A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization

Margarita Reyes-Sierra and Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Ingeniería Eléctrica, Sección Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07360, México
mreyes@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Abstract- In this paper, we study the use of fitness inheritance and approximation techniques to reduce the number of fitness evaluations into a PSO-based multi-objective algorithm previously proposed by the authors. Fifteen fitness inheritance techniques and four approximation techniques are applied to a set of four well-known test functions taken from the multi-objective optimization literature. A comparison of the best techniques found against other PSO-based multi-objective approaches is carried out using other test functions. The obtained results show a good performance of the enhancement techniques proposed.

1 Introduction

Given the high computational cost of fitness evaluations in many real-world applications, the use of Evolutionary Algorithms (EAs) as population-based techniques tends to become expensive. In order to improve the performance of EAs, several enhancement techniques have been proposed. Fitness Inheritance is an enhancement technique [10] in which the fitness value of an offspring is obtained from the fitness values of its parents. On the other hand, approximation techniques [5] let us estimate the fitness of an individual using the previously calculated fitness of its neighbors. In fact, fitness inheritance is a particular case of fitness approximation. In general, by using enhancement techniques, we do not need to evaluate every individual at each generation, and the computational cost is reduced. In this paper, we perform a study of different inheritance and approximation techniques applied to a real-coded Particle Swarm Optimizer (PSO) that has been previously proposed by the authors to solve multi-objective problems [9]. Since our major interest focus is on fitness inheritance techniques, we are proposing fifteen inheritance techniques and four approximation techniques. In our study, we use four well-known multi-objective test functions in order to find the best from the proposed techniques. Then, the best techniques found are compared against other PSO-based multi-objective optimizers representative of the state-of-the-art, using different test functions. This paper is organized as follows. A brief introduction to Fitness Inheritance and Fitness Approximation is given in Sections 2 and 3, respectively. Section 4 introduces the multi-objective PSO-based algorithm in which the proposed techniques are incorporated. The enhancement techniques proposed in this paper are presented in Section 5. In Section 6 and 7 we present the obtained results and their discussion, respectively. A comparison against other PSO-based algorithms is presented in Section 8. Finally, the conclusions and future work are described in Section 9.

2 Fitness Inheritance

The use of fitness inheritance to improve the performance of GAs was originally proposed by Smith et al. [10]. The authors proposed two possible ways of inheriting fitness: the first consists of taking the average fitnesses of the two parents and the other consists of taking a weighted (proportional) average of the fitnesses of the two parents. The second approach is related to how similar the offspring is with respect to its parents (this is done using a similarity measure). They applied inheritance to a very simple problem (the OneMax problem) [10] and found that the weighted fitness average resulted in a better performance and indicated that fitness inheritance was a viable alternative to reduce the computational cost of a genetic algorithm. In a previous work [8], we proposed the first attempt to incorporate the concept of fitness inheritance to a real-coded Multi-Objective PSO (MOPSO) previously proposed by us [9]. In [8], we tested the performance of weighted average fitness inheritance on a well-known test suite of multi-objective optimization problems ([12]). Based on the obtained results, we conclude that fitness inheritance reduces the computational cost without decreasing the quality of the results in a significant way. Also, the fitness inheritance technique used was able to generate non-convex and discontinuous Pareto fronts. These conclusions were somewhat surprising since, previous to our work, Ducheyne et al. [4] tested the performance of average and weighted average fitness inheritance on the same test suite, using a binary GA, and they concluded that although fitness inheritance efficiency enhancement techniques could be used to reduce the number of fitness evaluations, they found that if the Pareto surface was not convex or if it was discontinuous, the fitness inheritance strategies failed to reach the true Pareto front.

3 Fitness Approximation

Another promising possibility when an evaluation is very time consuming or expensive is not to evaluate every individual, but just estimate the quality of some of the individuals based on an approximate model of the fitness landscape. Approximation techniques estimate individual fitness on the basis of previously observed objective function values of neighboring individuals. There are many possible approximation models. In the simplest case, the fitness of a new individual is derived from its parents' fitnesses (fitness inheritance). However, there are some other methods like polyno-

Begin
Initialize swarm. Initialize leaders.
Send leaders to ϵ-archive
crowding(leaders), $g = 0$
While $g < gmax$
For each particle
Select leader. Flight. Mutation.
$\Rightarrow$ **If**(p_i) Inherit **Else** Evaluation.
Update $pbest$.
EndFor
Update leaders, Send leaders to ϵ-archive
crowding(leaders), g++
EndWhile
Report results in ϵ-archive
End

Figure 1: Pseudocode of our algorithm.

mials, the kriging model, neural networks [5] and interpolation and regression [1]. Reported experiments [1] show that using fitness estimation, it is possible to either reach a better fitness level in a certain given time, or to reach a desired fitness level much faster. In this paper, we adopt very simple approximation techniques, based only on the objective values of the closest neighbors.

4 Multi-Objective Particle Swarm Optimization

In this paper, we incorporate several fitness inheritance and approximation techniques into a MOPSO that was previously proposed by us in [9] and updated in [8]. The MOPSO proposed in [9, 8] is based on Pareto dominance, since it considers every non-dominated solution as a new leader. Additionally, the approach also uses a crowding factor [2] as a second discrimination criterion which is also adopted to filter out the list of available leaders. For each particle, we select the leader in the following way: 97% of the time a leader is selected, randomly, if and only if that leader dominates the current particle, and, the remaining 3% of the time, we select a leader by means of a binary tournament based on the crowding value of the available set of leaders. 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. We also proposed the use of different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. We proposed a scheme by which the swarm is subdivided in three parts (of equal size): the first sub-part has no mutation at all, the second sub-part uses uniform mutation and the third sub-part uses non-uniform mutation. The available set of leaders is the same for each of these sub-parts. Finally, the proposed approach also incorporates the ϵ-dominance concept [6] to fix the size of the set of final solutions produced by the algorithm. Figure 1 shows the pseudo-code of our proposed approach.

In Figure 1, the symbol ($\Rightarrow$) indicates the line in which the concept of fitness inheritance (or approximation) is incorporated. The *inheritance* or *approximation proportion*, p_i, is the proportion of individuals in the population whose fitness is inherited or approximated. It is very important to note that a particle that has inherited its objective values **can not** enter into the final Pareto front, since a final solution must have true objective values.

5 Proposed Techniques

5.1 Fitness Inheritance

Since PSO has no recombination operator, we adopted as "parents" of a particle the previous position of the particle, its $pbest$ and its leader.

5.1.1 Linear Combination Based on Distances (LCBD)

We propose to calculate the new position in the objective space of a particle by means of a linear combination of the positions of the particles that were considered to calculate the new position in the search space. We consider the position of the leader as the most important. Thus, the leader will be always considered.

Given a particle x_{old}, its personal best x_{pbest}, its assigned leader x_{ld} and the new particle x_{new}, we proceed to calculate the distance from x_{new} to its "parents" (as defined before): $d_1 = d(x_{new}, x_{old}), d_2 = d(x_{new}, x_{pbest}), d_3 = d(x_{new}, x_{ld})$, where d is an Euclidean distance. We propose variants of the same idea, based on the individuals that can be considered:

FI1 Previous position and leader: $r = \frac{d_1}{d_1+d_2}$,

$$f_i(x_{new}) = rf_i(x_{ld}) + (1-r)f_i(x_{old}), i = 1, ..., n.$$

FI2 $pbest$ and leader. $r = \frac{d_2}{d_2+d_3}$,

$$f_i(x_{new}) = rf_i(x_{ld}) + (1-r)f_i(x_{pbest}), i = 1, ..., n.$$

FI3 Previous position, $pbest$ and leader.
$r_1 = \frac{d_1}{d_1+d_2+d_3}$, $r_2 = \frac{d_2}{d_1+d_2+d_3}$, $r_3 = \frac{d_3}{d_1+d_2+d_3}$,
$r_1 = 1/r_1, r_2 = 1/r_2, r_3 = 1/r_3$

$$f_i(x_{new}) = r_1 f_i(x_{old}) + r_2 f_i(x_{pbest}) + r_3 f_i(x_{ld}),$$

$i = 1, ..., n$. Where f_i is the value of the objective function i and n is the number of objective functions. See Figure 2 for an illustration of these techniques.

The technique FI1 is the one proposed in [8]. As in [8], in all the inheritance techniques, if the leader selected does not dominate the current particle, we will proceed to calculate the inherited position and to assign the objective values of the closest leader to that position. This procedure is used to avoid the generation of invalid particles in the case of non-convex Pareto fronts. See Figure 3.

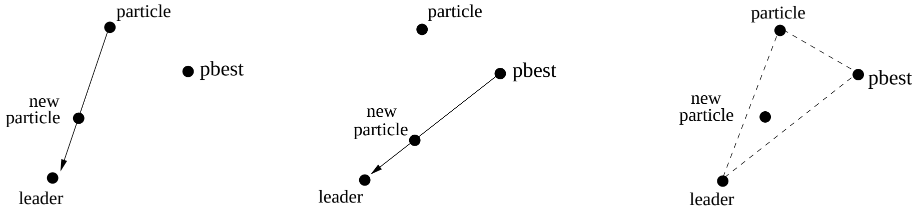
5.1.2 Flight Formula on Objective Space (FFOS)

As we know, in PSO, the position of each particle in the search space is updated using the formula:

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

$$\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1 r_1(\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 r_2(\vec{x}_{gbest_i} - \vec{x}_i(t))$$

In this case, we propose the analogous formula to update



FI1 **FI2** **FI3**

Figure 2: Illustration of techniques FI1, FI2 and FI3.

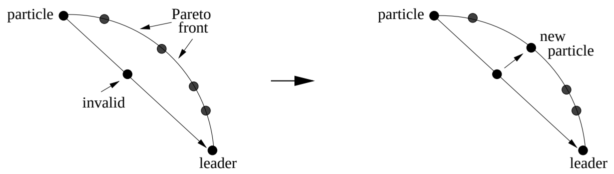


Figure 3: Case in which "invalid" particles can be obtained and the method used to repair them.

the position of each particle in the objective space:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \vec{\mathbf{vf}}_{\mathbf{i}}(t)$$

$$\vec{\mathbf{vf}}_{\mathbf{i}}(t) = W\vec{\mathbf{vf}}_{\mathbf{i}}(t-1) + C_1 r_1(\vec{\mathbf{f}}_{\mathbf{pbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t)) + C_2 r_2(\vec{\mathbf{f}}_{\mathbf{gbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t))$$

where $\vec{\mathbf{f}}_{\mathbf{i}}$, $\vec{\mathbf{f}}_{\mathbf{pbest_i}}$ and $\vec{\mathbf{f}}_{\mathbf{gbest_i}}$ are the values of the objective function i for the current particle, its $pbest$ and $gbest$, respectively. We use the same values of W, C_1, r_1, C_2 and r_2 previously used for the flight in the decision variable space. We will consider the following variants based on the vectors considered:

FI4 Considering the whole formula:

$$\vec{\mathbf{vf}}_{\mathbf{i}}(t) = W\vec{\mathbf{vf}}_{\mathbf{i}}(t-1) + C_1 r_1(\vec{\mathbf{f}}_{\mathbf{pbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t)) + C_2 r_2(\vec{\mathbf{f}}_{\mathbf{gbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t))$$

FI5 Ignoring the previous direction:

$$\vec{\mathbf{vf}}_{\mathbf{i}}(t) = C_1 r_1(\vec{\mathbf{f}}_{\mathbf{pbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t)) + C_2 r_2(\vec{\mathbf{f}}_{\mathbf{gbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t))$$

FI6 Ignoring the direction to the $pbest$:

$$\vec{\mathbf{vf}}_{\mathbf{i}}(t) = W\vec{\mathbf{vf}}_{\mathbf{i}}(t-1) + C_2 r_2(\vec{\mathbf{f}}_{\mathbf{gbest_i}} - \vec{\mathbf{f}}_{\mathbf{i}}(t))$$

5.1.3 Combination Using Flight Factors

Non-linear Combination (NLC)

In this case, we propose to calculate the new objective position of a particle using the elements of the flight formula: $\vec{\mathbf{f}}_{\mathbf{i}}(t) = W\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + C_1 r_1 \vec{\mathbf{f}}_{\mathbf{pbest_i}} + C_2 r_2 \vec{\mathbf{f}}_{\mathbf{gbest_i}}$

As in the previous cases, the variants considered are:

FI7 Considering the whole formula:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = W\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + C_1 r_1 \vec{\mathbf{f}}_{\mathbf{pbest_i}} + C_2 r_2 \vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI8 Ignoring the previous position:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = C_1 r_1 \vec{\mathbf{f}}_{\mathbf{pbest_i}} + C_2 r_2 \vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI9 Ignoring the position of the $pbest$:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = W\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + C_2 r_2 \vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

On the other hand, since $W \in (0.1, 0.5)$ and $C_1 r_1$, $C_2 r_2 \in (0.0, 2.0)$, we propose to modify the previous formula in the following way:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{0.5}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_1 r_1}{2.0}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{2.0}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

As a result, we obtain the following variants:

FI10 Considering the whole formula:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{0.5}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_1 r_1}{2.0}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{2.0}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI11 Ignoring the previous position:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{C_1 r_1}{2.0}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{2.0}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI12 Ignoring the position of the $pbest$:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{0.5}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_2 r_2}{2.0}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

Linear Combination (LC)

We propose to use the previous formula but in such a way that the result is a linear combination of the elements considered:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{r}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_1 r_1}{r}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{r}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

where $r = W + C_1 r_1 + C_2 r_2$. The corresponding variants are the following (note the changes in r):

FI13 Considering the whole formula, $r = W + C_1 r_1 + C_2 r_2$:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{r}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_1 r_1}{r}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{r}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI14 Ignoring the previous position, $r = C_1 r_1 + C_2 r_2$:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{C_1 r_1}{r}\vec{\mathbf{f}}_{\mathbf{pbest_i}} + \frac{C_2 r_2}{r}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

FI15 Ignoring the position of the $pbest$, $r = W + C_2 r_2$:

$$\vec{\mathbf{f}}_{\mathbf{i}}(t) = \frac{W}{r}\vec{\mathbf{f}}_{\mathbf{i}}(t-1) + \frac{C_2 r_2}{r}\vec{\mathbf{f}}_{\mathbf{gbest_i}}$$

5.2 Fitness Approximation (FA)

We propose four simple approximation techniques. In each case, the particle will take the objective values of the particle indicated:

FA1 The closest particle: leader or member of the swarm.

FA2 The closest leader.

FA3 The closest particle (member of the swarm).

FA4 The average of the 10 closest particles (leaders or members of the swarm).

We use the Euclidean distance in the decision variable space. In technique FA4, there are cases in which an invalid particle may be created. In this way, if among the 10 closest particles there are two or more leaders, or there is just one leader but this leader does not dominate the current particle, we will proceed as it was explained before. See Figure 3.

6 Comparison of Results

In order to compare the proposed techniques, we performed a study using four well-known test functions proposed in [12]: ZDT1, ZDT2, ZDT3 and ZDT4. The functions ZDT1, ZDT2 and ZDT3 have 30 variables and the function ZDT4 has 10 variables. The four functions have two objectives. Functions ZDT1 and ZDT4 have convex Pareto fronts, ZDT2 has a non-convex Pareto front and ZDT3 has a non-convex and discontinuous Pareto front. We performed experiments with different values of *inheritance (approximation) proportion* p_i. We experimented with: p_i= 0.1, 0.2, 0.3, 0.4. Note that this proportion of individuals indicates also the percentage by which the number of evaluations is reduced (e.g., $p_i = 0.1$ means that 10% less evaluations are performed). We performed 20 runs for each function and each technique. The parameters adopted for our MOPSO were: 100 particles, 200 generations and 100 particles in the external archive. We used several performance measures to validate the obtained results in a quantitative way. However, because of space reasons, we will only show the results corresponding to the *Success Counting* measure since this was the one that reflected in a better way the quality differences between the approaches compared. This measure gives the number of particles from the obtained Pareto front that belong to the true Pareto front. Tables 1, 2, 3, 4, 5, 6 and 7 present a summary of the results obtained. In each case, we present the average of the Success Counting measure over the 20 runs, and the percentage of decrement or increment on the quality of the results. Also, we present the average of the percentages for each value of inheritance proportion, for each technique.

7 Discussion of Results

Since comparing 19 different techniques is very difficult, we decided to represent each technique with a vector. The vector used is that containing the average of the change in the quality of results for each inheritance proportion value. For example, to represent technique FI1, we construct the following vector (see Table 1):

Inheritance proportion p_i	0.1	0.2	0.3	0.4
Average vector	2.6	-4.1	-13.7	-14.0

In this way, in Table 8 we present the vectors of all techniques. Since every entry in each vector is a change in the quality of the obtained results given a value of inheritance proportion, the bigger the values of the vector, the better the corresponding technique is. Thus, we are interested on the vector or vectors that represent the solution to the problem of maximizing all the entries (i.e. each entry is considered as an objective). The non-dominated vectors among all the 19 techniques are the vectors corresponding to techniques FI5 and FA3. That is, the techniques FI5 and FA3 are the best. For this reason these two techniques are marked with a level of 1 in Table 8. FI5 is an inheritance technique and FA3 is an approximation technique. For these two techniques, in the worst case, the decrement in quality of results is no more than 13%, even when a 40% of the total number of evaluations is saved. After eliminating techniques FI5 and FA3, we proceed again to locate the non-dominated vectors. In this case, the best techniques, marked with a level 2 are: FI1, FI2, FI4, FI6 and FA1. This leads us to conclude that, in general, the set of inheritance techniques based on the flight formula on the objective space (FFOS) are the best.

Group		0.1	0.2	0.3	0.4	level
LCBD	FI1	2.6	-4.1	-13.7	-14.0	**2**
	FI2	-3.6	-2.4	-11.9	-12.9	**2**
	FI3	0.1	-4.9	-13.8	-17.8	
FFOS	FI4	0.1	-1.7	-8.7	-13.6	**2**
	FI5	**4.7**	**-1.2**	**-8.1**	**-11.7**	**1**
	FI6	1.6	-2.8	-10.1	-16.7	**2**
NLC	FI7	-4.9	-10.3	-19.5	-30.2	
	FI8	-0.7	-7.5	-20.7	-29.8	
	FI9	-0.2	-7.3	-16.7	-28.5	
	FI10	-3.0	-9.2	-19.3	-33.3	
	FI11	-3.6	-6.0	-14.1	-26.7	
	FI12	-3.0	-9.5	-17.5	-22.1	
LC	FI13	-2.1	-2.6	-12.5	-18.8	
	FI14	-3.7	-4.9	-10.3	-16.0	
	FI15	0.3	-5.0	-12.3	-16.6	
FA	FA1	4.2	-3.4	-8.4	-14.1	**2**
	FA2	-0.3	-11.2	-16.6	-15.9	
	FA3	**1.5**	**0.4**	**-6.9**	**-12.9**	**1**
	FA4	0.3	-4.1	-12.3	-16.2	

Table 8: Vectors of change in quality for each technique, for each value of inheritance or approximation proportion.

8 Comparison with other PSO approaches

In the previous section, we found two enhancement techniques to be the best from the set proposed: one of fitness inheritance and one of fitness approximation. In this section, these two techniques will be compared against other two PSO-based multi-objective approaches representative of the state-of-the-art: the Sigma-MOPSO [7] and the Cluster-MOPSO [11]. For this comparison we will use two different test functions: DTLZ2 and DTLZ6 [3]. Both functions have 3 objectives. DTLZ2 has 12 variables and DTLZ6 has 22 variables. As in previous experiments, we used different values of p_i. We performed 20 runs for each function and each approach. The approaches without fitness inheritance or approximation performed 20000 objective function evaluations. The parameters adopted for our MOPSO were the same as before. 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 (these values were suggested by the author of the method). The PSO approaches will be identified with the following la-

FI1	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	77	(+8.5%)	64	(-9.9%)	62	(-12.7%)	61	(-14.1%)
ZDT2	89	83	(-6.7%)	86	(-3.4%)	79	(-11.2%)	77	(-13.5%)
ZDT3	68	73	(+7.4%)	65	(-4.4%)	64	(-5.9%)	59	(-13.2%)
ZDT4	80	81	(+1.3%)	81	(+1.3%)	60	(-25.0%)	68	(-15.0%)
Average			**+2.6%**		**-4.1 %**		**-13.7 %**		**-14.0 %**
FI2	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	68	(-4.2%)	68	(-4.2%)	59	(-16.9%)
ZDT2	89	81	(-9.0%)	82	(-7.9%)	78	(-12.4%)	77	(-13.5%)
ZDT3	68	64	(-5.9%)	67	(-1.5%)	58	(-14.7%)	63	(-7.4%)
ZDT4	80	77	(-3.8%)	83	(+3.8%)	67	(-16.3%)	69	(-13.8%)
Average			**-3.6%**		**-2.4 %**		**-11.9 %**		**-12.9 %**
FI3	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	73	(+2.8%)	69	(-2.8%)	69	(-2.8%)	50	(-29.6%)
ZDT2	89	87	(-2.2%)	82	(-7.9%)	71	(-20.2%)	76	(-14.6%)
ZDT3	68	67	(-1.5%)	63	(-7.4%)	64	(-5.9%)	60	(-11.8%)
ZDT4	80	81	(+1.3%)	79	(-1.3%)	59	(-26.3%)	68	(-15.0%)
Average			**+0.1%**		**-4.9 %**		**-13.8 %**		**-17.8%**

Table 1: Obtained results for different values of inheritance proportion, for techniques FI1, FI2 and FI3.

FI4	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	62	(-12.7%)	62	(-12.7%)	59	(-16.9%)	49	(-31.0%)
ZDT2	89	85	(-4.5%)	84	(-5.6%)	78	(-12.4%)	79	(-11.2%)
ZDT3	68	73	(+7.4%)	69	(+1.5%)	60	(-11.8%)	58	(-14.7%)
ZDT4	80	88	(+10.0%)	88	(+10.0%)	85	(+6.3%)	82	(+2.5%)
Average			**+0.1%**		**-1.7 %**		**-8.7 %**		**-13.6%**
FI5	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	69	(-2.8%)	61	(-14.1%)	56	(-21.1%)
ZDT2	89	89	(0.0%)	79	(-11.2%)	84	(-5.6%)	77	(-13.5%)
ZDT3	68	72	(+5.9%)	70	(+2.9%)	55	(-19.1%)	58	(-14.7%)
ZDT4	80	87	(+8.8%)	85	(+6.3%)	85	(+6.3%)	82	(+2.5%)
Average			**+4.7%**		**-1.2 %**		**-8.1 %**		**-11.7%**
FI6	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	70	(-1.4%)	61	(-14.1%)	62	(-12.7%)	47	(-33.8%)
ZDT2	89	83	(-6.7%)	82	(-7.9%)	76	(-14.6%)	70	(-21.3%)
ZDT3	68	72	(+5.9%)	72	(+5.9%)	59	(-13.2%)	61	(-10.3%)
ZDT4	80	83	(+3.8%)	84	(+5.0%)	80	(0.0%)	79	(-1.3%)
Average			**+1.6%**		**-2.8 %**		**-10.1 %**		**-16.7%**

Table 2: Obtained results for different values of inheritance proportion, for techniques FI4, FI5 and FI6.

FI7	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	64	(-9.9%)	58	(-18.3%)	57	(-19.7%)	47	(-33.8%)
ZDT2	89	83	(-6.7%)	74	(-16.9%)	68	(-23.6%)	66	(-25.8%)
ZDT3	68	66	(-2.9%)	69	(+1.5%)	64	(-5.9%)	57	(-16.2%)
ZDT4	80	80	(0.0%)	74	(-7.5%)	57	(-28.8%)	44	(-45.0%)
Average			**-4.9%**		**-10.3 %**		**-19.5 %**		**-30.2%**
FI8	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	62	(-12.7%)	53	(-25.4%)	47	(-33.8%)
ZDT2	89	85	(-4.5%)	84	(-5.6%)	66	(-25.8%)	65	(-27.0%)
ZDT3	68	71	(+4.4%)	67	(-1.5%)	61	(-10.3%)	52	(-23.5%)
ZDT4	80	80	(0.0%)	72	(-10.0%)	63	(-21.3%)	52	(-35.0%)
Average			**-0.7%**		**-7.5 %**		**-20.7 %**		**-29.8%**
FI9	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	67	(-5.6%)	58	(-18.3%)	54	(-23.9%)	44	(-38.0%)
ZDT2	89	90	(+1.1%)	85	(-4.5%)	69	(-22.5%)	68	(-23.6%)
ZDT3	68	68	(0.0%)	67	(-1.5%)	61	(-10.3%)	51	(-25.0%)
ZDT4	80	83	(+3.8%)	76	(-5.0%)	72	(-10.0%)	58	(-27.5%)
Average			**-0.2%**		**-7.3 %**		**-16.7 %**		**-28.5%**

Table 3: Obtained results for different values of inheritance proportion, for techniques FI7, FI8 and FI9.

FI10	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	71	(0.0%)	58	(-18.3%)	59	(-16.9%)	48	(-32.4%)
ZDT2	89	78	(-12.4%)	78	(-12.4%)	69	(-22.5%)	58	(-34.8%)
ZDT3	68	70	(+2.9%)	63	(-7.4%)	61	(-10.3%)	47	(-30.9%)
ZDT4	80	78	(-2.5%)	81	(+1.3%)	58	(-27.5%)	52	(-35.0%)
Average			**-3.0%**		**-9.2 %**		**-19.3 %**		**-33.3%**
FI11	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	62	(-12.7%)	63	(-11.3%)	55	(-22.5%)	37	(-48.0%)
ZDT2	89	84	(-5.6%)	87	(-2.2%)	81	(-9.0%)	76	(-14.6%)
ZDT3	68	69	(+1.5%)	60	(-11.8%)	57	(-16.2%)	44	(-35.3%)
ZDT4	80	82	(+2.5%)	81	(+1.3%)	73	(-8.8%)	73	(-8.8%)
Average			**-3.6%**		**-6.0 %**		**-14.1 %**		**-26.7%**
FI12	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	66	(-7.0%)	56	(-21.1%)	55	(-22.5%)	48	(-32.4%)
ZDT2	89	87	(-2.2%)	85	(-4.5%)	74	(-16.9%)	80	(-10.1%)
ZDT3	68	66	(-2.9%)	64	(-5.9%)	55	(-19.1%)	53	(-22.1%)
ZDT4	80	80	(0.0%)	75	(-6.3%)	71	(-11.3%)	61	(-23.8%)
Average			**-3.0%**		**-9.5 %**		**-17.5 %**		**-22.1%**

Table 4: Obtained results for different values of inheritance proportion, for techniques FI10, FI11 and FI12.

FI13	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	68	(-4.2%)	69	(-2.8%)	63	(-11.3%)	58	(-18.3%)
ZDT2	89	84	(-5.6%)	81	(-9.0%)	79	(-11.2%)	80	(-10.1%)
ZDT3	68	70	(+2.9%)	68	(0.0%)	63	(-7.4%)	54	(-20.6%)
ZDT4	80	79	(-1.3%)	81	(+1.3%)	64	(-20.0%)	59	(-26.3%)
Average			**-2.1%**		**-2.6 %**		**-12.5 %**		**-18.8%**
FI14	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	75	(+5.6%)	66	(-7.0%)	58	(-18.3%)	59	(-16.9%)
ZDT2	89	88	(-1.1%)	79	(-11.2%)	83	(-6.7%)	72	(-19.1%)
ZDT3	68	74	(+8.8%)	69	(+1.5%)	63	(-7.4%)	60	(-11.8%)
ZDT4	80	81	(+1.3%)	79	(-1.3%)	73	(-8.8%)	67	(-16.3%)
Average			**+3.7%**		**-4.9 %**		**-10.3 %**		**-16.0%**
FI15	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	63	(-11.3%)	69	(-2.8%)	56	(-21.1%)
ZDT2	89	86	(-3.4%)	81	(-9.0%)	72	(-19.1%)	73	(-18.0%)
ZDT3	68	72	(+5.9%)	70	(+2.9%)	64	(-5.9%)	58	(-14.7%)
ZDT4	80	81	(+1.3%)	78	(-2.5%)	63	(-21.3%)	70	(-12.5%)
Average			**+0.3%**		**-5.0 %**		**-12.3 %**		**-16.6%**

Table 5: Obtained results for different values of inheritance proportion, for techniques FI13, FI14 and FI15.

FA1	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	64	(-9.9%)	63	(-11.3%)	55	(-22.5%)
ZDT2	89	88	(-1.1%)	85	(-4.5%)	81	(-9.0%)	76	(-14.6%)
ZDT3	68	73	(+7.4%)	61	(-10.3%)	60	(-11.8%)	55	(-19.1%)
ZDT4	80	85	(+6.3%)	89	(+11.3%)	79	(-1.3%)	80	(0.0%)
Average			**+4.2%**		**-3.4 %**		**-8.4 %**		**-14.1%**
FA2	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	75	(+5.6%)	57	(-19.7%)	54	(-23.9%)	46	(-35.2%)
ZDT2	89	83	(-6.7%)	72	(-19.1%)	63	(-29.2%)	76	(-14.6%)
ZDT3	68	63	(-7.4%)	58	(-14.7%)	58	(-14.7%)	56	(-17.6%)
ZDT4	80	86	(+7.5%)	87	(+8.8%)	81	(+1.3%)	83	(+3.8%)
Average			**-0.3%**		**-11.2 %**		**-16.6 %**		**-15.9%**
FA3	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	71	(0.0%)	67	(-5.6%)	63	(-11.3%)	50	(-29.6%)
ZDT2	89	88	(-1.1%)	87	(-2.2%)	85	(-4.5%)	76	(-14.6%)
ZDT3	68	65	(-4.4%)	65	(-4.4%)	55	(-19.1%)	57	(-16.2%)
ZDT4	80	89	(+11.3%)	91	(+13.8%)	86	(+7.5%)	87	(+8.8%)
Average			**+1.5%**		**+0.4 %**		**-6.9 %**		**-12.9%**

Table 6: Obtained results for different values of approximation proportion, for techniques FA1, FA2 and FA3.

FA4	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	59	(-16.9%)	60	(-15.5%)	52	(-26.8%)
ZDT2	89	87	(-2.2%)	80	(-10.1%)	76	(-14.6%)	71	(-20.2%)
ZDT3	68	67	(-1.5%)	71	(+4.4%)	56	(-17.6%)	56	(-17.6%)
ZDT4	80	86	(+7.5%)	85	(+6.3%)	79	(-1.3%)	80	(0.0%)
Average			**+0.3%**		**-4.1 %**		**-12.3 %**		**-16.2%**

Table 7: Obtained results for different values of approximation proportion, for technique FA4.

Test Function DTLZ2								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	**25**	16	18	18	12	13	12
	std. dev.	4.2	6.7	6.9	7.7	5.2	6.4	6.8
IGD	mean	0.0014	0.0021	**0.0014**	0.0014	0.0015	0.0015	0.0015
	std. dev.	0.00005	0.0004	0.00004	0.00005	0.0001	0.00008	0.00009
Test Function DTLZ6								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	**62**	61	60	61	43
	std. dev.	0	0	13	17.3	21.7	17.2	20.7
IGD	mean	0.0673	0.0373	0.0091	**0.0074**	0.0089	0.0087	0.0101
	std. dev.	0.0000	0.0172	0.0058	0.0060	0.0060	0.0058	0.0058

Table 9: Obtained results for the test functions DTLZ2 and DTLZ6, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with the fitness inheritance technique FI5 incorporated (p_i=0.1,0.2,0.3,0.4).

bels: sMOPSO refers to [7], cMOPSO refers to [11], and oMOPSO is our MOPSO. All the algorithms were set such that they provided Pareto fronts with 100 points. In this case, we also show the obtained results with respect to the *Inverted Generational Distance* (IGD) measure. This measure indicates how far is the true Pareto front from the obtained Pareto front (using an Euclidean distance in the objective space). Thus, this measure gives an idea of how close and widely spread is the obtained Pareto front with respect to the true Pareto front.

Tables 9 and 10 present a summary of the results obtained. In each case, we present the average and standard deviation of the Success Counting (SCC) and IGD measures over the 20 runs. As we can see in Tables 9 and 10, in function DTLZ2 our approach (oMOPSO) is outperformed by one of the other PSO-based approaches, with respect to the SCC measure. However, the values obtained by oMOPSO in the IGD measure in this function indicate that our approach obtained as good approximations of the true Pareto front as the other algorithms (since values of IGD measure are mainly based on how are the obtained points distributed along the true Pareto front, it is possible that different values of the SCC measure correspond to similar values on the IGD measure). On the other hand, in function DTLZ6, our approach is clearly the best, with respect to the two measures. From Table 9, we conclude that the fitness inheritance technique FI5 has a better performance in function DTLZ6 than in function DTLZ2 with respect to the SCC measure. However, the IGD measure indicates a very good performance in all cases, even with respect to the other PSO-based approaches. Table 10 shows a very good performance of the fitness approximation technique FA3 in both functions and with respect to the two measures.

In general, technique FA3 was better than FI5 in function DTLZ2, in which it offers a 12% of decrement in quality with a saving of 30% in evaluations in the best case, and a 28% of decrement in quality with a saving of 40% in evaluations, in the worst case. On the other hand, technique FI5 was better than FA3 in function DTLZ6. In function DTLZ6, technique FI5 offers a 2% of decrement in quality with a saving of 30% in evaluations in the best case, and a 30% of decrement in quality with a saving of 40% in evaluations, in the worst case. These results agree with the results obtained before. As we can see in Table 11, the results obtained in the previous study show that technique FA3 is consistently better than technique FI5 in function ZDT4. Function ZDT4 has 10 variables, while functions ZDT1, ZDT2 and ZDT3 have 30 variables. Also, function DTLZ2 has 12 variables while function DTLZ6 has 22 variables. In this way, we can conclude that the fitness approximation technique FA3 has better results when the test function has a low dimensional decision space and that the fitness inheritance technique FI5 has better results when the test function has a high dimensional decision space. This conclusion seems to agree with the obtained results in our previous work [8].

9 Conclusions

We proposed several fitness inheritance and approximation techniques and incorporated them into a Multi-Objective Particle Swarm Optimizer proposed previously by the authors. We studied the proposed techniques using several well-known test functions from the multi-objective optimization literature. We found one fitness inheritance technique and one approximation technique to be the best techniques proposed. The best fitness inheritance technique is based on the flight formula for the objective space proposed in this paper. The best approximation technique is based on the simple idea of assigning to a particle the same objective values of the closest particle member of the swarm. Both techniques were tested on other functions and compared with other PSO-based multi-objective algorithms. The obtained results show that both enhancement techniques have

Test Function DTLZ2								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	**25**	16	18	16	15	16	13
	std. dev.	4.2	6.7	6.9	6	7.7	8.2	7.3
IGD	mean	0.0014	0.0021	**0.0014**	0.0014	0.0015	0.0015	0.0015
	std. dev.	0.00005	0.0004	0.00004	0.00005	0.00007	0.0001	0.0001
Test Function DTLZ6								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	**62**	62	51	54	53
	std. dev.	0	0	13	15	27	20	20
IGD	mean	0.0673	0.0373	**0.0091**	0.0109	0.0099	0.0100	0.0116
	std. dev.	0.0000	0.0172	0.0058	0.0056	0.0059	0.0061	0.0056

Table 10: Obtained results for the test functions DTLZ2 and DTLZ6, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with the fitness approximation technique FA3 incorporated (p_a=0.1,0.2,0.3,0.4).

FI5	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	**74**	(+4.2%)	**69**	(-2.8%)	61	(-14.1%)	**56**	(-21.1%)
ZDT2	89	**89**	(0.0%)	79	(-11.2%)	84	(-5.6%)	**77**	(-13.5%)
ZDT3	68	**72**	(+5.9%)	**70**	(+2.9%)	**55**	(-19.1%)	**58**	(-14.7%)
ZDT4	80	87	(+8.8%)	85	(+6.3%)	85	(+6.3%)	82	(+2.5%)
Average			**+4.7%**		**-1.2 %**		**-8.1 %**		**-11.7%**
FA3	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	71	(0.0%)	67	(-5.6%)	**63**	(-11.3%)	50	(-29.6%)
ZDT2	89	88	(-1.1%)	**87**	(-2.2%)	**85**	(-4.5%)	76	(-14.6%)
ZDT3	68	65	(-4.4%)	65	(-4.4%)	55	(-19.1%)	57	(-16.2%)
ZDT4	80	**89**	(+11.3%)	**91**	(+13.8%)	**86**	(+7.5%)	**87**	(+8.8%)
Average			**+1.5%**		**+0.4 %**		**-6.9 %**		**-12.9%**

Table 11: Obtained results for different values of approximation proportion, for techniques FI5 and FA3.

a good performance and are very promising. In fact, fitness inheritance techniques seem to be more appropriate for high-dimensional decision space problems and fitness approximation techniques seem more appropriate for low-dimensional decision space problems. As part of our future work, we plan to improve the enhancement techniques that were found to be the best in this study, in order to minimize the decrement in quality of results with a major saving in the number of evaluations performed.

Acknowledgments. The first author acknowledges CINVESTAV-IPN for its support and to CONACyT for granting her a scholarship to pursue graduate studies at CINVESTAV-IPN. The second author acknowledges support from CONACyT project number 42435-Y.

Bibliography

[1] J. Branke and C. Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9(1):13–20, January 2005.

[2] 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.

[3] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *CEC 2002*, volume 1, pages 825–830, USA, 2002. IEEE Service Center.

[4] Els I. Ducheyne, Bernard De Baets, and Robert De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In *EMO 2003*, pages 31–42. Springer. LNCS 2632, 2003.

[5] Yaochu Jin and Bernhard Sendhoff. Fitness Approximation in Evolutionary Computation: A Survey. In *GECCO 2002*, pages 1105–1112. Morgan Kaufmann, 2002.

[6] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[7] Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *IEEE Swarm Intelligence Symposium Proceedings*, pages 26–33, USA, 2003. IEEE Service Center.

[8] Margarita Reyes Sierra and Carlos A. Coello Coello. Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *IEEE Swarm Intelligence Symposium*, USA, 2005. IEEE Service Center (accepted for publication).

[9] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-based Multi-objective Optimization using Crowding, Mutation and ϵ-Dominance. In *EMO 2005*, pages 505–519. Springer-Verlag, LNCS 3410, 2005.

[10] Robert E. Smith, B. A. Dike, and S. A. Stegmann. Fitness Inheritance in Genetic Algorithms. In *SAC '95*, pages 345–350. ACM Press, 1995.

[11] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In *GECCO 2004. Part I*, pages 225–237, USA, 2004. Springer-Verlag, LNCS 3102.

[12] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.