

A New Multi-Objective Particle Swarm Optimizer with Improved Selection and Diversity Mechanisms

Margarita Reyes Sierra and Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group)
Electrical Eng. Department, Computer Science Dept.
Av. IPN No. 2508, Col. San Pedro Zacatenco,
México D.F. 07300, MÉXICO
mreyes@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

November 23, 2004

Abstract

This paper introduces a new Multi-Objective Particle Swarm Optimizer, which uses the concept of Pareto dominance and a crowding factor to choose the leaders that will be included in a (limited) list available at each generation. We also propose the use of a single swarm but subdivided into three different subsets in which a different mutation (or *turbulence*) operator is used. Finally, the proposed approach also adopts the concept of ϵ -dominance to select the particles obtained as the final solutions of the algorithm. Our approach is validated using several standard test functions taken from the specialized literature and is compared against five state-of-the-art algorithms, including three recently proposed PSO-based approaches. The results indicate that the approach proposed in this paper is highly competitive, being able to approximate the true Pareto front even in cases where all the other PSO-based approaches fail.

1 Introduction

Due to the multicriteria nature of most real-world problems, multi-objective optimization problems are very common, particularly in engineering applications. As the name indicates, multi-objective optimization problems involve multiple objectives, which should be optimized simultaneously and that often are in conflict with each other. This results in a group of alternative solutions which must be considered equivalent in the absence of information concerning the relevance of the others.

Since Evolutionary Algorithms (EAs) deal with a group of candidate solutions, it seems natural to use them in multi-objective optimization problems to find a group of

optimal solutions. Indeed, EAs have proved very efficient in solving multi-objective optimization problems [7, 8].

Another population-based optimization algorithm, particle swarm optimization, has proved to be an efficient optimization method for single objective optimization [17], and more recently has also shown promising results for solving multi-objective problems [6, 21].

In this paper, we present a new proposal for using particle swarm optimization to solve multiobjective optimization problems. Our approach is based on Pareto dominance and the use of a crowding factor for the selection of leaders. We also incorporate a scheme by which the swarm is subdivided in three portions of the same size. A different mutation operator (taken from the evolutionary algorithms literature) is applied to each of these portions. However, for all other purposes, a single swarm (or population) is considered. Finally, we use the concept of ϵ -dominance to select the particles that remain in the external archive.

The remainder of this paper is organized as follows. A brief introduction to the particle swarm optimization strategy is given in Section 2. In Section 3, we provide the definition of the problem that we want to solve. The previous related work is reviewed in Section 4. In Section 5, we describe our proposed approach. The obtained results and their corresponding discussion are presented in Sections 6 and 7, respectively. Finally, the conclusions and future work are described in Section 8.

2 Particle Swarm Optimization

Kennedy and Eberhart [16] initially proposed the swarm strategy for optimization. The particle swarm optimization (PSO) algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. In PSO, individuals, referred to as particles, are “flown” through hyperdimensional search space. Changes to the position of the particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals.

A swarm consists of a set of particles, where each particle represents a potential solution. The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{x}_i(t)$ denote the position of particle p_i , at time step t . The position of p_i is then changed by adding a velocity $\vec{v}_i(t)$ to the current position, i.e.:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \quad (1)$$

The velocity vector drives the optimization process and reflects the socially exchanged information. In general, based on the social information that is exchanged at each iteration, there are three different versions of PSO algorithm:

- **Individual best.** Each particle compares its current position to its own best position (*pbest*), only.
- **Global best.** Each particle uses its history of experiences in terms of its own best solution thus far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*).

```

Begin
  Initialize swarm
   $g = 0$ 
  While  $g < gmax$ 
    For each particle
      Update Position (Flight)
      Evaluation
      Update  $pbest$ 
    EndFor
     $g++$ 
  EndWhile
End

```

Figure 1: Pseudocode of the general PSO algorithm.

- **Local best.** Particles are influenced by the best position within their neighborhood ($lbest$), as well as their own past experience ($pbest$).

In the global best version (used here) of PSO, the velocity vector changes in the following way:

$$\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2r_2(\vec{x}_{gbest} - \vec{x}_i(t)) \quad (2)$$

where W is the inertia weight, C_1 and C_2 are the learning factors (usually defined as constants), and $r_1, r_2 \in [0, 1]$ are random values.

Figure 1 shows the way in which the general PSO algorithm works. First, the swarm is initialized: positions and velocities. The corresponding $pbest$ of each particle is initialized and the $gbest$ is located. Then, for a maximum number of iterations, each particle flies through the search space updating its position (using (1) and (2)) and its $pbest$ and, finally, the $gbest$ solution is updated too.

The successful application of PSO in many single objective optimization problems reflects the effectiveness of PSO [17]. However, PSO must be modified in order to apply it to multi-objective problems. In most approaches (which will be generically called MOPSOs, for Multiple-Objective Particle Swarm Optimizers), the major modifications of the PSO algorithm are the selection process of $pbest$ and $gbest$ [6, 25, 38, 2].

3 Statement of the Problem

We are interested in solving problems of the type:

$$\text{minimize } [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})] \quad (3)$$

subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \dots, m \quad (4)$$

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \dots, p \quad (5)$$

where k is the number of objective functions $f_i : R^n \rightarrow R$. We call $\vec{x} = [x_1, x_2, \dots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set \mathcal{F} of all the vectors that satisfy (4) and (5) to the vectors $x_1^*, x_2^*, \dots, x_n^*$ that are *Pareto optimal*.

We say that a vector of decision variables $\vec{x}^* \in \mathcal{F}$ is *Pareto optimum* if there does not exist another $\vec{x} \in \mathcal{F}$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ for every $i = 1, \dots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one j .

The vectors \vec{x}^* corresponding to the solutions included in the Pareto optimal set are called *nondominated*. The objective function values corresponding to the elements of the Pareto optimal set are called the *Pareto front* of the problem.

4 Related Work

There have been several proposals to extend PSO to handle multiple objectives. We will review next the most representative of them:

- Moore and Chapman [23]: This algorithm was presented in an unpublished document and it is based on Pareto dominance. The authors emphasize the importance of performing both an individual and a group search (a cognitive component and a social component). However, the authors did not adopt any scheme to maintain diversity.
- Ray and Liew [30]: This algorithm uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. The approach uses crowding to maintain diversity (by means of a roulette selection scheme of leaders based on this value) and a multilevel sieve to handle constraints (for this, the authors adopt the constraint and objective matrices proposed in some of their previous research [29]).
- Parsopoulos and Vrahatis [28]: Unlike the previous proposals, this algorithm adopts an aggregating function (three types of approaches were implemented: a conventional linear aggregating function, a dynamic aggregating function and the bang bang weighted aggregation approach [15]). In more recent work, Parsopoulos et al. [27] studied a parallel version of the Vector Evaluated Particle Swarm (VEPSO) method for multiobjective problems. VEPSO is a multi-swarm variant of PSO, which is inspired on the Vector Evaluated Genetic Algorithm (VEGA) [31]. In VEPSO, each swarm is evaluated using only one of the objective functions of the problem under consideration, and the information it possesses for this objective function is communicated to the other swarms through the exchange of their best experience.
- Hu and Eberhart [13]: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering [7]. Lexicographic ordering tends to be useful only when few objective functions are used (two or three), and it may be sensitive to the ordering of the objectives. In further work, Hu et al.

[14] adopted a secondary population (called “extended memory”) and introduced some further improvements to their dynamic neighborhood PSO approach.

- Fieldsend and Singh [11]: This approach uses an unconstrained elite archive (in which a special data structure called “dominated tree” is adopted) to store the nondominated individuals found along the search process. The archive interacts with the primary population in order to define local guides. This approach also uses a “turbulence” operator that is basically a mutation operator that acts on the velocity value used by PSO.
- Coello et al. [5, 6]: This proposal is based on the idea of having a global repository in which every particle will deposit its flight experiences after each flight cycle. Additionally, the updates to the repository are performed considering a geographically- based system defined in terms of the objective function values of each individual; this repository is used by the particles to identify a leader that will guide the search. It also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved. In more recent work, Toscano and Coello [34] use the concept of Pareto dominance to determine the flight direction of a particle. The authors adopt clustering techniques in order to divide the population of particles into several swarms in order to have a better distribution of solutions in decision variable space. In each sub-swarm, a PSO algorithm is executed and, at some point, the different sub-swarms exchange information: the leaders of each swarm are migrated to a different swarms in order to variate the selection pressure. Also, this approach does not use an external population since elitism in this case is an emergent process derived from the migration of leaders.
- Mostaghim and Teich [25]: They proposed a sigma method in which the best local guides for each particle are adopted to improve the convergence and diversity of a PSO approach used for multiobjective optimization. They also use a “turbulence” operator, but applied on decision variable space. The idea of the sigma method is similar to compromise programming. The use of the sigma values increases the selection pressure of PSO (which was already high). This may cause premature convergence in some cases. In further work, Mostaghim and Teich [24] studied the influence of ϵ -dominance [19] on MOPSO methods. ϵ -dominance is compared with existing clustering techniques for fixing the archive size and the solutions are compared in terms of computational time, convergence and diversity. The results show that the ϵ -dominance method can find solutions much faster than the clustering technique with a comparable (and even better in some cases) convergence and diversity. The authors suggest a new diversity measure (sigma method) inspired on their previous work [25]. Also, based on the idea that the initial archive from which the particles have to select a local guide has influence on the diversity of solutions, the authors propose the use of successive improvements using a previous archive of solutions. In more recent work, Mostaghim and Teich [26] propose a new method called *covering*MOPSO (cvMOPSO). This method works in two phases. In phase 1, a MOPSO algorithm is run with a restricted archive size and the goal is to obtain a good approximation

of the Pareto-front. In the phase 2, the non-dominated solutions obtained from the phase 1 are considered as the input archive of the cvMOPSO. The particles in the population of the cvMOPSO are divided into subswarms around each non-dominated solution after the first generation. The task of the subswarms is to cover the gaps between the non-dominated solutions obtained from the phase 1. No restrictions on the archive are made in the phase 2.

- Li [20]: This approach incorporates the main mechanisms of the NSGA-II [9] to the PSO algorithm. It combines the population of particles and all the personal best positions of each particle, and selects the best particles among them to conform the next population. It also selects the leaders randomly from the leaders set among the best of them, based on two different mechanisms: using a niche count and using a crowding distance. In more recent work, Li [21] proposes the *maximinPSO*, which uses a fitness function derived from the maximin strategy [1] to determine Pareto-domination. The author shows that one advantage of this approach is that no additional clustering or niching technique is needed, since the maximin fitness of a solution can tell us not only if a solution is dominated or not, but also if it is clustered with other solutions, i.e., the approach also provides diversity information.
- Srinivasan and Hou [32]: This approach, called Particle Swarm Inspired Evolutionary Algorithm (PS-EA), is a hybrid between PSO and an evolutionary algorithm. The main aim is to use EA operators to emulate the workings of PSO mechanisms.
- Zhang et al. [38]: This approach attempts to improve the selection of g_{best} and p_{best} when the velocity of each particle is updated. For each objective function, there exists both a g_{best} and a p_{best} for each particle. In order to update the velocity of a particle, the algorithm defines the g_{best} of a particle as the average of the complete set of g_{best} particles. Analogously, the p_{best} is computed using either a random choice or the average from the complete set of p_{best} values. This choice depends on the dispersion degree between the g_{best} and p_{best} values of each particle.
- Bartz et al. [2]: This approach starts from the idea of introducing elitism (archiving) into PSO. Different methods for selecting and deleting particles from the archive are analyzed to generate a satisfactory approximation of the Pareto front. The selection methods analyzed are based on the contribution of each particle to the diversity of the Pareto front. Deleting methods are either inversely related to the selection fitness or based on the previous success of each particle. The authors provide some statistical analysis in order to assess the impact of each of the parameters used by their approach.
- Baumgartner et al. [3]: This approach uses weighted sums (i.e., linear aggregating functions). In this approach, the swarm is equally partitioned into n subswarms, each of which uses a different set of weights and evolves into the direction of its own swarm leader. The approach adopts a gradient technique to identify the Pareto optimal solutions.

- Chow and Tsui [4]: In this paper, a novel autonomous agent response learning algorithm is presented. The authors propose to decompose the award function into a set of local award functions and, in this way, to model the response extraction process as a multiobjective optimization problem. A modified PSO called “Multi-Species PSO” is introduced by considering each objective function as a species swarm. A communication channel is established between the neighboring swarms for transmitting the information of the best particles, in order to provide guidance for improving their objective values. Also, the authors propose to modify the equation used to update the velocity of each particle, considering also the global best particle of its neighboring species.

Our approach is based on Pareto dominance and the use of a crowding factor for the selection of leaders. It is worth noticing that we also use the crowding factor to filter out the list of leaders whenever the maximum limit imposed on such list is exceeded. Additionally, we also use mutation. However, some of our preliminary studies in this regard have indicated that it is considerably difficult to balance the adequate amount of mutation required when using PSO for multiobjective optimization. This led us to propose a scheme in which we subdivide the population (or swarm) into three different subsets. A different mutation operator is applied to each subset (each of which contains one third of the total number of particles in the population). Note however, that for all other purposes, a single swarm is considered (e.g., for selecting leaders). Finally, we also incorporate the concept of ϵ -dominance to select the particles that will remain in the external archive. The motivation for incorporating this mechanism is the previous evidence of its effectiveness when used with other evolutionary algorithms [19].

5 Description of Our Approach

It should be obvious that the main issue when extending PSO to deal with multiple objectives is how to generalize the concept of leader in the presence of several (equally good) solutions. The most straightforward approach is simply to consider every non dominated solution as a new leader. This approach has, however, the drawback of increasing the size of the set of leaders very quickly. For example, based on the results reported by Li [21], we can conclude that almost a 70% of the particles in the swarm enter into the set of leaders at each generation. This increase on the size of the leaders set is very important because such set has to be updated at each generation, and this can become a very expensive process. Also, the size of the set of leaders has a significant impact on the selection of a leader in order to make a movement through the search space at each generation. If the set of leaders becomes too large, the election of a good leader turns out to be difficult (how do we discriminate from among many non dominated individuals?). This particular issue has not been addressed in most of the research conducted in this area and is the main focus of this work.

In our approach, we use a crowding factor [9] in order to establish a second discrimination criterion (additional to Pareto dominance). This criterion is also adopted to decide what leaders to keep over generations when the maximum list size has been exceeded. For each particle, we select the leader by means of a binary tournament

based on the crowding value of the leaders. The maximum size of the set of leaders is fixed equal to the size of the swarm or population (this value is provided by the user as a parameter). After each generation, the set of leaders is updated, and so are the corresponding crowding values. If the size of the set of leaders is greater than the maximum allowable size, only the best leaders are retained based on their crowding value. The rest of the leaders are eliminated. Although there are previous approaches that use the crowding factor to select the leaders (see for example [30, 20]), our approach is the first to adopt this information to fix the size of the set of leaders. This feature of our algorithm considerably simplifies the mechanism to control the set of leaders without requiring any additional parameter or selection criterion (e.g., in [20], a parameter is required to define the portion of the list of leaders to be retained). Additionally, the use of this selection criterion promotes diversity within the swarm.

When adopting PSO for solving multi-objective optimization problems, the use of a mutation operator is very important in order to escape from local optima and to improve the exploratory capabilities of PSO which, in some cases, becomes severely limited (see for example [6]). Several researchers have used mutation operators before. For example, Fieldsend & Singh [11] adopt a mutation operator that modifies the velocity of a particle by adding it a random factor (with a normal distribution) that allows a particle to move within a 10% distance from its original position (defined in terms of the values of the decision variables). Coello & Lechuga [5] use a mutation operator that acts on the position (decision variables) of the updated particles of the swarm. This mutation operator modifies the range of mobility of the particles: at the beginning of the process, the particles are allowed to move through the complete range of the decision variables, and this mobility is reduced as the evolutionary process progresses. Mostaghim & Teich [25] use a mutation operator that (as in [5]) acts on the updated position of the particles. In this case, the range of mobility of the particles depends on the actual position of the particle. It is worth noticing that all of the previous proposals require of some user-defined parameter.

From our perspective, the election of a good mutation operator is a difficult task that has a significant impact on performance. Thus, in this work we propose the use of two mutation operators that are well-known in the EA literature: uniform mutation (i.e., the variability range of each decision variable is kept constant over generations) [12] and non-uniform mutation (i.e., the variability range of each decision variable decreases over time) [22]. Both operators act on the decision variables of the updated particle. These operators modify the values of the decision variables of a particle with a certain probability. This makes a significant difference with respect to the previous proposals in which all the decision variables are modified when the turbulence (or mutation) operator is applied. Additionally, we considered the possibility of not using mutation at all, since in some of our previous research we found that such condition may be beneficial in some cases [6].

Given the uncertainty regarding how much mutation to apply, we propose a scheme by which the swarm is subdivided in three parts (of equal size). Each sub-part of the swarm will adopt a different mutation scheme: the first sub-part will have no mutation at all, the second sub-part will have uniform mutation and the third sub-part will have non-uniform mutation. This process is illustrated in the Figure 2. With the use of these different operators we are aiming to have the ability of exploring (uniform mutation)

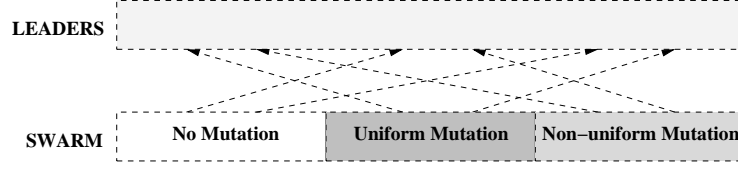


Figure 2: Graphical illustration of the subdivision of the swarm adopted by our scheme.

and exploiting (non-uniform mutation) the search space as the process progresses. The available set of leaders is the same for each of these sub-parts. Additionally, each particle can use as a leader a particle produced by a different sub-part of the swarm. In this way, the three different sub-parts of the swarm will share their particular success and the final results will be a combination of using different behaviors inside the same swarm. In order to avoid the definition of extra parameters for the mutation operators, we adopt a rule of thumb commonly used in the EA literature: the mutation rate is defined as $1/\text{codesize}$, where codesize refers to the total length of the string that encodes all the decision variables of the problem. Since real-numbers encoding is adopted, in our case codesize is equal to the number of decision variables of each problem.

Finally, we adopt the concept of ϵ -dominance [19] in order to fix the size of the external archive that contains the (non-dominated) solutions that will be reported by the algorithm. A decision vector x_1 is said to ϵ -dominate a decision vector x_2 for some $\epsilon > 0$ iff: $f_i(x_1)/(1 + \epsilon) \leq f_i(x_2), \forall i = 1, \dots, m$ and $f_i(x_1)/(1 + \epsilon) < f_i(x_2)$, for at least one $i = 1, \dots, m$.

It is worth noting that, when using ϵ -dominance, the size of the final external archive depends on the ϵ -value, which is normally a user-defined parameter [19]. For the sake of simplicity, in this paper, we consider the same value of ϵ for all the objective functions.

Figure 3 shows the way in which our algorithm works. We have marked with *italics* the processes that make this algorithm different from the general PSO algorithm. First, we initialize the swarm. The non-dominated particles found in the swarm will be introduced into the set of leaders. Later on, the crowding factor of each leader is calculated. At each generation, for each particle, we perform the flight and apply the corresponding mutation operator based on the subdivision of the swarm previously described. In order to perform the flight of each particle, the changes to the velocity vector are done in the following way:

$$\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2r_2(\vec{x}_{gbest} - \vec{x}_i(t))$$

where $W = \text{random}(0.1, 0.5)$, $C_1, C_2 = \text{random}(1.5, 2.0)$, and $r_1, r_2 = \text{random}(0.0, 1.0)$. Note that most of the previous PSO proposals fix the values of W, C_1 and C_2 instead of using random values as in our case. The only exception that we know (in the specific case of MOPSOs) is some of our own previous work [34]. We adopted this scheme since we found it as a more convenient way of dealing with the difficulties of fine tuning the parameters W, C_1 and C_2 for each specific test function.

```

Begin
  Initialize swarm
  Initialize leaders
  Send leaders to  $\epsilon$ -archive
  crowding(leaders)
   $g = 0$ 
  While  $g < gmax$ 
    For each particle
      Select leader
      Flight
      Mutation
      Evaluation
      Update  $p_{best}$ 
    EndFor
    Update leaders
    Send leaders to  $\epsilon$ -archive
    crowding(leaders)
     $g++$ 
  EndWhile
  Report results in  $\epsilon$ -archive
End

```

Figure 3: Pseudocode of our algorithm.

Then, we proceed to evaluate the particle and update its personal best value (p_{best}). A new particle replaces its p_{best} value if such value is dominated by the new particle or if both are incomparable (i.e., they are both non-dominated with respect to each other). After all the particles have been updated, the set of leaders is updated, too. Obviously, only the particles that outperform their p_{best} value will try to enter the leaders set. Once the leaders set has been updated, the ϵ -archive is updated. Finally, we proceed to update the crowding values of the set of leaders and we eliminate as many leaders as necessary in order avoid exceeding the allowable size of the leaders set. This process is repeated a fixed number ($gmax$) of iterations.

The parameters needed by our approach are:

1. $swarmsize$: size of the swarm (defined by the user).
2. $gmax$: number of iterations (defined by the user).
3. pm : bit mutation probability (fixed by the algorithm).
4. ϵ : value for the bounding the size of the ϵ -archive (defined by the user).

6 Comparison of Results

To validate our approach, we performed both quantitative (adopting four performance measures) and qualitative comparisons (plotting the Pareto fronts produced) with respect to two MOEAs that are representative of the state-of-the-art in the area: the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [41], and the Nondominated Sorting Genetic Algorithm II (NSGA-II) [9]. We also compared our approach against three PSO-based approaches recently proposed: MOPSO [7], Sigma-MOPSO [25] and Cluster-MOPSO [34]. For our comparative study, we implemented two unary and two binary measures of performance. The following are the unary measures:

- **Success Counting (SCC):** We define this measure based on the idea of the measure called *Error Ratio* proposed by Van Veldhuizen [35] which indicates the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set. In this case, we count the number of vectors (in the current set of nondominated vectors available) that are members of the Pareto optimal set:

$$SCC = \sum_{i=1}^n s_i,$$

where n is the number of vectors in the current set of nondominated vectors available; $s_i = 1$ if vector i is a member of the Pareto optimal set, and $s_i = 0$ otherwise. It should then be clear that $SCC = n$ indicates an ideal behavior, since it would mean that all the vectors generated by our algorithm belong to the true Pareto optimal set of the problem. For a fair comparison, when we use this measure, all the algorithms should limit their final number of non-dominated solutions to the same value. Note that SCC avoids the bias introduced by the Error Ratio measure, which normalizes the number of solutions found (which belong to the true Pareto front) and, therefore, provides only a percentage of solutions that reached the true Pareto front. This percentage does not provide any idea regarding the actual number of non-dominated solutions that each algorithm produced.

- **Inverted Generational Distance (IGD):** The concept of generational distance was introduced by Van Veldhuizen & Lamont [36, 37] as a way of estimating how far are the elements in the Pareto front produced by our algorithm from those in the true Pareto front of the problem. This measure is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^n d_i^2}}{n} \quad (6)$$

where n is the number of nondominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the true Pareto front. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem. Therefore, any other value will indicate

how “far” we are from the global Pareto front of our problem. In our case, we implemented an “inverted” generational distance measure (IGD) in which we use as a reference the true Pareto front, and we compare each of its elements with respect to the front produced by an algorithm. In this way, we are calculating how far are the elements of the true Pareto front, from those in the Pareto front produced by our algorithm. Computing this “inverted” generational distance value reduces the bias that can arise when an algorithm didn’t fully cover the true Pareto front.

The binary measures adopted are the following:

- **Two Set Coverage (SC):** This measure was proposed in [42], and it can be termed *relative coverage comparison of two sets*. Consider X', X'' as two sets of phenotype decision vectors. SC is defined as the mapping of the order pair (X', X'') to the interval $[0, 1]$:

$$SC(X', X'') \triangleq |\{a'' \in X''; \exists a' \in X' : a' \preceq a''\}| / |X''|$$

If all points in X' dominate or are equal to all points in X'' , then by definition $SC = 1$. $SC = 0$ implies the opposite. In general, $SC(X', X'')$ and $SC(X'', X')$ both have to be considered due to set intersections not being empty. If $SC(X', X'') = 0$ and $SC(X'', X') = 1$, we say that X'' is better than X' .

- **Two Set Difference Hypervolume (HV):** This measure was proposed in [39]. Consider X', X'' as two sets of phenotype decision vectors. HV is defined by:

$$HV(X', X'') = \delta(X' + X'') - \delta(X'')$$

where the set $X' + X''$ is defined as the nondominated vectors obtained from the union of X' and X'' , and δ is the unary hypervolume measure. $\delta(X)$ is defined as the hypervolume of the portion of the objective space that is dominated by X . In this way, $HV(X', X'')$ gives the hypervolume of the portion of the objective space that is dominated by X' but not for X'' .

In this paper, we use the origin as a reference point to compute the hypervolume. So, since all the test functions have to be minimized, with this measure we obtain a difference between the areas that *dominate* the analyzed Pareto fronts. In this way, if $HV(X', X'') = 0$ and $HV(X'', X') < 0$, we say that X'' is better than X' .

For each of the test functions shown below, we performed 20 runs per algorithm. The parameters of each approach were set such that they all performed 20000 objective function evaluations. The codes of NSGA-II and SPEA2 were obtained from PISA.¹ The code of MOPSO was obtained from the EMOO repository.² The codes of

¹<http://www.tik.ee.ethz.ch/pisa/>

²<http://delta.cs.cinvestav.mx/~ccoello/EMOO>

Sigma-MOPSO and Cluster-MOPSO were provided by their authors. The code of the approach proposed in this paper is available via email request to the first author.

The test functions adopted in our comparative study were the following:

- Test Function ZDT1 [40]:

$$\begin{aligned}
 &\text{Minimize} && (f_1(\vec{x}), f_2(\vec{x})) && (7) \\
 &f_1(\vec{x}) &= & x_1 \\
 &f_2(\vec{x}) &= & g(\vec{x})h(f_1, g) \\
 &g(\vec{x}) = 1 + 9 \sum_{i=2}^m x_i / (m-1), h(f_1, g) = 1 - \sqrt{f_1/g}
 \end{aligned}$$

where $m = 30$, and $x_i \in [0,1]$.

- Test Function ZDT2 [40]:

$$\begin{aligned}
 &\text{Minimize} && (f_1(\vec{x}), f_2(\vec{x})) && (8) \\
 &f_1(\vec{x}) &= & x_1 \\
 &f_2(\vec{x}) &= & g(\vec{x})h(f_1, g) \\
 &g(\vec{x}) = 1 + 9 \sum_{i=2}^m x_i / (m-1), h(f_1, g) = 1 - (f_1/g)^2
 \end{aligned}$$

where $m = 30$, and $x_i \in [0,1]$.

- Test Function ZDT3 [40]:

$$\begin{aligned}
 &\text{Minimize} && (f_1(\vec{x}), f_2(\vec{x})) && (9) \\
 &f_1(\vec{x}) &= & x_1 \\
 &f_2(\vec{x}) &= & g(\vec{x})h(f_1, g) \\
 &g(\vec{x}) = 1 + 9 \sum_{i=2}^m x_i / (m-1) \quad , \quad h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g) \sin(10\pi f_1)
 \end{aligned}$$

where $m = 30$, and $x_i \in [0,1]$.

- Test Function ZDT4 [40]:

$$\begin{aligned}
& \text{Minimize} && (f_1(\vec{x}), f_2(\vec{x})) && (10) \\
& f_1(\vec{x}) &= & x_1 \\
& f_2(\vec{x}) &= & g(\vec{x})h(f_1, g) \\
& g(\vec{x}) = 1 + 10(m-1) + \sum_{i=2}^m (x_i^2 - 10\cos(4\pi x_i)), h(f_1, g) = 1 - \sqrt{f_1/g}
\end{aligned}$$

where $m = 10$, $x_1 \in [0,1]$ and $x_i \in [-5,5]$, $i = 2, \dots, m$.

- Test Function DTLZ2 [10]:

$$\begin{aligned}
& \text{Minimize} && (f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})) && (11) \\
& f_1(\vec{x}) &= & (1 + g(\vec{x}))\cos(x_1\pi/2)\cos(x_2\pi/2) \\
& f_2(\vec{x}) &= & (1 + g(\vec{x}))\cos(x_1\pi/2)\sin(x_2\pi/2) \\
& f_3(\vec{x}) &= & (1 + g(\vec{x}))\sin(x_1\pi/2) \\
& g(\vec{x}) &= & \sum_{i=3}^m (x_i - 0.5)^2
\end{aligned}$$

where $m = 12$ and $x_i \in [0,1]$.

- Test Function DTLZ4 [10]:

$$\begin{aligned}
& \text{Minimize} && (f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})) && (12) \\
& f_1(\vec{x}) &= & (1 + g(\vec{x}))\cos(x_1^\alpha\pi/2)\cos(x_2^\alpha\pi/2) \\
& f_2(\vec{x}) &= & (1 + g(\vec{x}))\cos(x_1^\alpha\pi/2)\sin(x_2^\alpha\pi/2) \\
& f_3(\vec{x}) &= & (1 + g(\vec{x}))\sin(x_1^\alpha\pi/2) \\
& g(\vec{x}) &= & \sum_{i=3}^m (x_i - 0.5)^2
\end{aligned}$$

where $\alpha=100$, $m = 12$ and $x_i \in [0,1]$.

- Test Function DTLZ6 [10]:

$$\begin{aligned}
& \text{Minimize} && (f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})) && (13) \\
& f_1(\vec{x}) &= & x_1 \\
& f_2(\vec{x}) &= & x_2 \\
& f_3(\vec{x}) &= & (1 + g(\vec{x}))h(f_1, f_2, g) \\
& g(\vec{x}) = 1 + 9/(m-2) \sum_{i=3}^m x_i, \quad h(f_1, f_2, g) = 3 - \sum_{i=1}^2 \left[\frac{f_i}{1+g} (1 + \sin(3\pi f_i)) \right]
\end{aligned}$$

where $m = 22$ and $x_i \in [0,1]$.

All the algorithms compared adopted real-numbers encoding. The parameters for SPEA2 were $\alpha = \mu = \lambda = 100$ and 200 generations, and for the NSGA-II we used $popsiz=100$ and 200 generations. As recommended in [10], for the NSGA-II, the crossover probability p_c was set to 1.0 and the mutation probability p_m was set to $1/codesize$. For our proposed approach and the MOPSO algorithm the parameters were: swarm size of 100 particles and 200 iterations. Cluster-MOPSO used 40 particles, 4 swarms, 5 iterations per swarm and a total number of iterations of 100. In the case of Sigma-MOPSO, 200 particles were used through 100 iterations (this was done because the author suggested to use larger population sizes with her approach). As recommended in [25], the Sigma-MOPSO used a turbulence probability of 0.01 for all functions, except for ZDT4 in which the turbulence probability used was 0.05. As recommended by its authors, the MOPSO used a mutation probability of 0.5. Our proposed approach used a probability mutation of $1/codesize$. For each of the examples included, we empirically defined a suitable value for ϵ by doing some preliminary runs in which the aim was to obtain 100 solutions in the external archive. The PSO approaches will be identified with the following labels: MOPSO refers to the approach reported in [5, 6], sMOPSO refers to the approach reported in [25], cMOPSO refers to the approach reported in [34], and OMOPSO (Our Multi-Objective Particle Swarm Optimizer) refers to the approach reported in this paper.

The Pareto fronts that we will show correspond to the nondominated vectors obtained from the union of the 20 obtained Pareto fronts. It should be noted that the Pareto fronts shown were also used to apply the binary measures of performance. All the algorithms, except for the Cluster-MOPSO, were set such that they provided Pareto fronts with 100 points. The Cluster-MOPSO does not have a scheme to fix the size of its final archive. Thus, in order to allow a fair comparison with respect to the Cluster-MOPSO, the values of the SCC measures were scaled to the interval $[0,100]$.

From Table 1 to Table 14 we show the values of the performance measures obtained for each of the algorithms compared.

Test Function ZDT1							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	82	38	47	0	93	37
	median	43	20	26	0	58	7
	worst	5	9	15	0	23	1
	average	40	21	27	0	59	8
	std. dev.	21.5	7.5	8.1	0	24.2	7.9
<i>IGD</i>	best	0.0009	0.0008	0.0006	0.0240	0.0031	0.0016
	median	0.0010	0.0008	0.0007	0.0276	0.0260	0.0029
	worst	0.0013	0.0011	0.0008	0.0385	0.0448	0.0041
	average	0.0010	0.0009	0.0007	0.0286	0.0269	0.0030
	std. dev.	0.00008	0.00009	0.00006	0.0040	0.0095	0.0007

Table 1: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function ZDT1 with respect to the unary measures.

Test Function ZDT1 - Two Set Coverage Measure <i>SC</i>						
<i>SC</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.64	0.55	0.96	0.03	0.95
NSGA-II	0.05	0.00	0.22	1.00	0.01	0.99
SPEA2	0.11	0.49	0.00	1.00	0.01	1.00
MOPSO	0.00	0.00	0.00	0.00	0.00	0.00
sMOPSO	0.50	0.69	0.66	0.99	0.00	0.90
cMOPSO	0.00	0.01	0.00	1.00	0.00	0.00
Test Function ZDT1 - Two Set Hypervolume Measure <i>HV</i>						
<i>HV</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.001834	-0.001339	-0.323693	0.006828	-0.019389
NSGA-II	0.001304	0.000000	-0.000037	-0.322274	0.007647	-0.016607
SPEA2	0.001302	-0.000534	0.000000	-0.322771	0.007733	-0.017104
MOPSO	0.001719	0.000000	0.000000	0.000000	0.000000	0.000000
sMOPSO	-0.000922	-0.003241	-0.002658	-0.333162	0.000000	-0.020032
cMOPSO	0.000356	0.000000	0.000000	-0.305667	0.007463	0.000000

Table 2: Comparison of results using the binary measures for test function ZDT1. Our algorithm is denoted by OMOPSO.

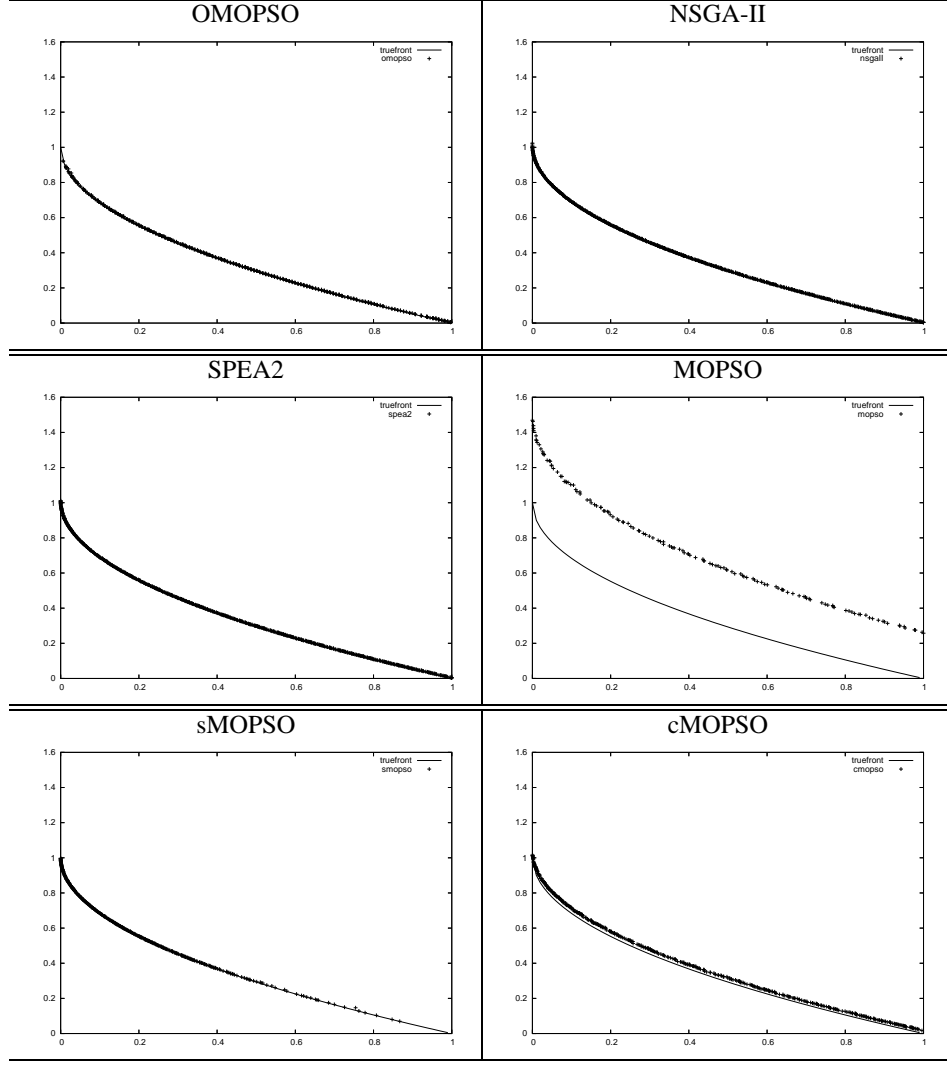


Figure 4: Pareto fronts obtained by all the approaches for test function ZDT1. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.0075$.

Test Function ZDT2							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	99	30	34	0	1	94
	median	49	0	0	0	1	0
	worst	0	0	0	0	1	0
	average	43	6	7	0	1	29
	std. dev.	34.1	9.8	10.4	0	0	38.9
<i>IGD</i>	best	0.0006	0.0008	0.0007	0.0271	0.0723	0.0030
	median	0.0009	0.0724	0.0723	0.1098	0.0723	0.0723
	worst	0.0303	0.0737	0.0736	0.3525	0.0723	0.0852
	average	0.0034	0.0512	0.0404	0.1561	0.0723	0.0680
	std. dev.	0.0078	0.0337	0.0367	0.0952	0.0000	0.0152

Table 3: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function ZDT2 with respect to the unary measures.

Test Function ZDT2 - Two Set Coverage Measure <i>SC</i>						
<i>SC</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.93	0.94	1.00	0.00	0.21
NSGA-II	0.01	0.00	0.34	1.00	0.00	0.21
SPEA2	0.01	0.21	0.00	1.00	0.00	0.21
MOPSO	0.00	0.00	0.00	0.00	0.00	0.00
sMOPSO	0.01	0.01	0.01	0.44	0.00	0.00
cMOPSO	0.01	0.02	0.02	0.99	0.00	0.00
Test Function ZDT2 - Two Set Hypervolume Measure <i>HV</i>						
<i>HV</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.004947	-0.005765	-0.342087	-0.666684*	-0.036710
NSGA-II	0.000547	0.000000	-0.000493	-0.336593	-0.672178*	-0.031559
SPEA2	0.000708	0.000486	0.000000	-0.335614	-0.673157*	-0.030560
MOPSO	0.000000	0.000000	0.000000	0.000000	-0.897843*	0.000000
sMOPSO	0.000000	0.000000	0.000000	-0.110928	0.000000	0.000000
cMOPSO	0.000126	-0.000217	-0.000197	-0.305251	-0.703520*	0.000000

Table 4: Comparison of results using the binary measures for test function ZDT2. Our algorithm is denoted by OMOPSO.

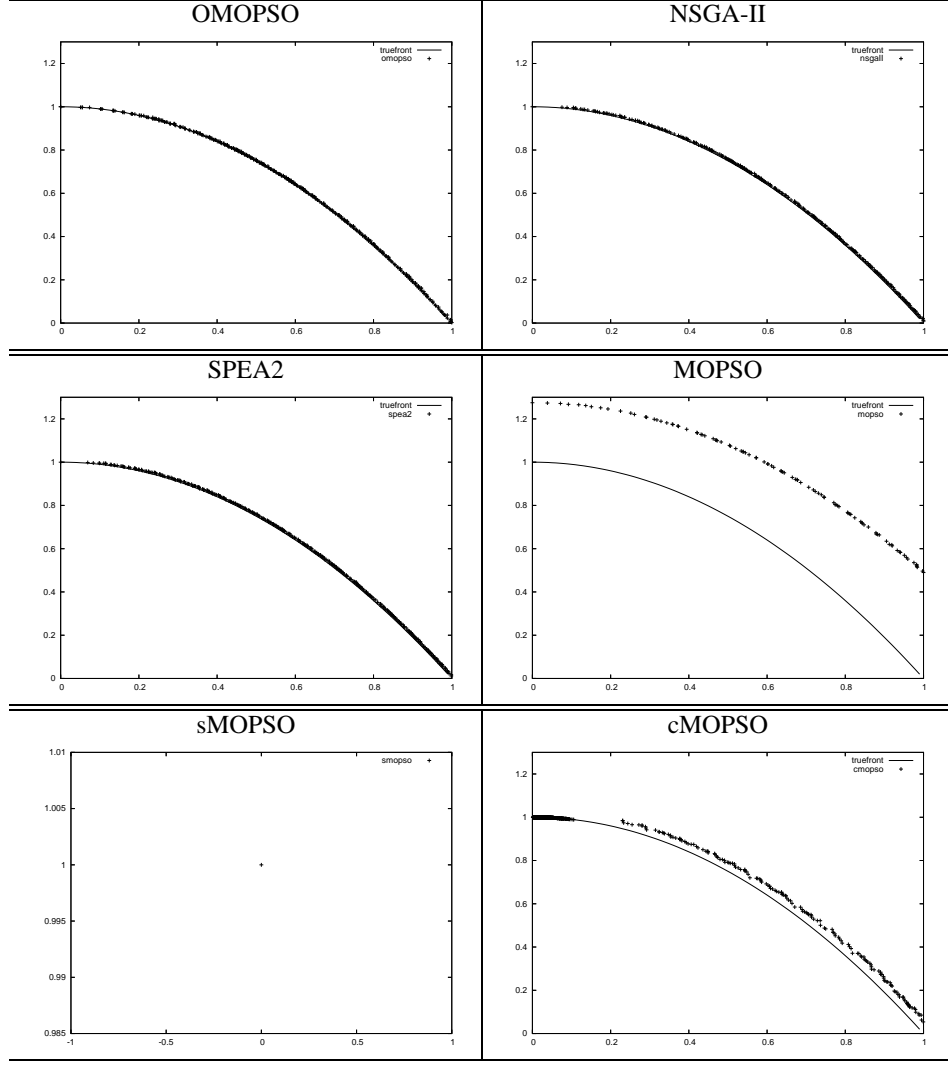


Figure 5: Pareto fronts obtained by all the approaches for test function ZDT2. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.0075$.

Test Function ZDT3							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	66	56	50	0	89	0
	median	51	42	39	0	15	0
	worst	19	33	25	0	0	0
	average	46	44	39	0	26	0
	std. dev.	14.9	6.8	6.0	0	25.4	0
<i>IGD</i>	best	0.0008	0.0008	0.0007	0.0281	0.0023	0.0028
	median	0.0008	0.0009	0.0009	0.0316	0.0249	0.0054
	worst	0.0107	0.0104	0.0106	0.0447	0.0374	0.0096
	average	0.0014	0.0013	0.0018	0.0334	0.0245	0.0062
	std. dev.	0.0022	0.0021	0.0030	0.0048	0.0095	0.0020

Table 5: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function ZDT3 with respect to the unary measures.

Test Function ZDT3 - Two Set Coverage Measure <i>SC</i>						
<i>SC</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.39	0.51	0.99	0.14	0.92
NSGA-II	0.15	0.00	0.66	1.00	0.14	1.00
SPEA2	0.08	0.08	0.00	1.00	0.13	1.00
MOPSO	0.00	0.00	0.00	0.00	0.00	0.00
sMOPSO	0.39	0.42	0.41	0.99	0.00	0.75
cMOPSO	0.00	0.00	0.00	1.00	0.01	0.00
Test Function ZDT3 - Two Set Hypervolume Measure <i>HV</i>						
<i>HV</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.001699	-0.003702	-0.506543	0.004034	-0.037177
NSGA-II	0.002657	0.000000	-0.002404	-0.503446	0.006138	-0.032923
SPEA2	0.003130	0.000072	0.000000	-0.500970	0.007052	-0.030447
MOPSO	0.001259	0.000000	0.000000	0.000000	0.000000	0.000000
sMOPSO	-0.000233	-0.002485	-0.004047	-0.512069	0.000000	-0.028319
cMOPSO	0.000102	0.000000	0.000000	-0.470523	0.013227	0.000000

Table 6: Comparison of results using the binary measures for test function ZDT3. Our algorithm is denoted by OMOPSO.

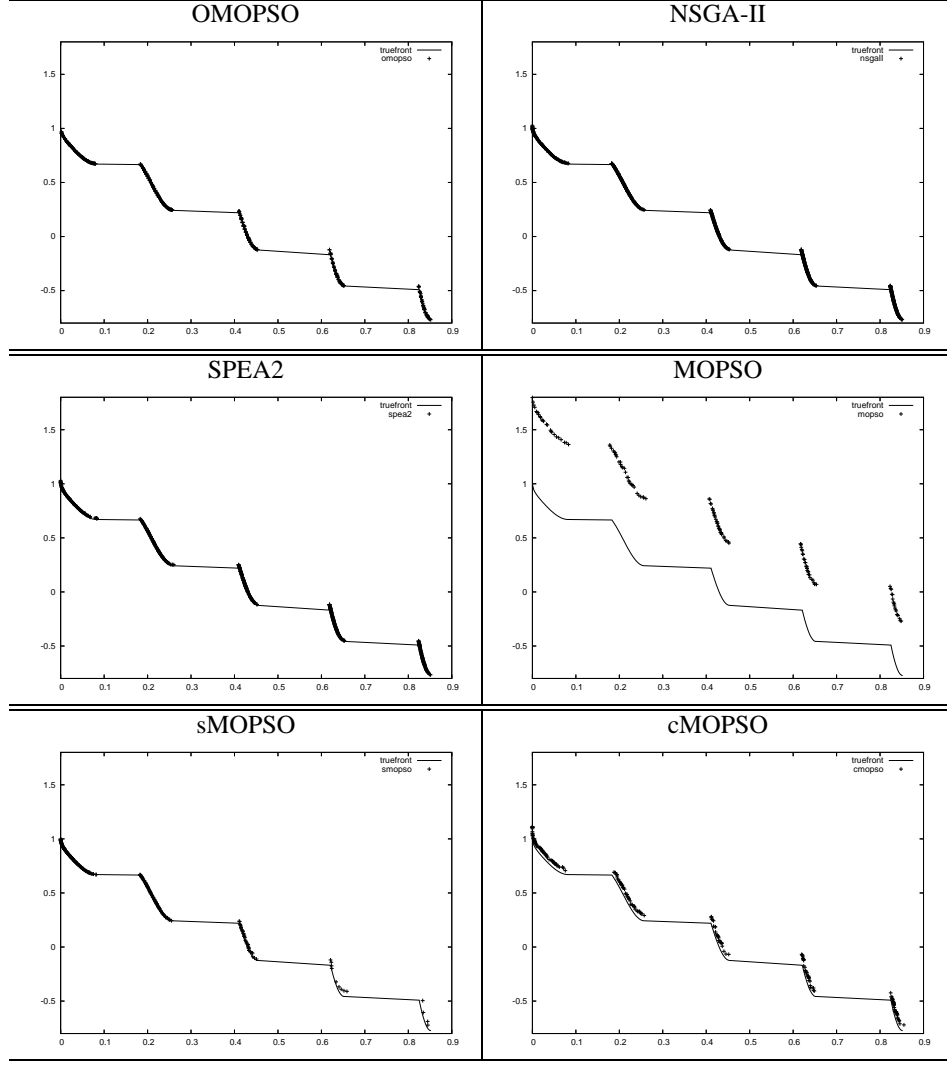


Figure 6: Pareto fronts obtained by all the approaches for test function ZDT3. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.0026$.

Test Function ZDT4							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	98	0	0	0	0	0
	median	88	0	0	0	0	0
	worst	0	0	0	0	0	0
	average	77	0	0	0	0	0
	std. dev.	26.2	0	0	0	0	0
<i>IGD</i>	best	0.0009	0.0126	0.0256	4.6415	0.1541	0.4203
	median	0.0009	0.1317	0.0811	12.407	0.7393	1.6404
	worst	0.0432	0.3219	0.3464	15.250	1.2865	4.1864
	average	0.0030	0.1508	0.1224	9.9195	0.7591	1.8621
	std. dev.	0.0095	0.0973	0.0943	4.0106	0.3147	0.9357

Table 7: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function ZDT4 with respect to the unary measures.

Test Function ZDT4 - Two Set Coverage Measure <i>SC</i>						
<i>SC</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.92	0.94	0.00	0.00	0.00
NSGA-II	0.00	0.00	1.00	1.00	1.00	1.00
SPEA2	0.00	0.00	0.00	1.00	1.00	1.00
MOPSO	0.00	0.00	0.00	0.00	0.00	0.00
sMOPSO	0.00	0.00	0.00	1.00	0.00	1.00
cMOPSO	0.00	0.00	0.00	1.00	0.00	0.00
Test Function ZDT4 - Two Set Hypervolume Measure <i>HV</i>						
<i>HV</i> (X_i)	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.164026	-0.343057	-0.333325*	-0.333325*	-0.333325*
NSGA-II	0.000574	0.000000	-0.179995	-0.497925*	-0.497925*	-0.497925*
SPEA2	0.001538	0.000000	0.000000	-0.677920*	-0.677920*	-0.677920*
MOPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
sMOPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
cMOPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Table 8: Comparison of results using the binary measures for test function ZDT4. Our algorithm is denoted by OMOPSO.

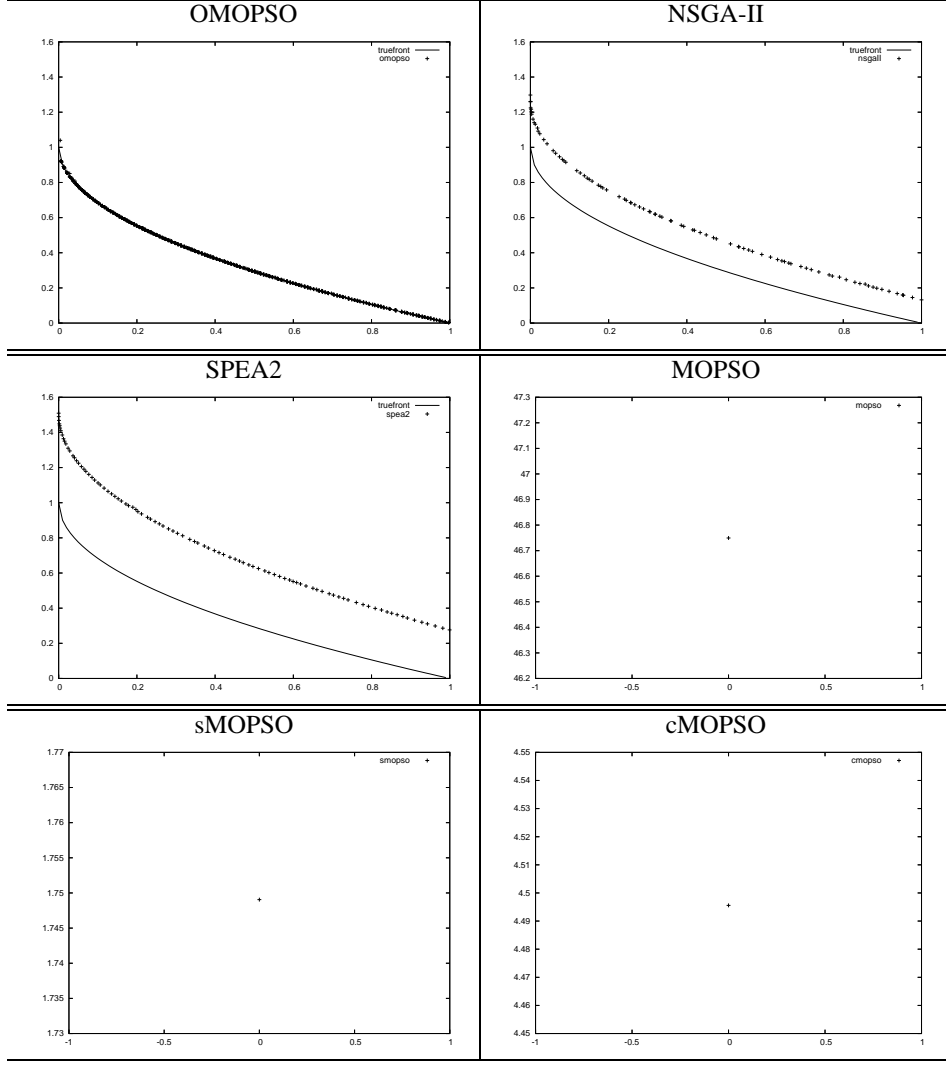


Figure 7: Pareto fronts obtained by all the approaches for test function ZDT4. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.0075$.

Test Function DTLZ2							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	25	24	26	98	32	31
	median	13	17	20	92	24	15
	worst	6	11	12	50	16	7
	average	13	16	20	89	25	16
	std. dev.	4.1	3.2	4.3	10.2	4.2	6.7
<i>IGD</i>	best	0.0015	0.0018	0.0013	0.0101	0.0013	0.0013
	median	0.0016	0.0020	0.0013	0.0118	0.0014	0.0021
	worst	0.0022	0.0025	0.0014	0.0213	0.0015	0.0027
	average	0.0017	0.0020	0.0013	0.0129	0.0014	0.0021
	std. dev.	0.0002	0.0002	0.00004	0.0030	0.00005	0.0004

Table 9: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function DTLZ2 with respect to the unary measures.

Test Function DTLZ2 - Two Set Coverage Measure <i>SC</i>						
<i>SC(X_i)</i>	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.10	0.04	0.00	0.02	0.42
NSGA-II	0.24	0.00	0.06	0.00	0.03	0.51
SPEA2	0.48	0.39	0.00	0.00	0.07	0.64
MOPSO	0.21	0.24	0.21	0.00	0.21	0.40
sMOPSO	0.60	0.53	0.35	0.00	0.00	0.97
cMOPSO	0.11	0.08	0.04	0.00	0.00	0.00
Test Function DTLZ2 - Two Set Hypervolume Measure <i>HV</i>						
<i>HV(X_i)</i>	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	0.002911	0.003373	-0.243514*	0.003036	-0.024646
NSGA-II	0.000674	0.000000	0.004201	-0.242413*	0.003977	-0.034526
SPEA2	-0.008894	-0.005829	0.000000	-0.235493*	0.003266	-0.047018
MOPSO	0.006987*	0.005851*	0.002741*	0.000000	0.002248*	0.013356*
sMOPSO	-0.012553	-0.009375	-0.000056	-0.232664*	0.000000	-0.050838
cMOPSO	0.011731	0.004088	0.001626	-0.273522*	0.001128	0.000000

Table 10: Comparison of results using the binary measures for test function DTLZ2. Our algorithm is denoted by OMOPSO.

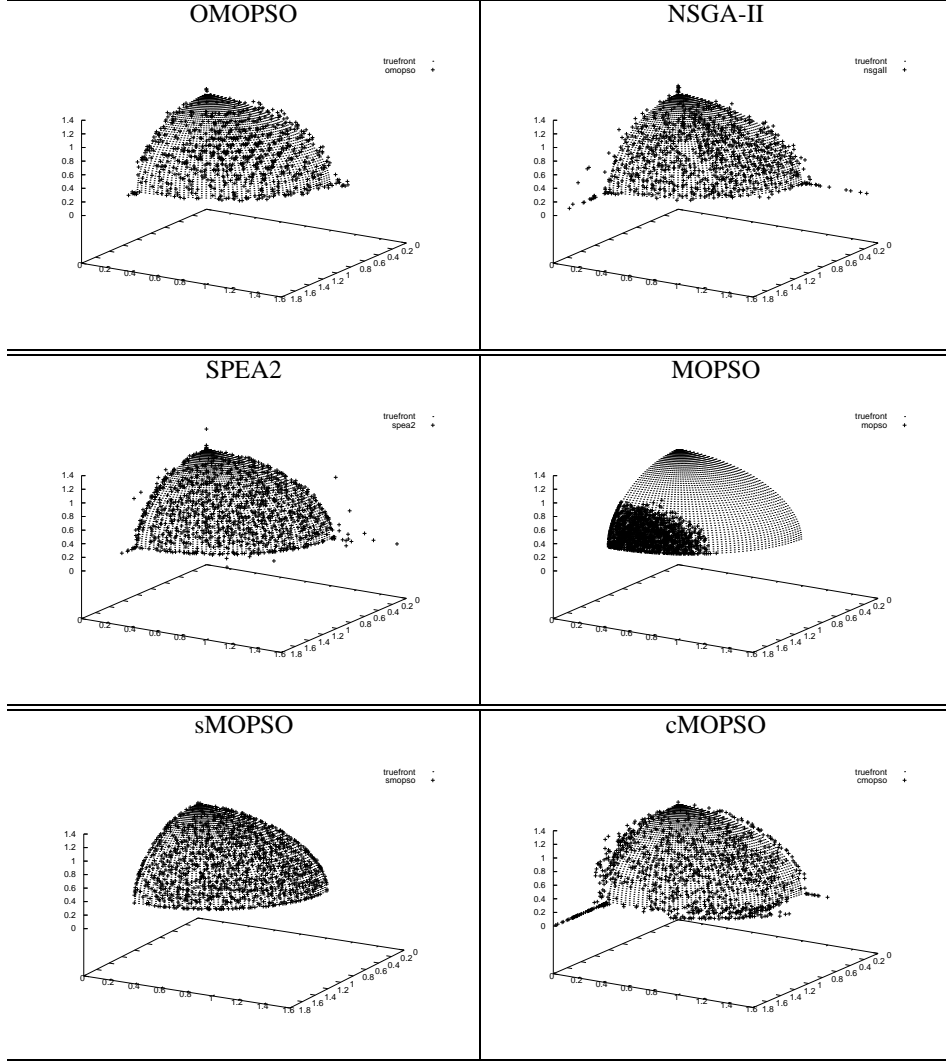


Figure 8: Pareto fronts obtained by all the approaches for test function DTLZ2. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.066$.

Test Function DTLZ4							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
<i>SCC</i>	best	98	88	91	92	68	10
	median	31	20	18	34	42	0
	worst	1	13	11	6	8	0
	average	43	34	35	37	39	1.4
	std. dev.	38.3	27.1	32.8	22.9	14.8	3.3
<i>IGD</i>	best	0.0015	0.0016	0.0013	0.0026	0.0051	0.0074
	median	0.0019	0.0018	0.0014	0.0047	0.0068	0.0234
	worst	0.0169	0.0168	0.0168	0.0117	0.0073	0.0342
	average	0.0048	0.0047	0.0045	0.0060	0.0064	0.0223
	std. dev.	0.0042	0.0054	0.0057	0.0031	0.0007	0.0074

Table 11: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function DTLZ4 with respect to the unary measures.

7 Discussion of Results

Through the use of binary measures of performance, and under certain conditions, we can conclude that a certain algorithm is better than another [43]. In this work, since we use two different binary measures, we will conclude that an algorithm is better than another when at least one of the measures indicates so, according to the definitions given in Section 6.

Since the conditions to conclude that an algorithm is better than another using the binary measures are very difficult to satisfy in most cases, we will use the values obtained by the SC binary measure in order to conclude *partial* results: We will say that an algorithm A is *relatively* better than algorithm B when $SC(A,B) > SC(B,A)$, and *almost* better than B when $SC(B,A)=0$ and $SC(A,B) > 0.9$. The values of the HV binary measure will be used to make only conclusions of the type: algorithm A is better than algorithm B, just like it was defined in Section 6.

Function ZDT1. From Table 1, we can conclude that the best results with respect to the SCC measure were obtained by the sMOPSO algorithm with an average of 59 points belonging to the true Pareto front. In this case, our proposed approach is the second best with an average of 40 points belonging to the true Pareto front. However, the sMOPSO algorithm was unable to generate the complete true Pareto front, as we can appreciate in Figure 4. This fact is reflected in the values of the IGD measure in Table 1. With respect to the IGD measure, our approach obtained results as good as those obtained by all the other MOEAs compared, improving the results obtained by the other three PSO-based approaches.

Regarding the binary measures (Table 2) and considering both of them, we can conclude that the NSGA-II, SPEA2, sMOPSO and cMOPSO are better than MOPSO. Also, we can conclude that NSGA-II and SPEA2 are better than cMOPSO. On the other hand, we can conclude that OMOPSO and sMOPSO are *relatively* better than

Test Function DTLZ4 - Two Set Coverage Measure SC						
$SC(X_i)$	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.23	0.21	0.09	0.06	0.99
NSGA-II	0.07	0.00	0.10	0.24	0.25	1.00
SPEA2	0.07	0.19	0.00	0.28	0.25	1.00
MOPSO	0.06	0.14	0.11	0.00	0.27	0.94
sMOPSO	0.05	0.09	0.08	0.10	0.00	1.00
cMOPSO	0.00	0.00	0.00	0.00	0.00	0.00
Test Function DTLZ4 - Two Set Hypervolume Measure HV						
$HV(X_i)$	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.007024	-0.006349	-0.025751*	-0.395378*	-0.492672*
NSGA-II	0.027684	0.000000	0.005186	-0.046700*	-0.430067*	-0.527380*
SPEA2	0.024974	0.001801	0.000000	-0.043671*	-0.426669*	-0.523995*
MOPSO	0.013743	0.000016*	0.000341	0.000000	-0.383423*	-0.480664*
sMOPSO	0.000364*	0.000383*	0.000396*	0.000311*	0.000000	0.000000
cMOPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Table 12: Comparison of results using the binary measures for test function DTLZ4. Our algorithm is denoted by OMOPSO.

Test Function DTLZ6							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
SCC	best	93	1	2	0	1	0
	median	23	0	0	0	1	0
	worst	0	0	0	0	1	0
	average	32	0.1	0.6	0	1	0
	std. dev.	30.9	0.3	0.9	0	0	0
IGD	best	0.0024	0.0064	0.0037	0.0375	0.0673	0.0110
	median	0.0029	0.0088	0.0045	0.0583	0.0673	0.0345
	worst	0.0213	0.0314	0.0214	0.1185	0.0673	0.0742
	average	0.0065	0.0132	0.0067	0.0658	0.0673	0.0373
	std. dev.	0.0060	0.0083	0.0051	0.0205	0.0000	0.0172

Table 13: Comparison of results between our approach (denoted by OMOPSO), NSGA-II [9], SPEA2 [41], MOPSO [5], sMOPSO [25] and cMOPSO [34], for test function DTLZ6 with respect to the unary measures.

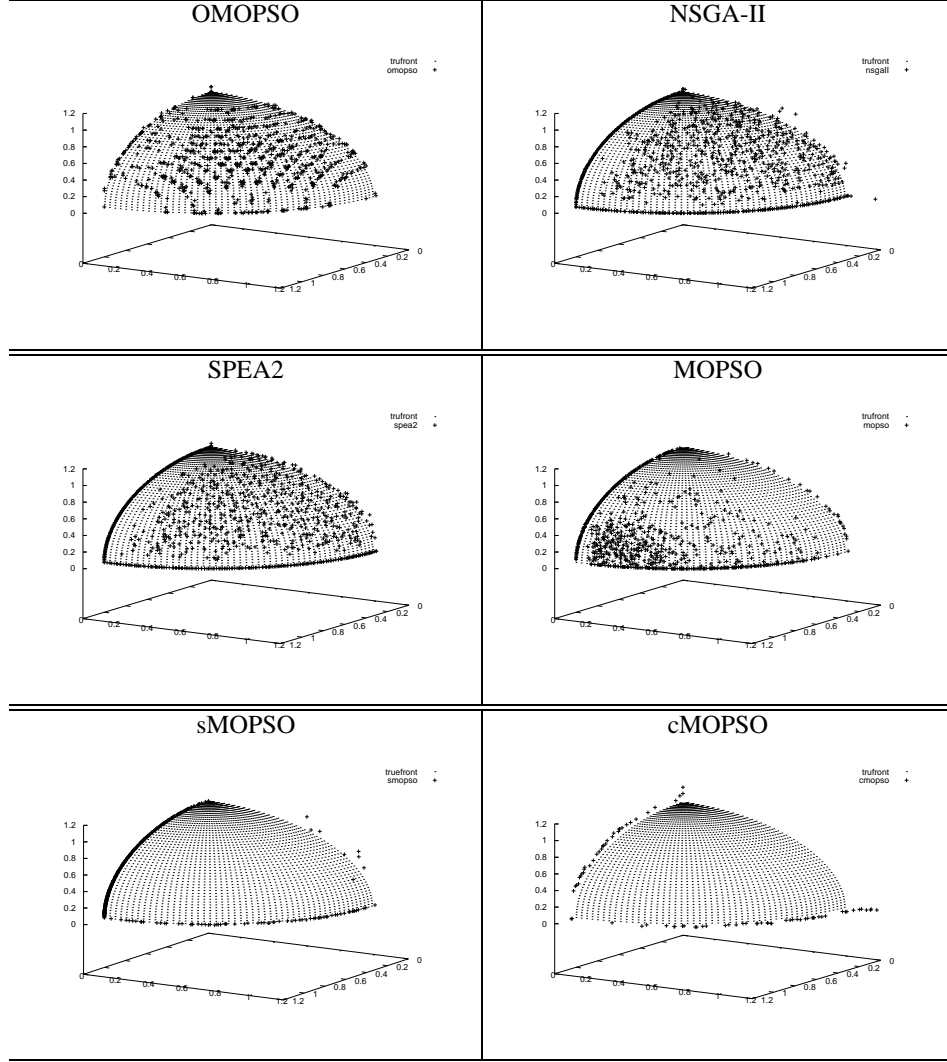


Figure 9: Pareto fronts obtained by all the approaches for test function DTLZ4. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.059$.

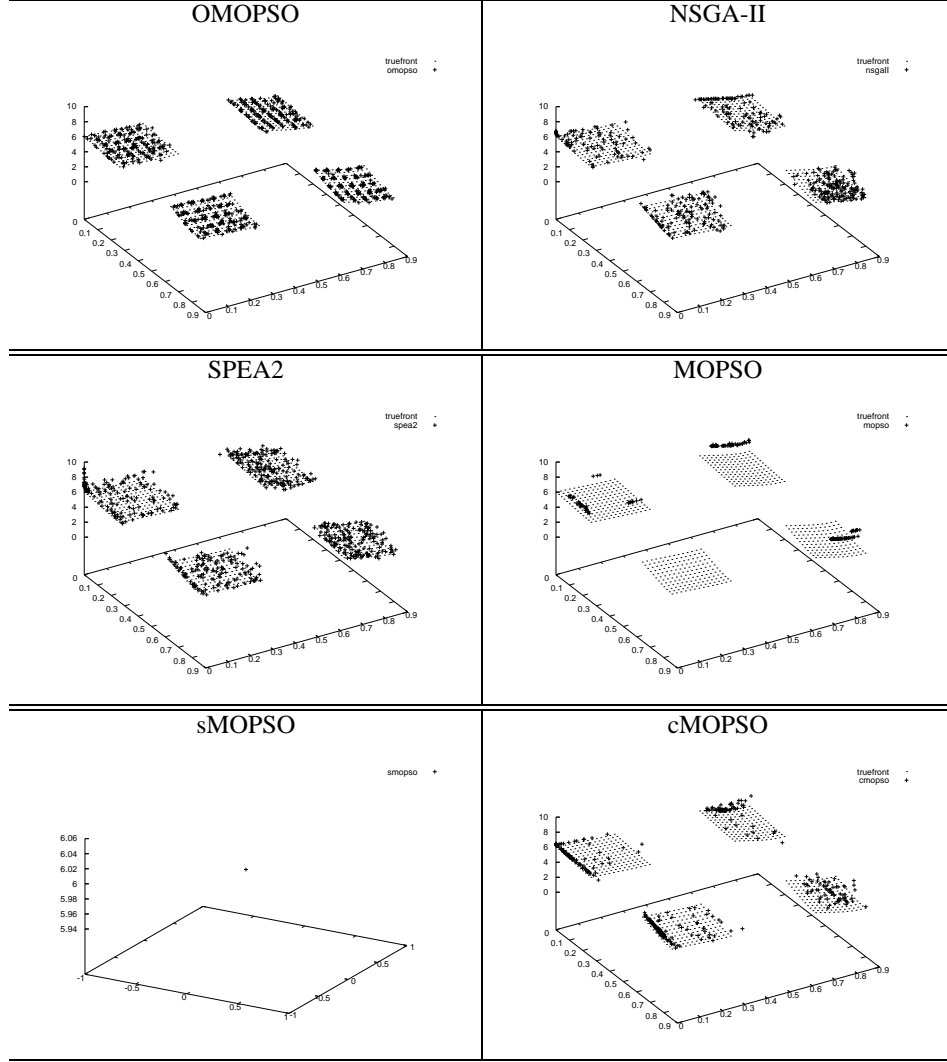


Figure 10: Pareto fronts obtained by all the approaches for test function DTLZ6. Our algorithm is denoted by OMOPSO and, in this case, it used $\epsilon=0.05$.

Test Function DTLZ6 - Two Set Coverage Measure SC						
$SC(X_i$	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.00	0.68	0.64	0.92	0.00	0.57
NSGA-II	0.01	0.00	0.31	1.00	0.00	0.80
SPEA2	0.01	0.30	0.00	0.98	0.00	0.57
MOPSO	0.00	0.00	0.00	0.00	0.00	0.00
sMOPSO	0.00	0.07	0.12	0.45	0.00	0.24
cMOPSO	0.00	0.04	0.12	1.00	0.00	0.00
Test Function DTLZ6 - Two Set Hypervolume Measure HV						
$HV(X_i$	OMOPSO)	NSGA-II)	SPEA2)	MOPSO)	sMOPSO)	cMOPSO)
OMOPSO	0.000000	-0.109957	-0.095828	-0.333723	-3.606059*	-0.629977
NSGA-II	0.109963	0.000000	0.001996	-0.113803	-3.825408*	-0.410346
SPEA2	0.141673	0.019577	0.000000	-0.096222	-3.843163*	-0.392616
MOPSO	0.000000	0.000000	0.000000	0.000000	-3.880365*	0.000000
sMOPSO	0.000000	0.000572*	0.000398*	0.059418*	0.000000	0.001163*
cMOPSO	0.000180	-0.000109	0.000040	0.296434	-4.235054*	0.000000

Table 14: Comparison of results using the binary measures for test function DTLZ6. Our algorithm is denoted by OMOPSO.

the rest of the algorithms, in particular, OMOPSO is *almost* better than MOPSO and cMOPSO. Finally, sMOPSO is *relatively* better than OMOPSO.

We will now analyze in more detail the results obtained by our algorithm in these measures. We can't conclude that OMOPSO is better than the NSGA-II (for example) since $SC(OMOPSO, NSGA-II) \neq 1$ and $SC(NSGA-II, OMOPSO) \neq 0$, but, since $SC(OMOPSO, NSGA-II) > SC(NSGA-II, OMOPSO)$, OMOPSO is *relatively* better than NSGA-II. On the other hand, we have $SC(MOPSO, OMOPSO) = 0$ and $SC(OMOPSO, MOPSO) = 0.95$, so OMOPSO is *almost* better than MOPSO. Although it should be clear that OMOPSO is better than MOPSO, the results obtained do not allow to reach this conclusion since OMOPSO lost the extreme superior point of the front, as we can see in Figure 4. This is due to the use of the ϵ -dominance scheme to fix the number of solutions in the external archive. This also explains the positive values obtained for the binary hypervolume measure in the column of OMOPSO in Table 2. In Figure 11, we show the Pareto front obtained from the union of the MOPSO and OMOPSO fronts. We can see that the hypervolume corresponding to the front shown in Figure 11 is marginally bigger than the hypervolume corresponding to the front of OMOPSO, giving a positive value to the difference in the binary measure. This exemplifies the sort of anomalous behavior that can go undetected even when using binary performance measures.

Function ZDT2. From Table 3, we can conclude that our algorithm (OMOPSO) obtained the best results in both unary measures, with the largest number of points (on average) belonging to the true Pareto front and the minimum IGD (on average).

Regarding the binary measures (considering both of them) (Table 4), we can conclude that OMOPSO, NSGA-II, SPEA2 and cMOPSO algorithms are better than MOPSO

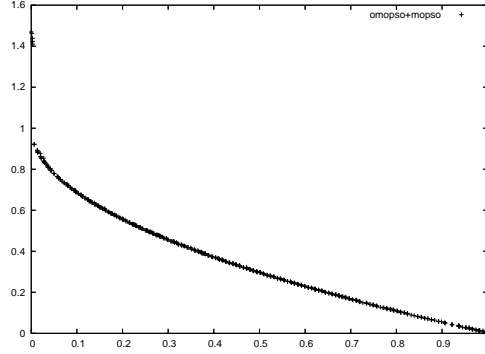


Figure 11: Pareto front obtained from the union of the fronts obtained by OMOPSO and MOPSO, in the first test function.

and sMOPSO. Also, we can say that OMOPSO is *almost* better than NSGA-II, SPEA2 and cMOPSO.

In this case, there are two interesting issues to discuss. First, we can see in the SC binary measure values that almost 80% of the points of the cMOPSO algorithm are concentrated on the top part of the true Pareto front. Thus, although the major part of the observed front of the cMOPSO algorithm (see Figure 5) is not on the true Pareto front, the corresponding results on the SC measure are not as expected. Second, the sMOPSO algorithm obtained just one point: $(0.0, 1.0)$. It is very interesting to note that none of the other algorithms were able to generate this point, as we can see in the SC measure values from Table 4. For example, in this case, our algorithm (OMOPSO) preserved two extreme points: $(0.0, 1.0005)$ and $(5 \times 10^{-10}, 1.0)$ (although they are not visible in Figure 5). These two points let the front obtained by OMOPSO to completely dominate the front obtained by MOPSO, but not the front obtained by sMOPSO. Fortunately, these problems with the SC measure are overcome by the HV measure with a small modification: the values that we have marked with an asterisk (*) in Table 4 were originally positive. However, we changed them to correspond more closely with reality, since the hypervolume corresponding to the front of sMOPSO is zero.

Function ZDT3. From the results shown in Table 5, we can conclude that our algorithm (OMOPSO) obtained the best result in the SCC measure and almost the same quality (on average) than the best result in the IGD measure (obtained in this case by the NSGA-II).

Regarding the binary measures (see Table 6), the results obtained are very similar to those on the function ZDT1: the OMOPSO algorithm is *almost* better than MOPSO and cMOPSO, it is *relatively* better than the NSGA-II and SPEA2, and it is *relatively* outperformed only by sMOPSO. We can see the Pareto fronts obtained for this function in Figure 6.

Function ZDT4. Based on the results shown in Table 7, we can conclude that our algorithm (OMOPSO) obtained the best results with respect to the two unary measures adopted, with the largest number of points (on average) belonging to the true Pareto front and the minimum IGD (on average).

Regarding the binary measures and considering both of them (see Table 8), we can conclude that OMOPSO, NSGA-II and SPEA2 are better than the other three PSO-based approaches. Also, we can say that the NSGA-II is better than SPEA2, and that OMOPSO is *almost* better than NSGA-II and SPEA2.

In this case, our approach is only *almost* better than the NSGA-II and SPEA2 for the same reason that we discussed in the case of function ZDT1. OMOPSO lost the top extreme point of the Pareto front due to the use of the ϵ -dominance scheme. For this reason, OMOPSO can't dominate completely the fronts produced by NSGA-II and SPEA2. In fact, it can't even dominate the isolated points obtained by the other PSO-based approaches. Additionally, for this same reason we find positive values in the column of OMOPSO for the binary hypervolume measure. However, the binary hypervolume measure lead us to conclude the superiority of OMOPSO compared with the other PSO-based approaches.

It is very important to note that our algorithm was the only PSO-based approach that was able to generate the entire Pareto front of this function. This illustrates the effectiveness of the mechanisms adopted in our approach to maintain diversity and to select and filter out leaders.

Function DTLZ2. From Table 9, we can conclude that the MOPSO algorithm obtained the best results in this function with respect to the SCC measure with an average of 89 points belonging to the true Pareto front. However, as we can see in Figure 8, all the points obtained by the MOPSO algorithm are concentrated on one of the inferior corners of the true Pareto front. This fact is reflected by the values obtained by the MOPSO algorithm in the IGD measure, giving the worst values in this case. In this case, although in the SCC measure our proposed approach didn't obtain better results compared with the other PSO-based approaches, the corresponding values of the IGD measure indicate that our algorithm obtained as good approximations to the true Pareto front as the other approaches.

Regarding the binary measures and considering both of them (see Table 10), we can conclude that no algorithm was better than any other. Also, we can say that the MOPSO is *relatively* better than all the algorithms and all the algorithms are *relatively* better than the cMOPSO algorithm. The sMOPSO is *relatively* better than SPEA2, the SPEA2 is *relatively* better than the NSGA-II and the NSGA-II is *relatively* better than OMOPSO. As in function ZDT2 and ZDT4, the sign of the values that we have marked with an asterisk (*) in Table 10 was changed to correspond more closely with reality, since the hypervolume corresponding to the front of MOPSO is less than the hypervolume of the other algorithms, but this fact is due to the poor distribution of the solutions obtained by MOPSO.

Function DTLZ4. From the results shown in Table 11, we can conclude that our algorithm (OMOPSO) obtained the best result in the SCC measure and almost the same quality (on average) than the best result in the IGD measure (obtained in this case

by the SPEA2).

Regarding the binary measures and considering both of them (see Table 12), we can conclude that all the algorithms are better than cMOPSO. Also, we can say that OMOPSO is *relatively* better than SPEA2, NSGA-II, MOPSO and sMOPSO. The values marked with an asterisk (*) in Table 12 were changed for the same reasons of function DTLZ2. We can see the Pareto fronts obtained for this function in Figure 9.

Function DTLZ6. From Table 13, we can conclude that our algorithm (OMOPSO) obtained the best results with respect to the two unary measures adopted, with the largest number of points (on average) belonging to the true Pareto front and the minimum IGD (on average).

Regarding the binary measures and considering both of them (see Table 14), we can conclude that OMOPSO, NSGA-II, SPEA2 and cMOPSO are better than MOPSO, and that OMOPSO is better than sMOPSO. Also, OMOPSO is *relatively* better than NSGA-II, SPEA2 and cMOPSO. The values marked with an asterisk (*) in Table 14 were changed to correspond more closely with reality, since the hypervolume corresponding to the front of sMOPSO is zero. We can see the Pareto fronts obtained for this function in Figure 10.

7.1 Overall Discussion

With respect to the unary performance measures, our approach obtained the best results in all functions, except for ZDT1 and DTLZ2. In these two functions (ZDT1 and DTLZ2), our algorithm was outperformed with respect to the SCC measure. In the case of function ZDT1, the best results with respect to the SCC measure were obtained by the sMOPSO algorithm and, in the case of function DTLZ2, by the MOPSO algorithm. However, in both cases, although those two PSO-based algorithms obtained more points belonging to the true Pareto front, both algorithms were unable to obtain a good distributed set of solutions. This was reflected by the IGD measure, since in those two functions (ZDT1 y DTLZ2), those two PSO-based algorithms obtained poor results (sMOPSO in ZDT1 and MOPSO in DTLZ2), whereas our approach obtained as good results as the best results obtained in each case. Thus, this indicates that OMOPSO was able to obtain a good approximation and a good number of points of the true Pareto of all the test functions used in this paper.

Regarding the binary measures, given the same reasons discussed previously, our approach was *relatively* outperformed only in three (out of seven) functions (ZDT1, ZDT3, and DTLZ2). Note that the “*relatively* better” criterion is based on the values of the SC binary measure, related directly with the SCC unary measure. However, OMOPSO was at least *relatively* better than almost all the algorithms in all functions, except in function DTLZ2. In fact, due to the use of ϵ -dominance, our approach lost the extreme superior points of the Pareto fronts, and for this reason it was not possible to conclude its superiority over some of the other algorithms in several functions (for example function ZDT4).

In general, OMOPSO was clearly superior compared with the other PSO-based approaches adopted in our comparative study. Also, the results obtained by OMOPSO

showed that it is highly competitive with respect to both NSGA-II and SPEA2, which are two algorithms representative of the state-of-the-art in evolutionary multi-objective optimization.

8 Conclusions and Future Work

We have proposed a new multi-objective particle swarm optimizer which uses Pareto dominance and a crowding-based selection mechanism to identify the leaders to be removed when there are too many of them. The selection of such in-excess leaders has been a topic often disregarded in the literature of multi-objective particle swarm optimizers, but it is a key issue to design robust and effective PSO-based multi-objective optimizers. This is clearly illustrated in this paper, since our approach was able to outperform the other PSO-based algorithms. Additionally, our approach was the only algorithm able to generate the Pareto front of a problem for which no other PSO-based approach was able to work properly.

Our approach uses a single population (or swarm). However, for mutation purposes, the swarm is subdivided in three parts of equal size and different mutation operators (taken from the EA literature) are applied to each of them (including no mutation at all). We also adopt the ϵ -dominance concept to fix the size of the set of final solutions produced by the algorithm.

After performing a comparative study with respect to three other PSO-based approaches and two highly competitive multi-objective evolutionary algorithms (the NSGA-II and SPEA2), we found our proposed approach to be highly competitive. Our results indicate superiority of our technique with respect to the other PSO-based approaches and a very similar behavior with respect to the NSGA-II and SPEA2.

As part of our future work, we are interested on improving the distribution of the final set of solutions obtained by our algorithm. We intend to fix the problem with the loss of the extrema of the Pareto fronts caused by the use of ϵ -dominance. This problem was observed in all the test functions used in this paper, however, it was more evident in the three-objective functions. So, it seems that the loss of extrema points tends to intensify as the number of objectives grows. In addition, we plan to implement an on-line adaptation mechanism of the ϵ parameter. In this way, the user won't need to tune this parameter of the algorithm.

We are also interested in studying the convergence properties of the PSO strategy with the aim of exploring mechanisms that can accelerate convergence while keeping the same quality of results currently achieved.

Furthermore, we have in mind to investigate processes that allow our approach to stop the search automatically (i.e., without having to define a maximum number of iterations) [18, 33].

Finally, we plan to study the impact of the use of different mutation operators and also the effect of the different parameters that are involved in the flight formula used by the PSO strategy, which is the main operator of this heuristic.

Acknowledgments

We thank Sanaz Mostaghim for providing us the source code of her Sigma-MOPSO. The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN. The second author acknowledges support from CONACyT project number 42435-Y.

References

- [1] Richard Balling. The Maximin Fitness Function; Multiobjective City and Regional Planning. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 1–15, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
- [2] Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.
- [3] U. Baumgartner, Ch. Magele, and W. Renhart. Pareto Optimality and Particle Swarm Optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, March 2004.
- [4] Chi-kin Chow and Hung-tat Tsui. Autonomous Agent Response Learning by a Multi-Species Particle Swarm Optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 1, pages 778–785, Portland, Oregon, USA, June 2004. IEEE Service Center.
- [5] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.
- [6] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.
- [7] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.
- [8] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.
- [9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

- [10] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.
- [11] Jonathan E. Fieldsend and Sameer Singh. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, pages 37–44, Birmingham, UK, September 2002.
- [12] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.
- [13] Xiaohui Hu and Russell Eberhart. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1677–1681, Piscataway, New Jersey, May 2002. IEEE Service Center.
- [14] Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi. Particle Swarm with Extended Memory for Multiobjective Optimization. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 193–197, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.
- [15] Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff. Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042–1049, San Francisco, California, 2001. Morgan Kaufmann Publishers.
- [16] James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, New Jersey, 1995. IEEE Service Center.
- [17] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.
- [18] Rajeev Kumar and Peter Rockett. Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm. *Evolutionary Computation*, 10(3):283–314, Fall 2002.
- [19] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.
- [20] Xiaodong Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 37–48. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

- [21] Xiaodong Li. Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation—GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 117–128, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.
- [22] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, third edition, 1996.
- [23] Jacqueline Moore and Richard Chapman. Application of Particle Swarm to Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.
- [24] Sanaz Mostaghim and Jürgen Teich. The role of ε -dominance in multi objective particle swarm optimization methods. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1764–1771, Canberra, Australia, December 2003. IEEE Press.
- [25] Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 26–33, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.
- [26] Sanaz Mostaghim and Jürgen Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1404–1411, Portland, Oregon, USA, June 2004. IEEE Service Center.
- [27] K.E. Parsopoulos, D.K. Tasoulis, and M.N. Vrahatis. Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, volume 2, pages 823–828, Innsbruck, Austria, February 2004. ACTA Press.
- [28] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization Method in Multiobjective Problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, pages 603–607, Madrid, Spain, 2002. ACM Press.
- [29] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.
- [30] Tapabrata Ray and K.M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization*, 34(2):141–153, March 2002.

- [31] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.
- [32] Dipti Srinivasan and Tian Hou Seow. Particle Swarm Inspired Evolutionary Algorithm (PS-EA) for Multiobjective Optimization Problem. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 2292–2297, Canberra, Australia, December 2003. IEEE Press.
- [33] Gregorio Toscano Pulido and Carlos A. Coello Coello. The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 252–266, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
- [34] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation—GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 225–237, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.
- [35] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.
- [36] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.
- [37] David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.
- [38] L.B. Zhang, C.G. Zhou, X.H. Liu, Z.Q. Ma, and Y.C. Liang. Solving Multi Objective Optimization Problems Using Particle Swarm Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 2400–2405, Canberra, Australia, December 2003. IEEE Press.
- [39] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

- [40] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.
- [41] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al., editor, *Proceedings of the EUROGEN2001 Conference*, pages 95–100, Barcelona, Spain, 2002. CIMNE.
- [42] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.
- [43] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.