002.

[7] E. Zitzeler, M. Laumanns and L. Thiele, "SPEA-2: Improving the strength Pareto evolutionary algorithm," Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Switzerland, TIK-Report, no. 103, 2001.

[8] Q. Zhang, and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," IEEE Transaction on Evolutionary Computation, vol. 11, no. 6, pp. 712-731, December 2007.

[9] H. Li, and Q. Zhang, "Multiobjective optimization problems with complicate Pareto sets, MOEA/D and NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 13, no.2, pp. 284-302, April 2009.

[10] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," IEEE congress on Evolutionary Computation, Trondheim Norway, May 2009.

[11] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School of Computer Science and Electronic Engineering, University of Essex, Technical-Report CES-487, April 2009.

[12] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Evolutionary computation, vol. 8, no. 2, pp. 173-195, Summer 2000.

[13] D. A. Van Veldhuizen, and G. B. Lamont, "Multiobjective Evolutionary algorithm research: A history and analysis," Department of Electrical and Computer Engineering, Air Force Institute of Technology, Technical-Report TR-98-03, October 1998.

[14] S. Zapotecas Martínez, and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland, July 12-16, 2011.

[15] K. Miettinen, Nonlinear Multiobjective Optimization. Norwell, MA: Kluwer, 1999.

[16] D. Karaboga, "An idea based on a honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Computer Engineering Department, Technical-Report TR06, October 2005.

[17] B. Akay, and D. Karaboga, "A modified artificial bee colony for Real-parameters optimization," Information Sciences, vol. 192, no. 1, pp. 120-142, June 2012.

[18] R. V. Rao, V. J. Savsani, and D. P. Vakhaira, "Teaching-learning based optimization: a novel method for constrained mechanical design optimization problems," Computer-Aided Design, vol. 43, no. 3, pp. 303-315, March 2011.