Fitness Inheritance 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

Given the high computational cost of function evaluation in most real world problems, the development of efficient optimization algorithms has raised a lot of interest. In this paper, we describe the design of both an efficient multi-objective optimization approach and an enhancement technique to reduce computational cost. First, we propose a multi-objective optimizer based on the particle swarm strategy, that provides very competitive results. Then, we provide the first proposal to incorporate the concept of fitness inheritance into a multi-objective particle swarm optimizer. After a study of several different techniques to incorporate fitness inheritance into our approach, we conclude that this enhancement technique is able to reduce computational cost without dramatically deteriorating the quality of the results. Also, we show that when fitness inheritance is applied dynamically throughout the evolutionary process, savings of about 32% of the total number of evaluations can be obtained without affecting the quality of the solutions. Furthermore, even reducing the computational cost by 78%, the proposed approaches are able to obtain very good approximations of the true Pareto front.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search — *Heuristic Methods*; G.1.6 [**Numerical Analysis**]: Optimization.

General Terms: Algorithms, Performance.

Keywords: Fitness Inheritance, Multi-Objective Optimization, Particle Swarm Optimization.

1. INTRODUCTION

Since function evaluation in real world applications is usually very expensive, the use of Evolutionary Algorithms (EAs), which are population-based techniques, becomes very expensive, too. The main motivation for our work was to design an efficient algorithm, in terms of the number of function evaluations performed, for solving multi-objective problems. First, we proposed a multi-objective optimization algorithm inspired on the particle swarm strategy. The results obtained by our approach were very competitive. Then, with the aim of reducing computational cost, we incorporated into our multi-objective algorithm an enhancement technique called fitness inheritance. In fitness inheritance [16], the fitness value of an offspring is obtained from the fitness values of its parents. In this way, we do not need to evaluate every individual at each generation, and the number of function evaluations is reduced. From the results obtained, we concluded that fitness inheritance techniques are able to significantly reduce the computational cost without decreasing the quality of the results in a dramatic way. Thus, we proceeded to study different mechanisms to incorporate fitness inheritance into our approach. Finally, with the aim of obtaining more savings in the number of evaluations performed, we decided to analyze the possibility of applying fitness inheritance following a dynamical scheme, along the evolutionary process. With this mechanism, that we applied using the best inheritance technique found in the previous study, we were able to obtain very good approximations of the true Pareto front, even with a 78% of savings in the total number of evaluations.

2. STATEMENT OF THE PROBLEM

We are interested in solving problems of the type:
minimize $[f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]$
subject to: (1) $g_i(\vec{x}) \leq 0, i = 1, 2, \dots, m$, (2) $h_i(\vec{x}) = 0, i = 1, 2, \dots, p$, where k is the number of objective functions $f_i : R^n \rightarrow R$. We call $\vec{x} = [x_1, x_2, \dots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set $\mathcal{F}$ of all the vectors that satisfy (1) and (2) to the vectors $x_1^*, x_2^*, \dots, x_n^*$ that are *Pareto optimal.* We say that a vector of decision variables $\vec{x}^* \in \mathcal{F}$ is *Pareto optimum* if there does not exist another $\vec{x} \in \mathcal{F}$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ for every $i = 1, \dots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one j. The vectors $\vec{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *nondominated.* The objective function values corresponding to the elements of the Pareto optimal set are called the *Pareto front* of the problem.

3. PARTICLE SWARM OPTIMIZATION

Kennedy and Eberhart [6] 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 a hyperdimensional search space. A swarm consists of a set of particles, where each particle represents a potential solution. The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{x}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\vec{v}_i(t)$ to the current position, i.e.:

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

The velocity vector drives the optimization process and reflects the socially exchanged information. In the global best version (used here) of PSO, each particle uses its history of experiences in terms of its own best solution thus far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*). Thus, the velocity vector changes in the following way:

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

where W is the inertia weight, C_1 and C_2 are the learning factors (usually defined as constants), and $r_1, r_2 \in [0,1]$ are random values. In this work, we use $W = random(0.1, 0.5)$ and $C_1, C_2 = random(1.5, 2.0)$.

PSO must be modified in order to apply it to multi-objective problems. In most approaches (which will be generically called MOPSOs, for Multiple-Objective Particle Swarm Optimizers), the major modifications of the PSO algorithm are the selection process of *pbest* and *gbest* [2, 9, 8].

4. FITNESS INHERITANCE

The use of fitness inheritance to improve the performance of GAs was originally proposed by Smith et al. [16]. 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 (based on the similarity between the offspring and the parents). They applied inheritance to a very simple problem (the OneMax problem) [16] 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.

Fitness inheritance is applied to an individual with certain probability (like the crossover or the mutation operator). Otherwise, the individual is evaluated using the true fitness function. This probability is called *inheritance proportion* p_i. The inheritance proportion p_i is a parameter that has to be fixed by the user, and its value determines the number of evaluations that are going to be saved. For example, if the inheritance proportion has the value of 0.1, it means that the corresponding EA is going to save a 10% of the total number of evaluations (this is an approximate value).

Salami et al. [15] proposed a "Fast Evolutionary Algorithm" in which they incorporated the concept of fitness inheritance. They obtained an average reduction of 40% in the number of evaluations while obtaining solutions of similar quality. Bui et al. [1] performed a comparison of the performance of anti-noise methods, particularly probabilistic and re-sampling methods, using the NSGA-II [3]. They applied the concept of fitness inheritance and obtained a substantial amount of savings in computational time (reaching 30% in the best case), without deteriorating the performance.

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

Figure 1: Pseudocode of the MOPSO algorithm. The symbol (⇒) indicates the line in which the concept of fitness inheritance is incorporated.

5. DESCRIPTION OF OUR WORK

In this section, we describe first the MOPSO approach proposed and then our studies about the incorporation of fitness inheritance techniques.

5.1 Multi-Objective Particle Swarm Optimization

In the first stage of our work, we proposed a MOPSO approach in [11], and an updated version in [10]. The fitness inheritance techniques proposed later, were incorporated into this approach. The MOPSO proposed in [11, 10] is based on Pareto dominance, since it considers every nondominated solution as a new leader (a leader is used as the *gbest* solution in Equation 2). Additionally, the approach also uses a crowding factor [3] as a second discrimination criterion which is also adopted to filter out the list of available leaders. For each particle, a leader is selected in the following way: 97% of the time a leader is randomly selected if and only if that leader dominates the current particle, and, the remaining 3% of the time, the selection is done 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. This approach also uses different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. This is done by means of 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, this MOPSO approach also incorporates the ϵ-dominance concept [7] to fix the size of the set of final solutions produced by the algorithm. Details about distinctive aspects of this algorithm can be found in [11]. Figure 1 shows the pseudocode of the proposed MOPSO algorithm.

We performed a comparative study in which we tested our approach (without inheritance) using four test functions taken from the specialized literature of multi-objective optimization: ZDT1, ZDT2, ZDT3 and ZDT4 [18]. We compared our approach against three PSO-based approaches:

the Sigma-MOPSO [9], the Cluster-MOPSO [17] and the MOPSO [2]. Also, we compared our approach against two multi-objective evolutionary algorithms representative of the state of the art: NSGA-II [3] and SPEA2 [19]. We performed 20 runs for each function and each approach. The parameters of each approach were set such that they all performed 20000 objective function evaluations. Table 1 shows the obtained results for the *Success Counting (SCC)*[1] measure, which was one of the three measures used in [11, 10]. Also, Figure 2 shows the Pareto fronts corresponding to the nondominated vectors obtained from the union of the 20 Pareto fronts produced by each approach, for function ZDT4, which is considered the hardest of the test functions used. As we can see in Table 1 and Figure 2, the obtained results indicated that our MOPSO approach was highly competitive.

5.2 Fitness Inheritance in MOPSO

In [10], we proposed the first attempt to incorporate the concept of fitness inheritance to a real-coded MOPSO. Since PSO has no recombination operator, we adopted as "parents" of a particle the previous position of the particle and its leader. Thus, based on the distance of the particle to its parents (in decision space), we proposed to assign, as the new objective function values of the particle, a linear combination of the corresponding values of the parents. It should be noted that, in this case, we are not inheriting fitness (since our MOPSO approach does not use fitness for selection), but the values of the objective functions. Also, in order to avoid the generation of invalid particles, we changed the fitness inheritance mechanism when the leader chosen does not dominate the current particle. In this case, we use the values obtained using the original scheme to locate the closest leader to that position. Then, the objective function values of such leader are assigned to the new particle. We can see both types of inheritance in Figure 3.

We tested our approach using the same four ZDT functions previously mentioned. We performed experiments with different values of *inheritance proportion*: p_i= 0.1, 0.2, 0.3, 0.4. Since the parameters were set such that 20000 objective function evaluations were performed, the approach with fitness inheritance performed approximately 18000 evaluations when p_i= 0.1, 16000 when p_i= 0.2, 14000 when p_i= 0.3 and 12000 when p_i= 0.4. Table 2 shows the obtained results (over 20 runs) for the *SCC* measure, which was one of the three measures used in [10]. In this case, in Figure 4, we show the Pareto fronts corresponding to the nondominated vectors obtained from the union of the 20 Pareto fronts produced by each approach, for functions ZDT1, ZDT2 and ZDT3.

From our comparative study, we concluded that fitness inheritance reduces the computational cost without decreasing the quality of the results in a significant way [10]. Also, the fitness inheritance technique used was able to generate non-convex and discontinuous Pareto fronts. These conclusions were somewhat surprising since, previous to the work presented in [10], 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.

Since the results obtained with the first fitness inheritance technique proposed were promising, we decided to study other ways of incorporating such concept into our PSO-based approach. Also, we decided to apply the concept of fitness approximation in order to reduce the computational cost. Approximation techniques let us estimate the fitness of an individual using the previously calculated fitness of its neighbors [5]. We proposed a total of nineteen techniques: fifteen fitness inheritance techniques and four approximation techniques [12]. The fitness inheritance techniques are based on three main ideas:

1. Linear Combination Based on Distances (LCBD). In this case, we generalized the idea of the first inheritance technique proposed. We proposed 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 (previous position, *pbest* and leader). Three variants of this technique were proposed: **FI1** ignoring the *pbest* position (first technique proposed), **FI2** ignoring the previous position and **FI3** considering the three particles[2]. See Figure 5 for an illustration of these techniques. In all the inheritance techniques, if the leader selected does not dominate the current particle, we proceed as before. See Figure 3.

2. Flight Formula on Objective Space (FFOS). In this case, we propose a formula analogous to Equation 2, to update the position of particle i in the objective space:

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

where $\vec{f_i}$, $\vec{f}_{pbest_i}$ and $\vec{f}_{gbest_i}$ are the objective vectors for the current particle i, its *pbest* and *gbest*, respectively. We use the same values of W, C_1, r_1, C_2 and r_2 previously adopted for the flight in decision variable space. As in the previous case, three variants were considered: **FI4** considering the whole formula, **FI5** ignoring the previous direction and **FI6** ignoring the direction to the *pbest*.

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:

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

The three variants corresponding to this technique (defined as in the previous case) were called: **FI7**, **FI8** and **FI9**. 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:

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

We obtain the following variants: **FI10**, **FI11** and **FI12**.
Linear Combination (**LC**). In this case, we propose to use the previous formula but in such a way that the result is a linear combination of the elements considered:

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

[1]$SCC = \sum_{i=1}^{n} s_i$, where n is the number of vectors in the obtained set of nondominated solutions; $s_i = 1$ if solution i is a member of the Pareto optimal set, and $s_i = 0$ otherwise.

[2]We consider the position of the leader as the most important. Thus, the leader is always considered.

Table 1: Obtained results for all the approaches, for functions ZDT1, ZDT2, ZDT3 and ZDT4.

Function	*SCC*	oMOPSO[10]	NSGA-II[3]	SPEA2[19]	MOPSO[2]	sMOPSO[9]	cMOPSO[17]
ZDT1	average	**71**	21	27	0	59	8
	st. dev.	13.6	7.5	8.1	0	24.2	7.9
ZDT2	average	**89**	6	7	0	1	29
	st. dev.	10	9.8	10.4	0	0	38.9
ZDT3	average	**68**	44	39	0	26	0
	st. dev.	18.2	6.8	6.0	0	25.4	0
ZDT4	average	**80**	0	0	0	0	0
	st. dev.	16.3	0	0	0	0	0

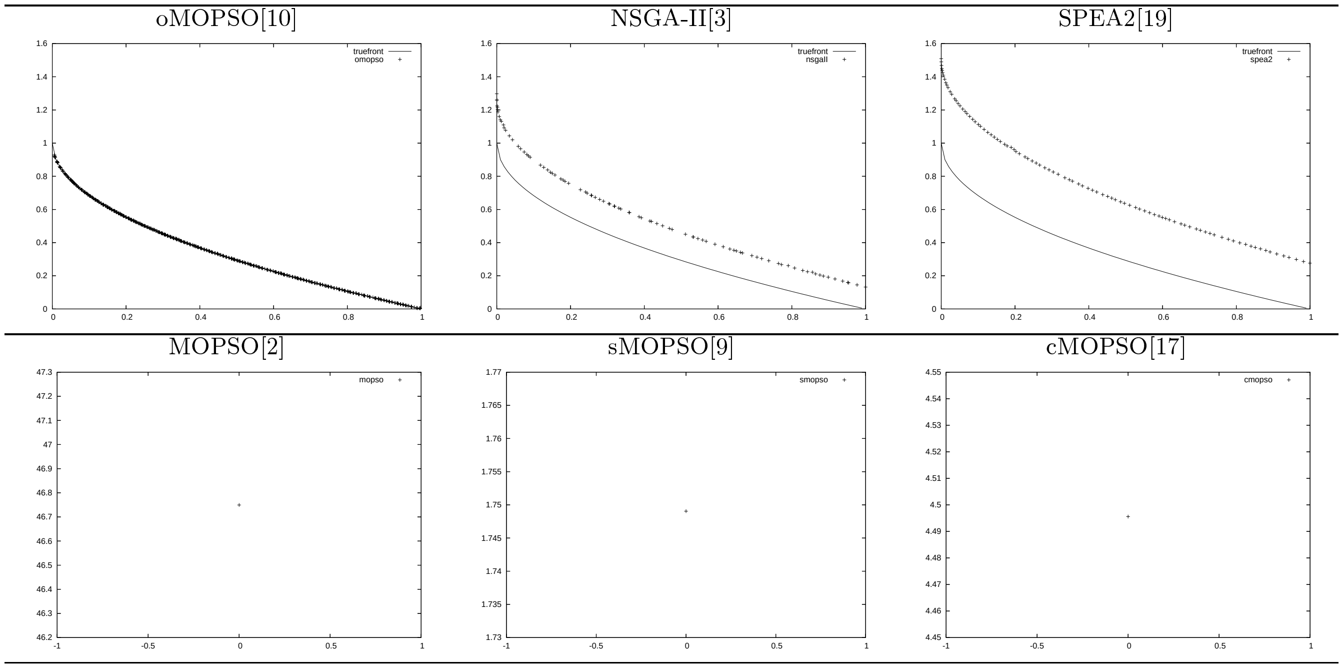

Figure 2: Pareto fronts obtained by all the approaches, for test function ZDT4, which is considered the hardest among the test functions used. We can observe that our MOPSO approach obtained the best approximation of the true Pareto front.

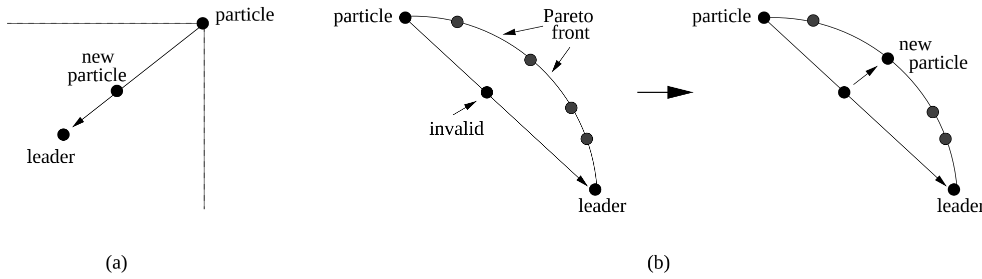

Figure 3: The two possible cases of fitness inheritance: (a) when the leader dominates the particle, the new particle inherits a linear combination of the objective values of the parents; (b) when the leader does not dominate the particle, the new particle inherits the objective values of the closest leader to the position corresponding to the linear combination previously calculated.

Table 2: Obtained results by the approach without and with inheritance, for functions ZDT1, ZDT2, ZDT3 and ZDT4.

Function	*SCC*	oMOPSO	p_i=0.1	p_i=0.2	p_i=0.3	p_i=0.4
ZDT1	average	71	77	64	62	61
	st. dev.	13.6	14.5	19.8	16.4	14.8
ZDT2	average	89	83	86	79	77
	st. dev.	10	22.5	13.1	22.4	20.7
ZDT3	average	68	73	65	64	59
	st. dev.	18.2	11.2	13.9	14.5	21.2
ZDT4	average	80	81	81	60	68
	st. dev.	16.3	13	12.8	22.7	25.4

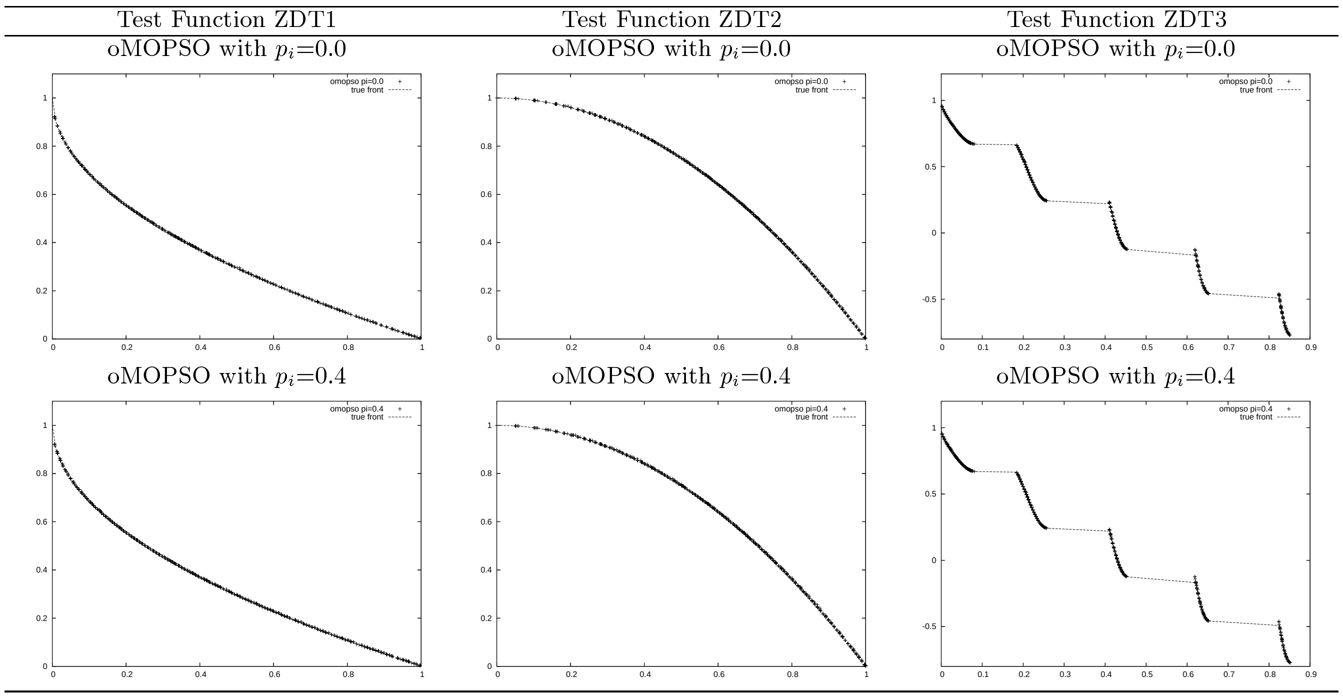

Figure 4: Pareto fronts obtained by the approach without and with inheritance, for functions ZDT1, ZDT2 and ZDT3. In the case of the approach with inheritance, for each function, we show the Pareto front corresponding to the case with the value of p_i that gave the worst results with respect to the *SCC* measure.

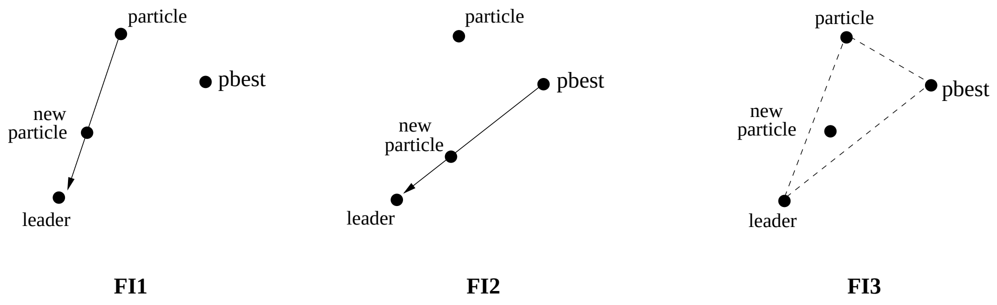

Figure 5: Illustration of techniques FI1, FI2 and FI3. When using technique FI1, the new particle inherits a linear combination of the objective values of the previous particle and the leader. In the case of technique FI2, the new particle inherits a linear combination of the objective values of the *pbest* and the leader. Finally, when using technique FI3, the new particle inherits a linear combination of the objective values of the previous particle, the *pbest* and the leader.

Table 3: Vectors of change in quality for each technique, for each value of inheritance or approximation proportion.

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	

where $r = W + C_1r_1 + C_2r_2$. The corresponding variants are **FI13**, **FI14** and **FI15**.
In the case of fitness approximation, we adopt very simple techniques. On 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) and **FA4** the average of the 10 closest particles (leaders or members of the swarm). In all cases, we use the Euclidean distance in decision variable space.

We performed a study of the nineteen techniques using the same four ZDT functions and the *SCC* measure of performance [12]. All the parameters were set in the same way as in the previous study [10]. Since comparing 19 different techniques is very difficult, we decided to represent each technique with a vector. The vector used is the one containing the average of change in the quality of results with respect to the approach without inheritance, for each inheritance proportion value. In this way, in Table 3 we present the vectors of all techniques. Since every entry in each vector is a change in the quality, the bigger the values of the vector, the better the corresponding technique is. Thus, we are interested on the vectors that maximize all the entries. The non-dominated vectors among all the 19 techniques are the vectors corresponding to techniques FI5 and FA3. That is, techniques FI5 and FA3 are the best. For this reason, these two techniques are marked with a level of 1 in Table 3. For these two techniques, in the worst case, the decrement in quality is no more than 13%, even when a 40% of the total number of evaluations is saved. After eliminating techniques FI5 and FA3, the nondominated vectors (marked with level 2) correspond to techniques: 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) were the best.

As we have seen, in all the experiments previously performed, the savings in the number of evaluations was completely determined by the value of inheritance proportion p_i. With the aim of obtaining more savings, we decided to study the possibility of setting the value of the inheritance proportion parameter p_i following a dynamical scheme [13].

From the previous work, we concluded that the use of fitness inheritance decreases the quality of the results as we increase the value of the parameter p_i [10, 12]. So, our main idea was to increase the savings in number of function evaluations but setting the value of p_i in such a way that fitness inheritance can be less harmful.

We proceeded to analyze the behavior of our MOPSO approach, with respect to the improvement on the current Pareto front throughout the evolutionary process, that is, along the generations. With this aim, we used the *Two Set Coverage (SC)* measure of performance[3]. Given the current Pareto front in generation t, $PF(t)$, and the Pareto front in the previous generation $PF(t-1)$, we calculated the value of the SC measure: $SC(PF(t), PF(t-1))$. In this way, we could know "how much" better is the current Pareto front with respect to the front of the previous generation.

We performed a set of experiments using our MOPSO approach without inheritance and function ZDT1, in which we obtained the average of $SC(PF(t), PF(t-1))$ at each generation [13]. We observed that the most important improvement takes place during the first quarter of the total of generations. In this way, we concluded that at the beginning of the process it is not convenient to use too much fitness inheritance. However, towards the end of the process, fitness inheritance is more suitable. Thus, given the previous conclusions, we proposed to set the value of the parameter p_i dynamically with respect to the current generation number, with the aim of increasing the use of fitness inheritance throughout the evolutionary process. We proposed six different functions to adapt the value of the inheritance proportion: Let *gen* be the number of the current generation and *Gmax* the total number of generations:

- nonlinear1: $p_i(gen) = \left(\frac{gen}{Gmax}\right)^4$
- nonlinear2: $p_i(gen) = \left(\frac{gen}{Gmax}\right)^2$
- nonlinear3: $p_i(gen) = \frac{gen}{Gmax} - \frac{\sin(2\Pi\frac{gen}{Gmax})}{6.3}$
- linear: $p_i(gen) = \frac{gen}{Gmax}$
- nonlinear4: $p_i(gen) = \left(\frac{gen}{Gmax}\right)^{\frac{1}{2}}$
- nonlinear5: $p_i(gen) = \left(\frac{gen}{Gmax}\right)^{\frac{1}{4}}$

These six functions (that we will call *adaptive functions*) increase the inheritance proportion value throughout the evolutionary process. Figure 6 presents a plot of the six adaptive functions. As we can see in Figure 6, the adaptive functions are numbered following an ascending order with respect to the savings on the total number of evaluations that they define.

In order to test the mechanism proposed, we proceeded to incorporate it into our MOPSO approach using the fitness inheritance technique FI5, since it was found to obtain the best results in [12]. In this case, we ran the algorithm 30 times using the same four ZDT functions. Also, based on the results obtained from an ANOVA (Analysis of Variance) performed on the parameters of our MOPSO approach

[3]This binary measure $SC(X, X')$ indicates the percentage of solutions in X' that are dominated by solutions in X.

Table 4: Obtained results for all the test functions and all the adaptive functions.

function	*SCC*	no-inherit	nonlinear1	nonlinear2	nonlinear3	linear	nonlinear4	nonlinear5
ZDT1	average	87	84	74	71	68	53	21
	st. dev.	12.5	12.6	21	18.6	22.7	21.6	13.5
ZDT2	average	92	93	89	83	84	69	45
	st. dev.	12.9	6.1	12.2	21.7	22.9	26.6	34.2
ZDT3	average	76	73	72	53	59	37	16
	st. dev.	12.7	11.6	15.9	21.5	16.2	18	12.6
ZDT4	average	96	94	93	89	90	77	47
	st. dev.	4.8	6.6	6.0	12.6	14.2	18.1	22.6
evaluations (avg.)		20200	16302	13632	10298	10302	6964	4321
savings (avg.)		0%	19%	32%	49%	49%	65%	78%

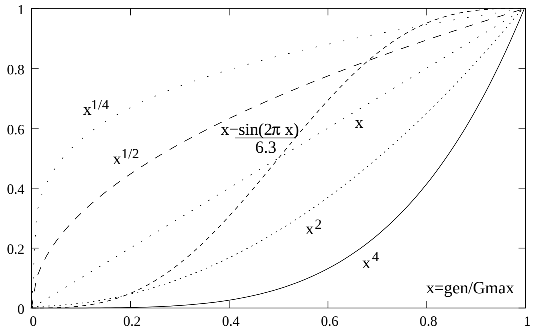

Figure 6: Plot of the six different functions proposed to adapt the value of the inheritance proportion (p_i) through the evolutionary process.

[14], we were able to improve the performance of our algorithm by changing the configuration of the values used, while performing the same number of function evaluations[4]. In this way, our MOPSO approach performed 20200 objective function in the absence of inheritance. Table 4 shows the obtained results for the *SCC* measure, which was one of the three measures used in [13]. The Pareto fronts shown in Figure 7 correspond to the median value with respect to the *SCC* measure and are distributed, for each test function, in three plots: (1) approach without inheritance and with inheritance and adaptive functions nonlinear1 and nonlinear2, (2) with inheritance and adaptive functions nonlinear3 and linear and (3) with inheritance and adaptive functions nonlinear4 and nonlinear5.

From the obtained results, we concluded that, in general, it is possible to save even a 32% of evaluations without affecting the quality of the obtained solutions (using adaptive functions nonlinear1 and nonlinear2). On the other hand, as it was expected from their definition, adaptive functions nonlinear3 and linear, provide the same percentage of savings in the number of evaluations (about 49%). However, the results when using the function linear are, in general, better than the corresponding results using function nonlinear3. In this way, we conclude that, given their corresponding definition, it is important to maintain the true evaluations at the end of the run, even if we try not to apply inheritance at the beginning of the evolutionary process. Finally, adaptive functions that save more than a 50% of the total number of evaluations, nonlinear4 (about 65%) and nonlinear5 (about 78%), affect the results more dramatically. However, as we can see in Figure 7 (which shows the cases on which this effect was more stressed), even in those cases, the obtained Pareto fronts are very good approximations of the true Pareto front.

[4]In all the previous experiments, we used a swarm of 100 particles for 200 generations. The ANOVA indicated that better results could be obtained by using a swarm of 200 particles for 100 generations.

6. CONCLUSIONS AND FUTURE WORK

With the aim of designing an efficient multi-objective optimization algorithm, we have first proposed a new multi-objective particle swarm optimizer that was found to be highly competitive. Then, we provided the first attempt to incorporate the concept of fitness inheritance into a real-coded MOPSO approach, in order to reduce the computational cost. We studied several different ways of incorporating such enhancement technique and also experimented with some simple fitness approximation schemes. Using the best inheritance technique found, we proposed a mechanism to adapt the value of the inheritance proportion in a dynamical way, throughout the evolutionary process. From the obtained results, we concluded that it is possible to save until a 32% of the total number of evaluations without significantly deteriorating the quality of the results. In fact, although the quantitative quality of the Pareto fronts provided by the approaches that save 65% and 78% of evaluations is more affected, the corresponding plots show that such approaches are able to generate very good approximations of the true Pareto front. In this way, although it seems to be very harmful to save such percentage of evaluations, if in a real-world application, it is very expensive to evaluate the objective functions and we are interested only in a few solutions, the proposed approach may be a suitable choice.

As part of our future work, we plan to test the proposed approaches on different test functions (including real-world applications) and also to design new adaptive functions in order to analyze two possibilities: (1) to increase the savings obtained without deteriorating more the quality of the results (2) to improve the quality of the obtained results while keeping constant the percentage of savings. Also, it would be very interesting to incorporate the proposed approaches into a different evolutionary algorithm in order to study how is the quality of the results affected when a high percentage of evaluations is saved.

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

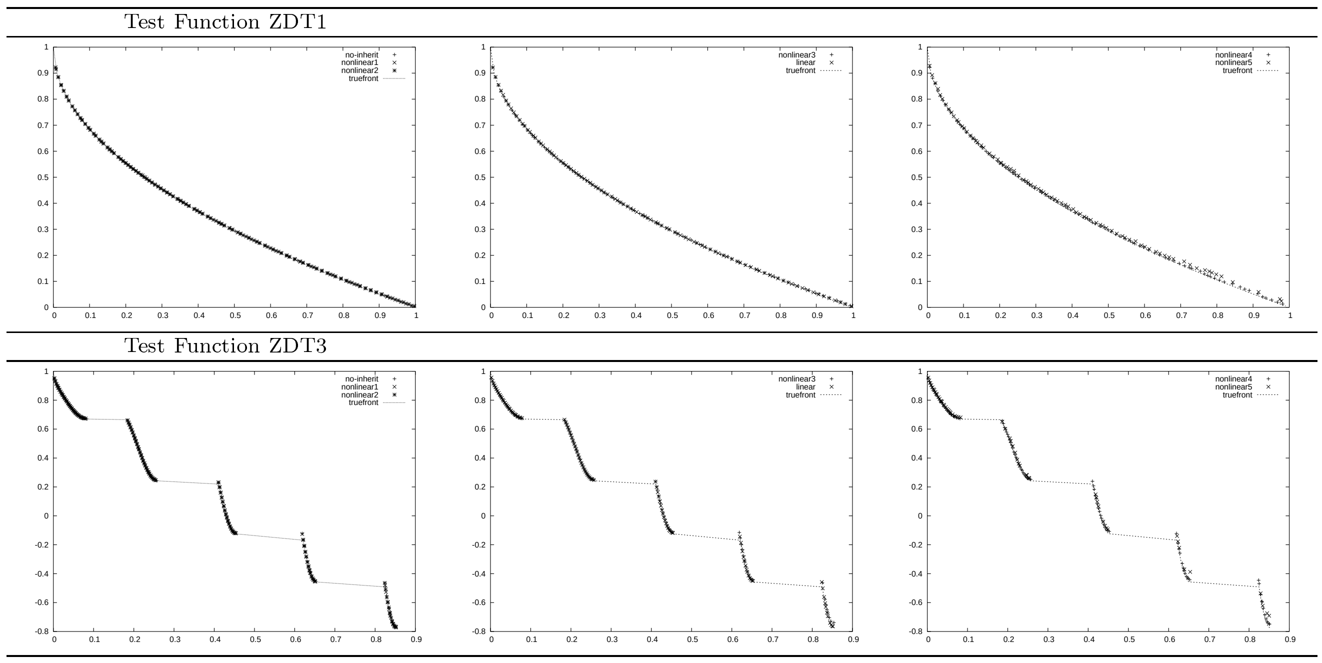

Figure 7: Pareto fronts obtained for functions ZDT1 and ZDT3, in which the quality of the results was more affected by the use of fitness inheritance.

7. REFERENCES

[1] L. T. Bui, H. A. Abbass, and D. Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 25–29. ACM, 2005.

[2] C. A. Coello Coello, G. Toscano-Pulido, and M. Salazar-Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[4] E. I. Ducheyne, B. De Baets, and R. De Wulf. Is fitness inheritance useful for real-world applications? In *Second International Conference on Evolutionary Multi-Criterion Optimization*, pages 31–42. Springer. LNCS 2632, 2003.

[5] Y. Jin and B. Sendhoff. Fitness approximation in evolutionary computation: A survey. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 1105–1112. Morgan Kaufmann, 2002.

[6] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Service Center, 1995.

[7] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[8] X. Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 37–48. Springer. LNCS 2723, 2003.

[9] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *IEEE Swarm Intelligence Symposium*, pages 26–33. IEEE Service Center, 2003.

[10] M. Reyes-Sierra and C. A. Coello Coello. Fitness inheritance in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 116–123. IEEE Service Center, 2005.

[11] M. Reyes-Sierra and C. A. Coello Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Third International Conference on Evolutionary Multi-Criterion Optimization*, pages 505–519. LNCS 3410, Springer-Verlag, 2005.

[12] M. Reyes-Sierra and C. A. Coello Coello. A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In *Congress on Evolutionary Computation*, pages 65–72. IEEE Service Center, 2005.

[13] M. Reyes-Sierra and C. A. Coello Coello. Dynamic fitness inheritance proportion for multi-objective particle swarm optimization. Technical Report EVOCINV-03-2006, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, April 2006.

[14] M. Reyes-Sierra and C. A. Coello Coello. On-line adaptation in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*. IEEE Service Center (*accepted for publication*), 2006.

[15] M. Salami and T. Hendtlass. A fast evaluation estrategy for evolutionary algorithms. *Applied Soft Computing*, 2(3):156–173, 2003.

[16] R. E. Smith, B. A. Dike, and S. A. Stegmann. Fitness inheritance in genetic algorithms. In *Proc. of the 1995 ACM Symposium on Applied Computing*, pages 345–350. ACM Press, 1995.

[17] G. Toscano-Pulido and C. A. Coello Coello. Using clustering techniques to improve the performance of a particle swarm optimizer. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 225–237. Springer-Verlag, LNCS 3102, 2004.

[18] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[19] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proc. of the EUROGEN2001 Conference*, pages 95–100. CIMNE, 2002.