# A Multi-Objective Particle Swarm Optimizer Enhanced with a Differential Evolution Scheme

Jorge S. Hernández Domínguez[1], Gregorio Toscano Pulido[1] and Carlos A. Coello Coello[2]

[1]Information Technology Laboratory, CINVESTAV - Tamaulipas, Parque Científico y Tecnológico TECNOTAM. Km. 5.5, carretera Cd. Victoria-Soto La Marina. Cd. Victoria, Tamaulipas, 87130, MÉXICO
jhernandez@tamps.cinvestav.mx, gtoscano@cinvestav.mx
[2]CINVESTAV-IPN (Evolutionary Computation Group), Depto. de Computación, Av. IPN No. 2508, San Pedro Zacatenco, México, D.F. 07360, MÉXICO
ccoello@cs.cinvestav.mx

**Abstract.** Particle swarm optimization (PSO) and differential evolution (DE) are meta-heuristics which have been found to be very successful in a wide variety of optimization tasks. The high convergence rate of PSO and the exploratory capabilities of DE make them highly viable candidates to be used for solving multi-objective optimization problems (MOPs). In previous studies that we have undertaken [2], we have observed that PSO has the ability to launch particles in the direction of a leader (i.e., a non-dominated solution) with a high selection pressure. However, this high selection pressure tends to move the swarm rapidly towards local optima. DE, on the other hand, seems to move solutions at smaller steps, yielding solutions close to their parents while exploring the search space at the same time. In this paper, we present a multi-objective particle swarm optimizer enhanced with a differential evolution scheme which aims to maintain diversity in the swarm while moving at a relatively fast rate. The goal is to avoid premature convergence without sacrificing much the convergence rate of the algorithm. In order to design our hybrid approach, we performed a series of experiments using the ZDT test suite. In the final part of the paper, our proposed approach is compared (using 2000, 3500, and 5000 objective function evaluations) with respect to four state-of-the-art multi-objective evolutionary algorithms, obtaining very competitive results.

## 1 Introduction

Particle swarm optimization (PSO) [4] is a meta-heuristic that mimics the behavior of bird flocks by "searching" based on social and personal knowledge acquired by a set of particles. Due to its effectiveness in single-objective optimization, PSO has been extended to multi-objective optimization problems (MOPs). In addition, differential evolution (DE) [6] is another meta-heuristic which has been particularly successful as a single-objective optimizer. DE works by mutating solutions based on the population's variance and this strategy has been

found to be a very powerful optimizer in continuous search spaces. Similar to PSO, several DE algorithms have been migrated to multi-objective optimization. Even though both meta-heuristics (PSO and DE) are simple to conceptualize and have shown competitive results on a variety of MOPs, relatively few research has been performed in regards to comparing and contrasting these two meta-heuristics in a multi-objective context. We believe that obtaining more detailed knowledge about the search capabilities of multi-objective evolutionary algorithms (MOEAs) such as these, is of utmost importance to design more powerful algorithms. For example, the few studies currently available indicate that several multi-objective particle swarm optimizers (MOPSOs) have difficulties to deal with multifrontal problems [3], while multi-objective differential evolution (MODE) approaches have shown better results on this type of problems [8,2]. The different behavior of MOPSO and MODE on this type of problems may serve as indicative that a better understanding of these two meta-heuristics will be fruitful.

In this article, we propose a hybrid MOEA that attempts to combine the advantages of MOPSO and MODE. Some of the features adopted for our hybrid approach were obtained from a series of experiments (some of these experiments are included in this document while others were obtained from our previous research [2]) performed on the Zitzler-Deb-Thiele (ZDT) test suite. As a result, aiming to adopt the mechanisms that promote desirable effects (in MODE and MOPSO), we have devised a MOEA which shows competitive results (using the IGD [9] performance measure) when compared to four state-of-the-art MOEAs.

The remainder of this paper is organized as follows. Section 2 introduces some basic concepts related to multi-objective optimization as well as the PSO and DE algorithms. In Section 3, we show a series of experiments that allowed us to better understand the way in which a MODE and a MOPSO algorithms perform the search. Then, in Section 4, we introduce the designed algorithm whose performance is assessed in Section 5. Finally, our conclusions and some possible paths for future research are provided in Section 6.

## 2 Basic Concepts

In this study, we assume that all the objectives are to be minimized and are equally important. We are interested in solving the general *multi-objective optimization problem* with the following form:

$$\begin{aligned} &\text{Minimize} \;\; \mathrm{f}(\mathbf{X}_i) = (f_1(\mathbf{X}_i), f_2(\mathbf{X}_i), \ldots, f_M(\mathbf{X}_i))^T \\ &\text{subject to} \;\; \mathbf{X}_i \in \mathcal{F} \end{aligned} \tag{1}$$

where $\mathbf{X}_i$ is a *decision vector* (containing our *decision variables*), $\mathrm{f}(\mathbf{X}_i)$ is the $M$-dimensional *objective vector*, $f_m(\mathbf{X}_i)$ is the $m$-th objective function, and $\mathcal{F}$ is the feasible region delimited by the problem's constraints.

### 2.1 Particle Swarm Optimization

The flight of particles in PSO is typically directed by the following three components: $i$) ***velocity*** - this component is conformed by a velocity vector which aids in moving the particle to its next position. Moreover, an inertia weight $w$ is used to control the amount of previous velocity to be applied. $ii$), ***cognitive*** - this component represents the "memory" of a particle. This is done with a *personal best* vector (we will refer to this vector as pbest) which remembers the best position found so far by a particle. $iii$) ***social*** - this component represents the position of a particle known as the *leader*. The leader is the particle with the best performance on the neighborhood of the current particle. These three components of PSO can be seen on its flying formula. When a particle is about to update its position, a new velocity vector is computed using:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1 r_1(t)(\mathbf{y}_i(t) - \mathbf{x}_i(t)) + c_2 r_2(t)(\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) \tag{2}$$

Thereafter, the position of the particle is calculated using the new velocity vector:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{3}$$

In Equations (2) and (3), $\mathbf{x}_i(t)$ denotes the position of particle $i$ at time $t$, $\mathbf{v}_i(t)$ denotes the velocity of particle $i$ at time $t$, $w$ is the inertia weight, $c_1$ and $c_2$ are the cognitive and social factors, respectively, and $r_1$, $r_2 \sim$ U(0,1). Additionally, $\mathbf{y}_i$ is the best position found so far by particle $i$ (pbest), and, $\hat{\mathbf{y}}_i$ is the neighborhood best position for particle $i$ (leader).

### 2.2 Differential Evolution

Differential evolution was proposed under the idea that a convenient source for perturbation is the population itself. Therefore, in DE, the step size is obtained from the current population. In this manner, for each parent vector $\mathbf{x}_i$, a difference vector $\mathbf{x}_{i_1} - \mathbf{x}_{i_2}$ is used to perturb another vector $\mathbf{x}_{i_3}$. This can be seen in the following mutation equation,

$$\mathbf{z}_i(t) = \mathbf{x}_{i_3} + F * (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) \tag{4}$$

where $\mathbf{x}_{i_1}$, $\mathbf{x}_{i_2}$, $\mathbf{x}_{i_3}$ are pairwise different random vectors, and $F$ is a scaling factor.

Recombination in DE is achieved by combining elements from a parent vector $\mathbf{x}_i(t)$ with elements from $\mathbf{z}_i(t)$,

$$\mu_{i,j}(t) = \begin{cases} z_{i,j}(t) \text{ if } U(0,1) < Pr \text{ or } j = r \\ x_{i,j}(t) \text{ otherwise.} \end{cases} \tag{5}$$

where $r$ is a random integer from $[1, 2, \ldots, Dim]$, $Pr$ is the recombination probability and $j$ is used as index for the $Dim$ dimensions of a solution.

## 3 Analysis of MOPSO and MODE

### 3.1 Velocity on MOPSO

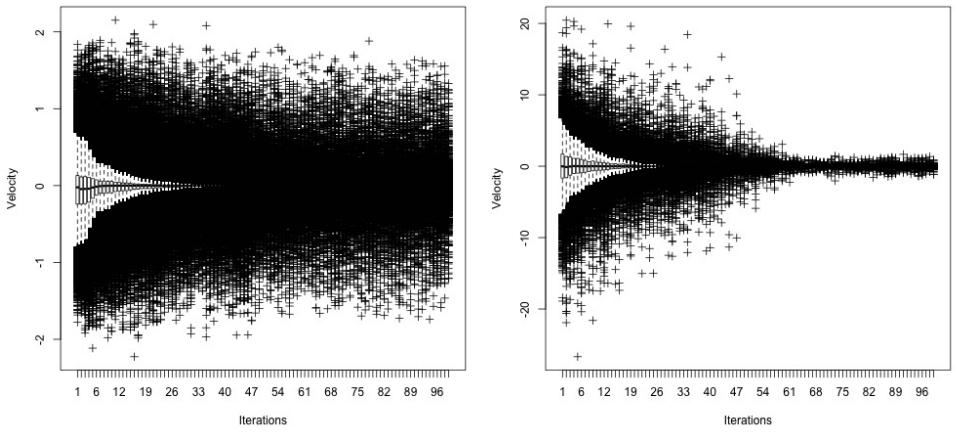
Most of the current MOPSOs have premature convergence in the presence of multifrontal problems [3]. Some researchers have analyzed the behavior of the velocity of MOPSOs in multifrontal problems and have reported that the reason for their premature convergence is that velocity values grow too much [5,3]. An experiment was developed to study the velocity of MOPSO in more detail. For this experiment, we selected two test problems: ZDT1 (which can be solved relatively easily by a MOPSO), and ZDT4 (which is a multifrontal problem that is very hard to solve for several MOPSOs). For our study, we will adopt OMOPSO[1] which obtained the best results in [3] but still was unable to solve ZDT4. In order to observe the behavior of velocity, we have used boxplots to show the values reached in $30$ executions using $50$ particles and $100$ generations (see Figures 1(a) and 1(b)). In the figures, the $y$ axis shows the velocity values, while the $x$ axis shows the iterations. Please note that the actual boxes is where the majority of observations are located while the dark region (composed of many $+$ symbols) shows the outliers. Figure 1(a) shows that the velocity values for ZDT1 are (approximately) in the range $[-2,2]$. For the case of ZDT4, Figure 1(b) shows that the velocity values are in the range $[-20,20]$. As previously noted by other researchers, the velocity values for ZDT4 are much bigger than for ZDT1. Nonetheless, it should be pointed out that the velocity values shown are proportional to the range of the decision variables of each problem [2]. Moreover, it is important to note that the velocity values in Figure 1(b) fall close to $0$ around iteration $60$. Since velocity depends on the difference of the leader and the pbest with the current particle's position, we infer that, at some point during the run, particles get really close to the leader and the pbest producing very small velocities. This premise led us to the following experiment.

### 3.2 Distance of Movement

Our hypothesis is that the observed velocity values are really a consequence, rather than a cause for the premature convergence of OMOPSO in ZDT4. To validate our hypothesis, we will now try to see why is that MODE can achieve much better results on ZDT4. In this experiment we have used the same structure of OMOPSO to design a MODE algorithm. The variant used is DE/Best/1/Bin which showed the best results in previous research. In Figures 2(a) and 2(b), we plotted the Euclidean distance traveled by particles of OMOPSO and solutions of MODE in ZDT4. For OMOPSO, this is the distance between the previous and the new position while in MODE this is the distance between the parent and the candidate position.

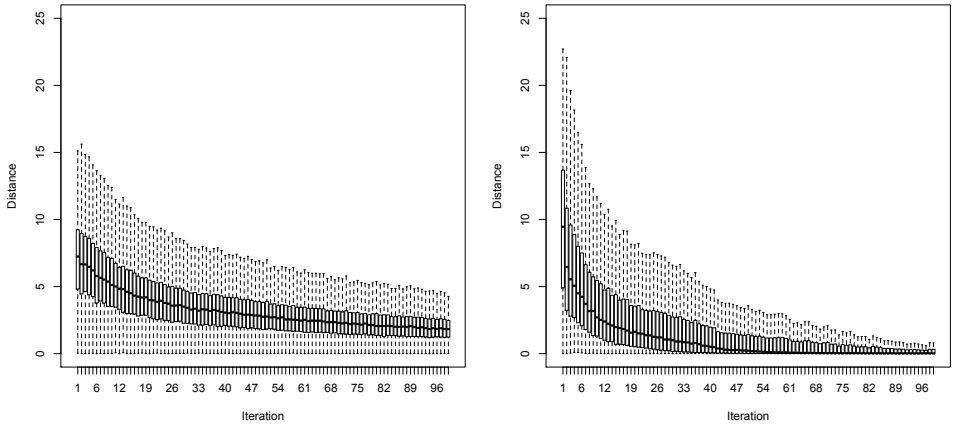
[1] We have removed the turbulence operator included in this algorithm, since we aim to observe its raw behavior.

[2] ZDT1 has its variables in the range $[0,1]$ while ZDT4 has the first variable in the range $[0,1]$ while the rest are in the range $[-5,5]$




(a) ZDT1 (b) ZDT4

**Fig. 1.** Velocity of OMOPSO for ZDT1 (left) and ZDT4 (right)




(a) MODE (b) OMOPSO

**Fig. 2.** Euclidean distance traveled from parent to offspring for MODE (left) and OMOPSO (right) on ZDT4

Figures 2(a) and 2(b) show different behaviors. OMOPSO reaches bigger distance values than MODE at the beginning of the execution but as the iterations elapse distances in OMOPSO fall rapidly to zero (meaning all particles are landing very close to its previous position). Based on our previous research

[2] and from the information seen at the presented plots, we argue that this is due to the following: OMOPSO uses only leaders and pbests to move particles while MODE uses information from the entire population. Indeed, in OMOPSO all particles are heavily attracted towards their leaders. If these leaders are diverse enough, then OMOPSO seems to move fast towards improvement. However, if leaders at some point get stuck in a local front, then all solutions will rapidly move towards that front making it harder for OMOPSO to escape (as diversity is very limited). This is not the case for MODE, in which every solution of the population can be used for perturbation and, therefore, more of the search space is explored. Moreover, solutions will land close to their parent (as the recombination operator allows certain variables to pass intact from the parent to the candidate), thus preventing that the whole population moves quickly towards local Pareto fronts.

## 4 Our Proposal

The two previous experiments led us to conjecture that, if properly integrated, a MOPSO combined with a MODE could be a very powerful yet efficient MOEA. Specifically, our aim is to use a MOPSO scheme, but to incorporate a DE mechanism that places a few particles very close to the leaders (rather than trying to strongly dominate the leaders as in MOPSO). We hypothesized that this sort of hybrid scheme might provide a fast convergence rate, while preserving the required diversity to avoid premature convergence. The next paragraphs describe our proposed approach.

### 4.1 Leader Selection Scheme

We have developed a scheme which attempts to select a leader that is diverse enough (with respect to the rest of the leaders). Our proposed mechanism works as follows. First, we obtain the centroid of all the available leaders. Then, we use roulette wheel selection to pick a leader such that the probability for each leader to be selected is proportional to the distance of that leader to the centroid (the bigger the distance of the leader to the centroid, the greater its chance of being selected). We believe that this scheme should be able to provide enough diversity in cases in which most of the leaders move towards the same region.

Here, one can argue that a leader selection scheme based on crowding would give similar results to ours. There is, however, an important difference. Crowding will favor both boundaries of the search space at all times, whereas our proposed scheme will favor the boundary opposite to the location of most of the leaders (see Figure 3).

### 4.2 The Use of the Velocity

It has been a common practice in PSO to decrease the previous velocity using a factor $w$. Moreover, this factor has traditionally been set using one of three

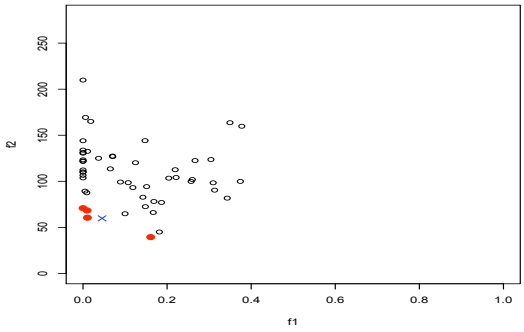



**Fig. 3.** Leader selection scheme for our proposal. Leaders are represented by filled red dots, solutions are black circles and the "$X$" symbol is the centroid of the leaders

schemes: $i$) $w$ adopts a constant value, $ii$) $w$ depends on the current iteration, and $iii$) $w$ is randomly selected from a range. Here, we have adopted a different scheme in which $w$ depends on the previous success of the particle. We believe that if a particle succeeded at previous iterations, then it should be beneficial to use a bigger portion of its previous velocity. On the contrary, if the particle has not been successful, then the previous velocity portion should be decreased giving a higher chance to the social and cognitive factors to take action. In short, Equation 6 works in the following way[3]. If a particle is the selected leader (meaning the particle has found a very good position at its previous iteration), then we use all of its previous velocity ($w = 1$). If a particle's current position is its pbest (the particle found its best position of the whole execution in the previous iteration), then we use a high range of its previous velocity ($w = U \sim (0.5, 1.0)$). In none of the two previous cases occur, then we use a lower percentage of the velocity ($w = U \sim (0.1, 0.5)$).

$$w = \begin{cases} 1 & \text{if } \mathbf{x}_i(t) = \hat{\mathbf{y}}_i(t) \text{ and } flip(0.5) = true \\ U \sim (0.5, 1.0) & \text{if } \mathbf{x}_i(t) = \mathbf{y}_i(t) \text{ and } flip(0.5) = true \\ U \sim (0.1, 0.5) & \text{otherwise.} \end{cases} \quad (6)$$

### 4.3 Moving Towards an Specific Leader

We adopt a DE mechanism to place some particles close to the selected leader rather than finding a position that strongly dominates the leader but might be moving "deeper" into some local optima. Thus, using a low probability we adopt a mechanism which will make several variables (with a high probability) to be equal to the selected leader. The aim is that the obtained particle will be close to the selected leader and that region of the search space is not lost.

[3] $flip(0.5)$ refers to a function which returns true with 50% probability

Moreover, a few variables will be obtained from a mutation based on three different random pbests. Equation (7) describes this mechanism.

$$\mathbf{x}_{i,j}(t) = \begin{cases} \mathbf{y}_{i_3,j} + F * (\mathbf{y}_{i_1,j} - \mathbf{y}_{i_2,j}) & \text{if } U(0,1) < Pr \text{ or j} = \text{r} \\ \hat{\mathbf{y}}_{p,j} & \text{otherwise.} \end{cases} \quad (7)$$

### 4.4 The Algorithm of Our Proposed Approach

Algorithm 1 describes our proposal, called **Multi-Objective Particle Swarm Optimizer Enhanced with a Differential Evolution Scheme** (***MOPEDS***).

**Algorithm 1** Proposed algorithm (MOPEDS)

```
Initialize Population
Find non-dominated solutions (Leaders)
g = 0
while g < gMax do
    for each Particle i do
        Select leader (from non-dominated solutions) using mechanism from Section 4.1
        if U ~ (0, 1) > Pm then
            Select a w value using mechanism from Section 4.2
            Update velocity
            Update position
        else
            Select three random (different) pbest
            Move particle using DE scheme from Section 4.3
        end if
        Evaluate particle i
        if Particle i position dominates its pbest then
            update pbest to current position
        end if
    end for
    Update non-dominated solutions (Leaders)
end while
```

## 5 Experimental Study

Next, we compare our proposal with four state-of-the-art MOEAs: OMOPSO[4] [3], SMPSO [5], NSGA-II[5] [1], and DEMO [6] [8]. The following parameters were adopted. For NSGA-II: $0.9$ for the crossover rate, $1/Dim$ for the mutation rate, $15$ for the distribution index for crossover, and $20$ for the distribution index for mutation. DEMO uses $Pr = 0.3$, and $F = 0.5$. OMOPSO uses $C1, C2 = U \sim (1.5, 2.0)$, and $w = U \sim (0.1, 0.5)$. Finally, SMPSO uses $C1, C2 = U \sim (1.5, 2.5)$, and $w = U \sim (0.1, 0.5)$. MOPEDS adopted $Pm = 0.2$, $F = G(0.5, 0.5)$, $C1, C2 = U \sim (1.2, 2.0)$ and $Pr = 0.2$, since these parameters provided the best behavior in our preliminary tests. The $w$ parameter is used as described in Section 4.2. We have measured all algorithms with the IGD performance measure using $2000$, $3500$, and $5000$ objective function evaluations in order to obtain more detailed information about their performance. There is one plot for each problem

[4] The implementation of OMOPSO used in our experiments differs from its original proposal [7] in that $\epsilon$-dominance is not adopted.

[5] We took the code available at: `http://www.iitk.ac.in/kangal/codes.shtml`

[6] Please do not confuse DEMO with MODE. DEMO is an specific implementation of MODE which refers to multi-objective differential evolution.

where each algorithm is presented at the three mentioned function evaluation numbers (see Figure 4).

**ZDT1** - In this problem it can be observed that MOPEDS is ahead of all the other algorithms at the $2000$ function evaluations. This shows the fast convergence rate of our proposal on this problem. Moreover, at the $3500$ evaluations MOPEDS is still ahead and DEMO is just a little behind. Finally, when reaching the $5000$ evaluations MOPEDS and DEMO show IGD values very close to $0$ (the ideal value).

**ZDT2** - Again, MOPEDS has the best performance at $2000$ function evaluations. In fact, MOPEDS is capable of obtaining considerable advantage over the rest of the algorithms in this number of evaluations. At the $3500$ evaluations, IGD values for MOPEDS are already very close to $0$ while OMOPSO and DEMO are a little behind. Finally, at the $5000$ function evaluations, OMOPSO and DEMO have reached values as good as MOPEDS.

**ZDT3** - MOPEDS shows the best IGD values at $2000$ and $3500$ evaluations. Nonetheless, at $3500$ evaluations DEMO is very competitive also. Furthermore, at the $5000$ evaluations MOPEDS and DEMO have a very similar performance.

**ZDT4** - In this problem, it is clear the SMPSO shows much better results than any other algorithm, while OMOPSO shows the poorest. Acknowledging this, we will omit OMOPSO and SMPSO from the discussion of this problem. At $2000$ evaluations, our proposal is ahead of DEMO and a little behind NSGA-II. At $3500$ evaluations, MOPEDS and NSGA-II show similar results. Nonetheless, the box is bigger for MOPEDS indicating a bigger dispersion than NSGA-II. Finally, at $5000$ evaluations, our proposal is again competitive with NSGA-II.

**ZDT6** - It is clear that all algorithms (except one) get close to the true Pareto front very early in the search (only $2000$ function evaluations). In this problem NSGA-II takes a bigger number of function evaluations to reach the true Pareto front.

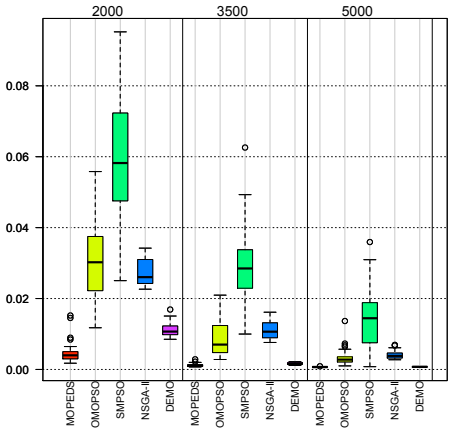
Since SMPSO achieved such excellent results in ZDT4, we decided to analyze this algorithm in more detail. SMPSO is actually a modification of OMOPSO in which a velocity constriction mechanism is used[7]. Basically, this constriction factor limits the values that the velocity can take before moving a particle. The velocity is limited using Equations (8) and (9):

$$v_{i,j} = \begin{cases} delta & \text{if } v_{i,j} > delta_j \\ -delta_j & \text{if } v_{i,j} \leq -delta_j \\ v_{i_j} & \text{otherwise.} \end{cases} \tag{8}$$

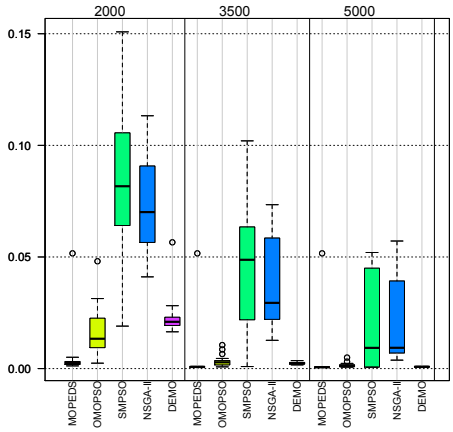
$$delta_j = \frac{(upper\_limit_j - lower\_limit_j)}{2} \tag{9}$$

This constriction factor is the main difference with respect to the original OMOPSO. After analyzing the behavior of this approach in ZDT4, we reached the following conclusions. For ZDT4 (from the second to the tenth variables) $\delta = 5$ and $-\delta = -5$. Moreover, SMPSO (as many other MOEAs) truncates variables to their upper and lower limits when these go beyond their predefined

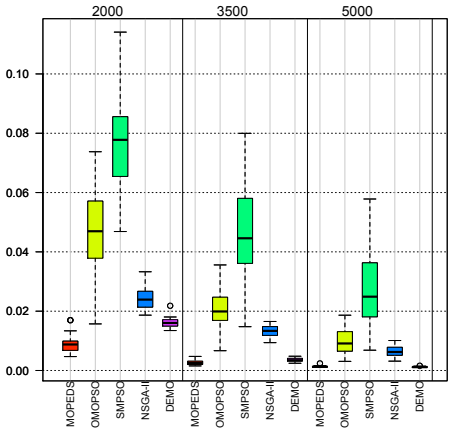
[7] Please refer to [5] for further details on this algorithm



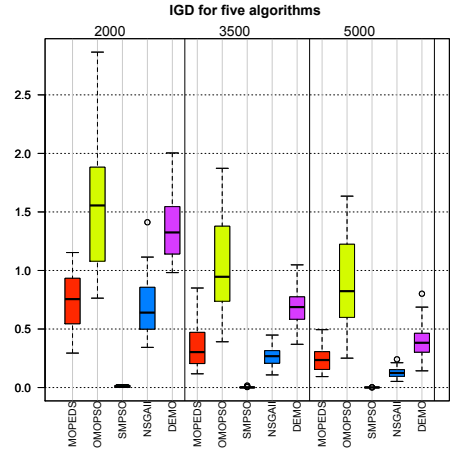

(a) Five algorithms on ZDT1



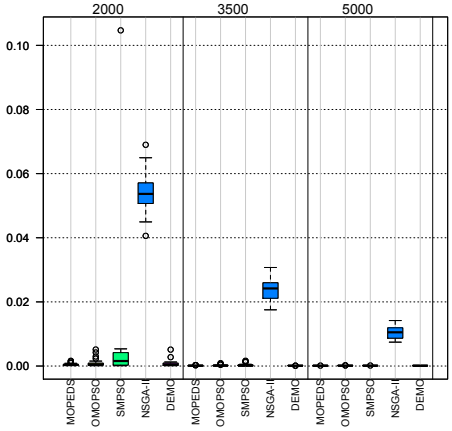

(b) Five algorithms on ZDT2




(c) Five algorithms on ZDT3




(d) Five algorithms on ZDT4




(e) Five algorithms on ZDT6

**Fig. 4.** Comparison of 5 algorithms using the IGD performance measure at 2000, 3500, and 5000 function evaluations in the ZDT test suite.

bounds. For ZDT4, these bounds are $-5$ and $5$. Therefore, if a variable at iteration $t$ lands above its upper limit, this variable will be truncated to $5$. Then, at iteration $t+1$, if it happens that the velocity goes below $-5$ (which is $-\delta$), this velocity will be truncated to $-\delta = -5$. Therefore, when we add the velocity to the current particle's position $5 + (-5)$, we end up with $0$ which is precisely in the region where the Pareto optimal set for this test problem is located. Even when this is a very clever mechanism and works perfectly in ZDT4, we decided to observe the robustness of SMPSO when the ranges of ZDT4 are changed. Thus, we moved the lower limit (again from the second to the tenth variable) to $-2$. The upper limit was not changed. Moreover, we also tested our proposal using these modifications with ZDT4. Both algorithm were run using $25000$ function evaluations ($100$ particles and $250$ iterations). Results are shown in Figure 5.

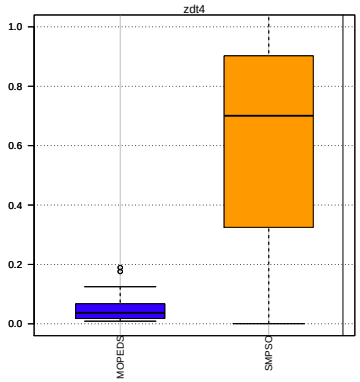



**Fig. 5.** MOPEDS and SMPSO at a modified version of ZDT4, using $25,000$ evaluations

From Figure 5, we can observe that the performance of SMPSO is clearly deteriorated when changing the ranges of the variables for the ZDT4 problem. We can see that SMPSO works very well some times while reporting poor results at other executions. Regarding our proposal, we observe that the modification did not have a significant impact on its performance. In fact, some improvements were achieved with regards to its execution at $2000$, $3500$, and $5000$ evaluations. It is important to note, however, that our proposal was never able to reach the true Pareto front but could only closely approximate it.

## 6 Conclusions and Future Work

In conclusion, our proposal shows competitive results when compared with other state of the art MOEAs using a small number of function evaluations. Moreover, it is important to note that our proposal is able to reach competitive results in the multifrontal problem ZDT4 which has found to be quite difficult for most current MOPSOs. Therefore, we believe this indicates that MOPEDS is benefitting from the high convergence rate of MOPSO while maintaining diversity using the DE scheme. It is important to note, however, that the number of

evaluations adopted (except for $25000$) are relatively small and, therefore, the plots presented here do not give further information about the performance of MOPEDS if we extend its execution. This seems important since our algorithm was not able to reach the true Pareto front at $25000$ evaluations. This is certainly an issue that deserves some further work. Finally, we also believe that further investigation on mechanisms to fine tune the parameters of our proposed approach are a promising research path.

## Acknowledgements

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-Tamaulipas. The second author gratefully acknowledges support from CONACyT through project 105060. The third author acknowledges support from CONACyT project no. 103570

## References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (April 2002)
2. Dominguez, J.S.H., Pulido, G.T.: A comparison on the search of particle swarm optimization and differential evolution on multi-objective optimization. In: Evolutionary Computation (CEC), 2011 IEEE Congress on. pp. 1978 –1985 (june 2011)
3. Durillo, J.J., García-Nieto, J., Nebro, A.J., Coello Coello, C.A., Luna, F., Alba, E.: Multi-Objective Particle Swarm Optimizers: An Experimental Comparison. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.K., Sevaux, M. (eds.) Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009, pp. 495–509. Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France (April 2009)
4. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948 (1995)
5. Nebro, A.J., Durillo, J.J., Garcia-Nieto, J., Coello Coello, C.A., Luna, F., Alba, E.: SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In: 2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM'2009). pp. 66–73. IEEE Press, Nashville, TN, USA (March 30 - April 2 2009), iSBN 978-1-4244-2764-2
6. Price, K., Storn, R.: Differential Evolution - a simple evolution strategy for fast optimization (April 1997)
7. Reyes Sierra, M., Coello Coello, C.A.: Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and $\epsilon$-Dominance. In: Coello Coello, C.A., Aguirre, A.H., Zitzler, E. (eds.) Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005. pp. 505–519. Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México (March 2005)
8. Robič, T., Filipič, B.: DEMO: Differential Evolution for Multiobjective Optimization. In: Coello Coello, C.A., Aguirre, A.H., Zitzler, E. (eds.) Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005. pp. 520–533. Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México (March 2005)
9. Veldhuizen, D.A.V., Lamont, G.B.: On Measuring Multiobjective Evolutionary Algorithm Performance. In: 2000 Congress on Evolutionary Computation. vol. 1, pp. 204–211. IEEE Service Center, Piscataway, New Jersey (July 2000)