Improving PSO-based Multi-Objective Optimization using Crowding, Mutation and ϵ-Dominance

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

Abstract. In this paper, we propose a new Multi-Objective Particle Swarm Optimizer, which is based on Pareto dominance and the use of a crowding factor to filter out the list of available leaders. We also propose the use of different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. Finally, the proposed approach also incorporates the ϵ-dominance concept to fix the size of the set of final solutions produced by the algorithm. Our approach is compared against five state-of-the-art algorithms, including three PSO-based approaches recently proposed. The results indicate that the proposed approach is highly competitive, being able to approximate the front even in cases where all the other PSO-based approaches fail.

1 Introduction

Kennedy and Eberhart [1] 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 $\boldsymbol{x}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\boldsymbol{v}_i(t)$ to its current position, i.e.: $\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t)$. The velocity vector drives the optimization process and reflects the socially exchanged information. In the global best version (used here) of PSO, the social knowledge used to drive the movement of particles includes the position of the best particle from the entire swarm ($gbest$) and its history of experiences in terms of its own best solution thus far ($pbest$). In this case, the velocity vector changes in the following way: $\boldsymbol{v}_i(t) = W\boldsymbol{v}_i(t-1) + C_1 r_1(\boldsymbol{x}_{pbest_i} - \boldsymbol{x}_i(t)) + C_2 r_2(\boldsymbol{x}_{gbest} - \boldsymbol{x}_i(t))$, 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. The successful application of PSO in many single objective optimization problems reflects the effectiveness of PSO. However, in order to handle multiple objectives, PSO must be obviously modified. In most approaches (which will be generically called MOPSOs, for Multiple-Objective Particle

Swarm Optimizers), the major modifications to the basic PSO algorithm are the selection process of $pbest$ and $gbest$ [2, 3]. In this paper, we present a new proposal which is based on Pareto dominance and the use of a crowding factor for the selection of leaders. We also incorporate mutation operators (taken from the evolutionary algorithms literature) and the concept of ϵ-dominance. This paper is organized as follows. The previous related work is reviewed in Section 2. In Section 3, we describe our proposed approach. The obtained results and discussion are presented in Sections 4 and 5, respectively. Finally, the conclusions and future work are described in Section 6.

2 Related Work

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

Ray and Liew [4]: This algorithm uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. The approach uses crowding to maintain diversity and a multilevel sieve to handle constraints.

Hu and Eberhart [5]: In this algorithm, only one objective is optimized at a time using a scheme similar to lexicographic ordering. In further work, Hu et al. [6] adopted a secondary population (called "extended memory") and introduced some further improvements to their dynamic neighborhood PSO approach.

Fieldsend and Singh [7]: 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" (or mutation) operator.

Coello et al. [2]: This approach uses a global repository in which every particle deposits its flight experiences. 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 [8] adopted 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 swarm to variate the selection pressure.

Mostaghim and Teich [3]: 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. In further work, the authors [9] studied the influence of ϵ-dominance [10] 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. In more recent work, the authors [11] proposed 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 phase 2, the non-dominated solutions obtained from 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 phase 1.

Li [12]: This approach incorporates the main mechanisms of the NSGA-II [13] into PSO. It combines the population of particles and all the $pbest$ positions of each particle, and selects the best from them to conform the next population. It also selects the leaders randomly from the leaders set based on both a niche count and a crowding distance. In more recent work, Li [14] proposed the *maximinPSO*, which uses a fitness function that requires no additional clustering or niching procedure to maintain diversity.

3 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. In our approach, we use a crowding factor [13] 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). 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 [4, 12]), 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. We also propose the use of two mutation operators that are well-known in the EA literature: uniform mutation (i.e., the variability range allowed for each decision variable is kept constant over generations) and non-uniform mutation (i.e., the variability range allowed for each decision variable decreases over time). 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. Given the uncertainty regarding the type of mutation to apply, we proposed 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. With the use of these different operators we are aiming to have the ability of exploring (uniform mutation) 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 normally used in the EA literature [15]: the mutation rate is defined as $1/codesize$, where $codesize$ refers to the total length of the string that encodes all the decision variables of the problem (the number of variables in our case). Finally, we adopt the concept of ϵ-dominance [10] 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: $f_i(x_1)/(1+\epsilon) \leq f_i(x_2), \forall i = 1, ..., m$ and $f_i(x_1)/(1+\epsilon) < f_i(x_2),$ for at least one $i = 1, ..., m$. 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 [10]. For the sake of simplicity, in this paper, we consider the same value of ϵ for all the objective functions of a given problem. For each problem, the value of ϵ was tuned based on the desired amount of points in the final Pareto-front. Figure 1 shows the way in which our algorithm works. 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: $\boldsymbol{v}_i(t) = W\boldsymbol{v}_i(t-1) + C_1 r_1(\boldsymbol{x}_{pbest_i} - \boldsymbol{x}_i(t)) + C_2 r_2(\boldsymbol{x}_{gbest} - \boldsymbol{x}_i(t)),$ where $W = random(0.1, 0.5)$, $C_1, C_2 = random(1.5, 2.0)$, and $r_1, r_2 = random$ $(0.0, 1.0)$. 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 [8]. 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. Then, we proceed to evaluate the particle and update its personal best value ($pbest$). A new particle replaces its $pbest$ value if such value is dominated by the new particle or if both are 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 $pbest$ 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), (2) $gmax$ (number of iterations), (3) pm (mutation rate—automatically computed), and (4) ϵ (value for bounding the size of the ϵ-archive).

```
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 $pbest$.
        EndFor
        Update leaders, Send leaders to $\epsilon$-archive
        crowding(leaders), $g$++
    EndWhile
    Report results in $\epsilon$-archive
End
```

Fig. 1. Pseudocode of our algorithm.

4 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 SPEA2 [16] and the NSGA-II [13]. We also compared our approach against three PSO-based approaches recently proposed: MOPSO [2], Sigma-MOPSO [3] and Cluster-MOPSO [8]. For our comparative study, we implemented two unary and two binary measures of performance. The following are the unary measures:

Success Counting (SCC): This measure counts 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. 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.

Inverted Generational Distance (IGD): In this measure, 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. This measure is defined as: $IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$ where n is the number of elements in the true Pareto front and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the set of nondominated vectors found by the algorithm. It should be clear that a value of $IGD = 0$ indicates that all the elements generated are in the true Pareto front of the problem.

The binary measures adopted are the following:

Two Set Coverage (SC): This measure was proposed in [17]. 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 $[0, 1]$: $SC(X', X'') \triangleq |\{a'' \epsilon X''; \exists a' \epsilon X' : a' \preceq a''\}|/|X''|$. 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 [18]. 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 Sigma-MOPSO and Cluster-MOPSO were provided by their authors. The code of our approach is available via email request to the first author. We adopted several test functions [19], however, given the available space we only present results corresponding to the following four: ZDT1, ZDT2, ZDT4 [20] and DTLZ6 [21]. 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 $popsize$=100 and 200 generations. As recommended in [21], for the NSGA-II and SPEA2, 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 (author suggestion). As recommended in [3], 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 $1/codesize$. The Pareto fronts that we will show correspond to the nondominated vectors obtained from the union of the 20 Pareto fronts produced by each approach. It should be noted that the Pareto fronts shown were also used to apply the binary measures of performance. All the algorithms, except for cMOPSO, were set such that they provided Pareto fronts with 100 points. cMOPSO does not have a scheme to fix the size of its final archive. Thus, in order to allow a fair comparison the values of the SCC measures were scaled (in the case of cMOPSO) to the interval [0,100]. From Table 1 to Table 6 we show the values of the performance measures obtained for each of the algorithms compared.

[1] `http://www.tik.ee.ethz.ch/pisa/`

[2] `http://delta.cs.cinvestav.mx/~ccoello/EMOO`

Test Function ZDT1

		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
SCC	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
IGD	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

Test Function ZDT1 - Two Set Coverage Measure SC

$SC(X,$	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 HV

$HV(X,$	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 1. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **ZDT1**.

Test Function ZDT2

		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
SCC	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
IGD	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 2. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **ZDT2**, with respect to the unary measures.

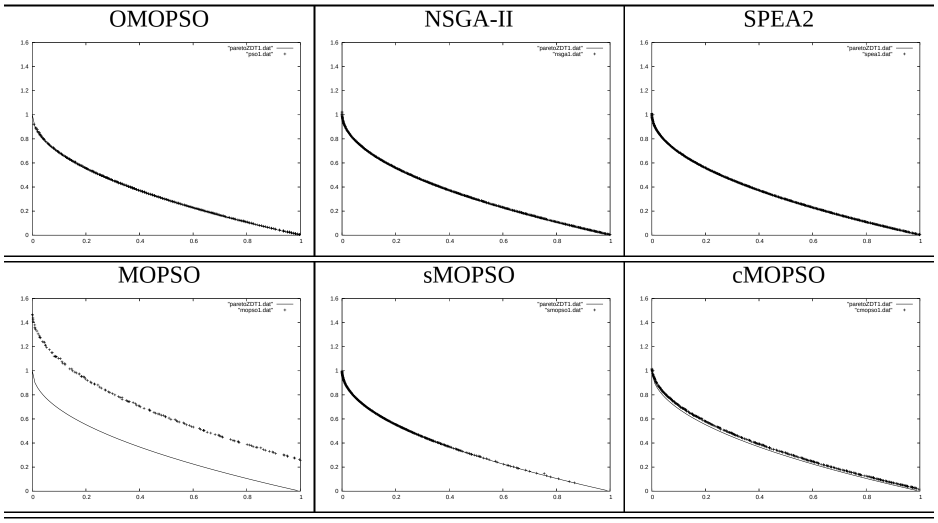

Fig. 2. Pareto fronts obtained by all the approaches for **ZDT1**. Our algorithm is denoted by OMOPSO and, in this case, it used ϵ=0.0075.

Test Function ZDT2 - Two Set Coverage Measure SC						
$SC(X,$	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 HV						
$HV(X,$	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 3. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **ZDT2**, with respect to the binary measures.

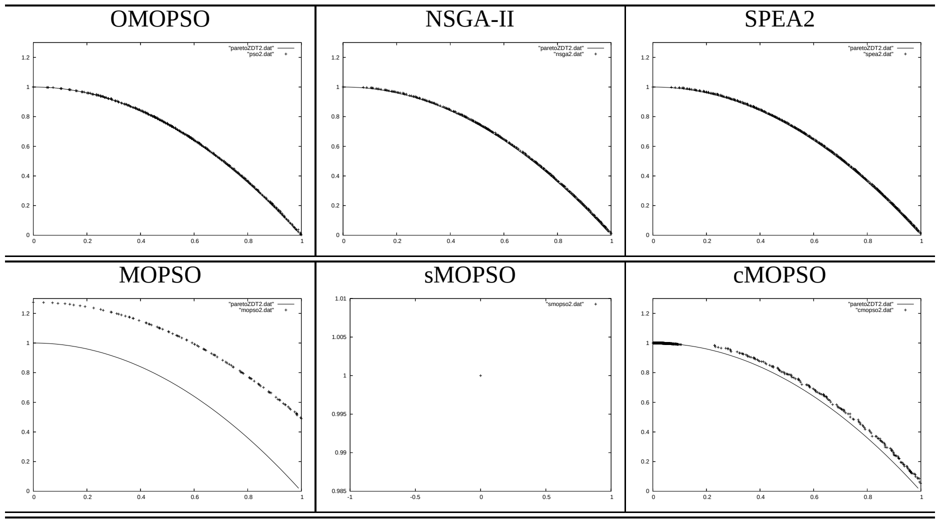

Fig. 3. Pareto fronts obtained by all the approaches for **ZDT2**. Our algorithm is denoted by OMOPSO and, in this case, it used ϵ=0.0075.

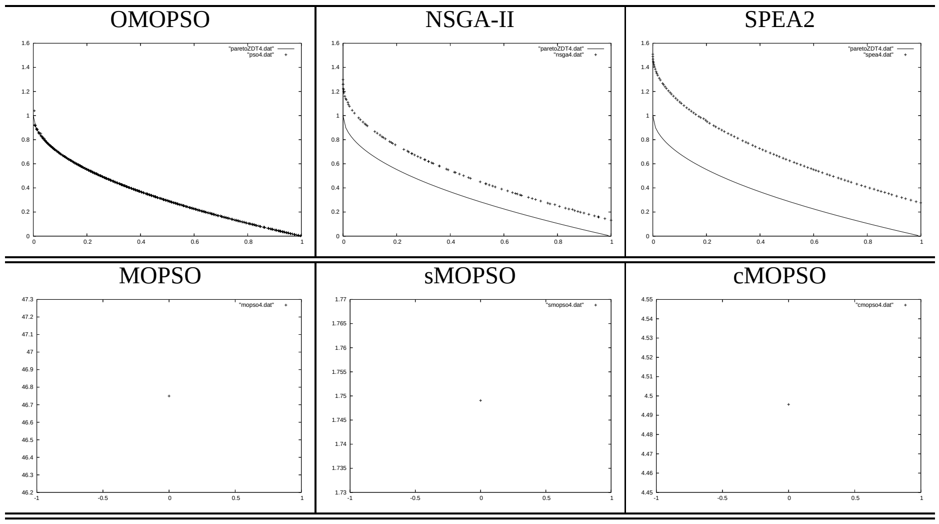

Fig. 4. Pareto fronts obtained by all the approaches for **ZDT4**. Our algorithm is denoted by OMOPSO and, in this case, it used ϵ=0.0075.

Test Function ZDT4							
		OMOPSO	NSGA-II	SPEA2	MOPSO	sMOPSO	cMOPSO
SCC	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
IGD	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

Test Function ZDT4 - Two Set Coverage Measure SC						
$SC(X,$	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 HV						
$HV(X,$	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 4. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **ZDT4**.

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 5. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **DTLZ6**, with respect to the unary measures.

Test Function DTLZ6 - Two Set Coverage Measure SC						
$SC(X,$	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						
$HV(X,$	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 6. Comparison of results between our approach (denoted by OMOPSO), NSGA-II [13], SPEA2 [16], MOPSO [2], sMOPSO [3] and cMOPSO [8], for **DTLZ6**, with respect to the binary measures.

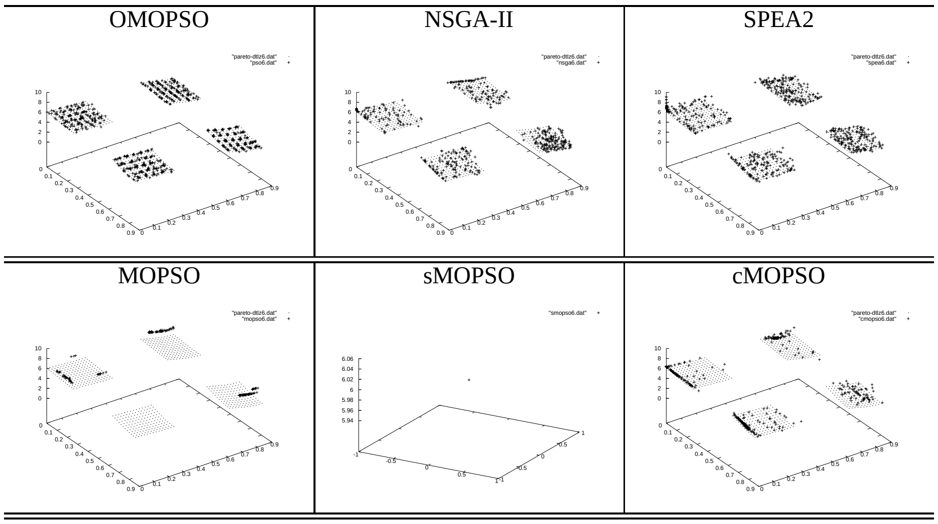

Fig. 5. Pareto fronts obtained by all the approaches for **DTLZ6**. Our algorithm is denoted by OMOPSO and, in this case, it used ϵ=0.05.

5 Discussion of Results

Through the use of binary measures of performance, and under certain conditions, we can conclude that an algorithm is better than another [22]. 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 4. Since the conditions to conclude that an algorithm is better that another one 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 4.

ZDT1. From Table 1, we can conclude that the best results with respect to the SCC measure were obtained by the sMOPSO algorithm. In this case, our proposed approach is the second best. However, the sMOPSO algorithm was unable to generate the complete Pareto front, as we can appreciate in Figure 2. 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 1) and considering both of them, we can conclude that the NSGA-II and SPEA2 are better than cMOPSO and MOPSO. Also, we can conclude that sMOPSO and cMOPSO are better than MOPSO. On the other hand, we can conclude that OMOPSO and sMOPSO are *relatively* better than 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 2. 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 1, since the hypervolume corresponding to the front obtained from the union of the MOPSO and OMOPSO fronts 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.

ZDT2. From Table 2, 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 3), we can conclude that OMOPSO, NSGA-II, SPEA2 and cMOPSO are better than MOPSO and sMOPSO. Also, we can say that OMOPSO is *almost* better than NSGA-II, SPEA2 and cMOPSO. 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 3) is not on the true Pareto front, the corresponding results on the SC measure are not as expected. Additionally, the sMOPSO algorithm obtained just one point: (0.0,1.0). None of the other algorithms were able to generate this point, as we can see in the SC measure values from Table 3. 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 3). 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 modifi cation: the values that we have marked with an asterisk (*) in Table 3 were originally positive. However, we changed them to correspond more closely with reality, since the hypervolume corresponding to the front of sMOPSO is zero.

ZDT4. Based on the results shown in Table 4, we can conclude that our 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 4), 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. As we can see in Figure 4, 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 fi nd 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 fi lter out leaders.

DTLZ6. From Table 5, 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 6), 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. We can see the Pareto fronts obtained for this function in Figure 5.

Overall Discussion. With respect to the unary performance measures, our approach obtained the best results in all functions, except for ZDT1. 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, although in function ZDT1 our approach was relatively outperformed by sMOPSO, 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.

6 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. 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 intend to fix the problem with the loss of the extrema of the Pareto fronts caused by the use of $\epsilon-$dominance. We are also interested in exploring mechanisms that can accelerate convergence and that allow our approach to stop the search automatically (i.e., without having to define a maximum number of iterations).

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. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, New Jersey, IEEE Service Center (1995) 1942–1948
2. Coello Coello, C.A., Toscano Pulido, G., Salazar Lechuga, M.: Handling Multiple Objectives With Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation **8** (2004) 256–279
3. Mostaghim, S., Teich, J.: Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 26–33
4. Ray, T., Liew, K.: A Swarm Metaphor for Multiobjective Design Optimization. Engineering Optimization **34** (2002) 141–153

5. Hu, X., Eberhart, R.: Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In: Congress on Evolutionary Computation (CEC'2002). Volume 2., Piscataway, New Jersey, IEEE Service Center (2002) 1677–1681
6. Hu, X., Eberhart, R.C., Shi, Y.: Particle Swarm with Extended Memory for Multiobjective Optimization. In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 193–197
7. Fieldsend, J.E., Singh, S.: 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, Birmingham, UK (2002) 37–44
8. Toscano Pulido, G., Coello Coello, C.A.: Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In et al., K.D., ed.: Proceedings of the 2004 Genetic and Evolutionary Computation Conference. Part I, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102 (2004) 225–237
9. Mostaghim, S., Teich, J.: 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., Canberra, Australia, IEEE Press (2003) 1764–1771
10. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. Evolutionary Computation **10** (2002) 263–282
11. Mostaghim, S., Teich, J.: Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In: 2004 Congress on Evolutionary Computation (CEC'2004). Volume 2., Portland, Oregon, USA, IEEE Service Center (2004) 1404–1411
12. Li, X.: A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In et al., E.C.P., ed.: Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I, Springer. Lecture Notes in Computer Science Vol. 2723 (2003) 37–48
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
14. Li, X.: Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function. In et al., K.D., ed.: Proceedings of the 2004 Genetic and Evolutionary Computation Conference. Part I, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102 (2004) 117–128
15. Bäck, T., ed.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In et al., K.G., ed.: Proceedings of the EUROGEN2001 Conference, Barcelona, Spain, CIMNE (2002) 95–100
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In Eiben, A.E., ed.: Parallel Problem Solving from Nature V, Amsterdam, Springer-Verlag (1998) 292–301
18. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (1999)
19. Reyes-Sierra, M., Coello, C.A.C.: A New Multi-Objective Particle Swarm Optimizer with Improved Selection and Diversity Mechanisms. Technical Report EVOCINV-05-2004, Sección de Computación, Depto. de Ingeniería Eléctrica, CINVESTAV-IPN, México (2004)
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8** (2000) 173–195
21. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation (CEC'2002). Volume 1., Piscataway, New Jersey, IEEE Service Center (2002) 825–830
22. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation **7** (2003) 117–132