Dynamic Fitness Inheritance Proportion 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

Fitness inheritance is an enhancement technique used to reduce the computational cost of evolutionary algorithms. What fitness inheritance does is, with certain probability, to avoid evaluating an individual. Instead, such individual gets assigned a fitness value which is somehow obtained from the fitness values of its parents. The probability of applying fitness inheritance to an individual is called inheritance proportion. Such parameter is usually given by the user, and its value determines the savings on the number of evaluations performed. In this paper, we propose a dynamic mechanism to vary the value of the inheritance proportion through the run in order to obtain a greater reduction in computational cost, without dramatically affecting the quality of the results. We apply our proposed approach into a multi-objective particle swarm optimizer with a fitness inheritance technique previously proposed by the authors. Several functions to adapt the inheritance proportion through the run are tested using functions taken from the specialized multi-objective optimization literature. The results obtained show that it is possible to reduce the computational cost by 32% without affecting the quality of the obtained Pareto front. Also, the proposed approaches are able to obtain very good approximations of the true Pareto front even when reducing the computational cost by 78%.

*Margarita Reyes-Sierra and Carlos A. Coello Coello, "Dynamic Fitness Inheritance Proportion For Multi-Objective Particle Swarm Optimization", Technical Report EVOCINV-03-2006, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, April 2006.

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. Fitness inheritance is an enhancement technique that has been proposed to improve the performance of EAs [11]. In fitness inheritance, the fitness value of an offspring is obtained from the fitness values of its parents. When using fitness inheritance, we do not need to evaluate every individual at each generation, and the computational cost is reduced. When a fitness inheritance technique is incorporated into an EA, the user has to fix a new parameter called *inheritance proportion*. Such parameter indicates the probability with which an individual is going to inherit its fitness from its parents. In this way, a fixed value of inheritance proportion determines how much the computational cost is going to be reduced.

In this paper, we propose a mechanism to adapt the value of the inheritance proportion in a dynamic way through the run, in order to analyze how much can we reduce the computational cost without dramatically deteriorating the quality of the obtained results. The proposed approach is incorporated into a Multi-Objective Particle Swarm Optimization (MOPSO) algorithm previously proposed in [7]. We use four well-known multi-objective test functions in order to test the performance of the proposed mechanism.

This paper is organized as follows. In Section 2, we define the problem that we want to solve. A brief introduction to fitness inheritance is given in Section 3. Section 4 introduces the MOPSO algorithm in which the fitness inheritance technique and the proposed mechanism are incorporated. The fitness inheritance technique used is described in Section 5. The dynamic mechanism proposed to adapt the value of inheritance proportion is presented in Section 6. In Section 7 and 8 we present the obtained results and their discussion, respectively. Finally, the conclusions and future work are described in Section 9.

2 Statement of the Problem

We are interested in solving problems of the type:

$$\text{minimize } [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}$$

subject to:

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

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where k is the number of objective functions $f_i : R^n \rightarrow R$. We call $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set $\mathcal{F}$ of all the vectors that satisfy (2) and (3) to the vectors $x_1^*, x_2^*, \ldots, x_n^*$ 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 $i = 1, \ldots, k$ 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 Fitness Inheritance

The use of fitness inheritance to improve the performance of GAs was originally proposed by Smith et al. [11]. 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) [11] 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*. The inheritance proportion 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).

Sastry et al. [10] provided some theoretical foundations for fitness inheritance. They investigated convergence times, population sizing and the optimal inheritance proportion for the OneMax problem. Chen et al. [2] investigated fitness inheritance as a way to speed up multi-objective GAs and EAs. They extended the analytical model proposed by Sastry et al. for multi-objective problems. Convergence and population-sizing models are derived and compared with respect to experimental results. The authors concluded that the number of function evaluations can be reduced with the use of fitness inheritance.

Salami et al. [9] proposed a "Fast Evolutionary Algorithm" in which a fitness and associated reliability value are assigned to each new individual that is only evaluated using the true fitness function if the reliability value is below a threshold. Also, they incorporated random evaluation and error compensation strategies. The authors obtained an average reduction of 40% in the number of evaluations while obtaining solutions of similar quality. In the same work, they presented an application of fitness inheritance to image compression obtaining reductions between 35% and 42% of the number of evaluations.

Bui et al. [1] performed a comparison of the performance of anti-noise methods, particularly probabilistic and re-sampling methods, using NSGA-II

[3]. They applied the concept of fitness inheritance to both types of methods in order to reduce calculation time. The authors obtained a substantial amount of savings in computational time (reaching 30% in the best case), without deteriorating the performance.

4 Multi-Objective Particle Swarm Optimization

In this paper, we incorporate our proposed approach into a MOPSO that was previously proposed in [7] and updated in [6]. The MOPSO proposed in [7, 6] is based on Pareto dominance, since it considers every non-dominated solution as a new leader. 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 [5] to fix the size of the set of final solutions produced by the algorithm. Figure 1 shows the pseudocode of the MOPSO algorithm used in this work.

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

5 Fitness Inheritance in MOPSO

The first attempt to incorporate the concept of fitness inheritance to a real-coded MOPSO was proposed in [6]. After testing the performance of weighted average fitness inheritance on a well-known test suite of multi-objective optimization problems [12], the authors concluded that fitness inheritance reduces the computational cost without decreasing the quality of the results in a significant way [6]. 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 [6], Ducheyne et al. [4] tested

```
Begin
    Initialize swarm. Initialize leaders.
    Send leaders to ϵ-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 ϵ-archive
        crowding(leaders), g++
    EndWhile
    Report results in ϵ-archive
End
```

Figure 1: Pseudocode of the MOPSO algorithm.

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.

In this work, we use a fitness inheritance technique previously proposed [8]. In [8], the authors studied several different techniques for inheriting the fitness of individuals in the MOPSO algorithm previously described. In this paper, we use the fitness inheritance technique that was found to obtain the best results. 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 [8], the authors proposed an analogous formula to update the position of each particle in the objective space:

$$\vec{\mathbf{f_i}}(t) = \vec{\mathbf{f_i}}(t-1) + \vec{\mathbf{vf_i}}(t)$$

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

where $\vec{\mathbf{f_i}}$, $\vec{\mathbf{f}}_{\mathbf{pbest_i}}$ and $\vec{\mathbf{f}}_{\mathbf{gbest_i}}$ are the values of the objective function i for the current particle, its *pbest* and *gbest*, respectively. The values of W, C_1, r_1, C_2 and r_2 are the same previously used for the flight in the decision variable space.

The fitness inheritance technique that gave the best results in [8], called FI5, is based on the flight formula for the objective space. However, the authors found the best performance of technique FI5 when the corresponding flight formula ignores the previous direction of the particle ($W = 0$):

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

6 Proposed Approach

From the work presented in [6, 8], the authors concluded that fitness inheritance techniques are able to reduce the computational cost significantly without decreasing the quality of the results in a dramatic way. However, in all the previous experiments 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 in the number of evaluations performed, we decided to study the possibility of setting the value of the inheritance proportion parameter p_i following a dynamical scheme.

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 [6, 8]. So, our main idea is 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 the MOPSO approach previously described, with respect to the improvement on the current Pareto front through the evolutionary process, that is, through the generations. With this aim, we use the following binary measure of performance:

Two Set Coverage (SC): This metric was proposed in [12]. Consider X', X'' as two sets of phenotype decision vectors. SC is defined as the mapping of the order pair (X', X'') to the interval $[0, 1]$:

$$SC(X', X'') = |\{a'' \epsilon X''; \exists a' \epsilon X' : a' \prec a''\}|/|X''|$$

where $\prec$ means "dominates". If all points in X' dominate or are equal to all points in X'', then by definition $SC(X', X'') = 1$. $SC(X', X'') = 0$ implies the opposite.

Given the current Pareto front in generation t, $PF(t)$, and the Pareto front in the previous generation $PF(t-1)$, we calculate the value of the SC measure:

$$SC(PF(t), PF(t-1))$$

In this way, we can know "how much" better is the current Pareto front with respect to the front of the previous generation.

We performed 30 runs of the MOPSO approach without inheritance and using function ZDT1 (whose definition is given in the next section), 100 particles

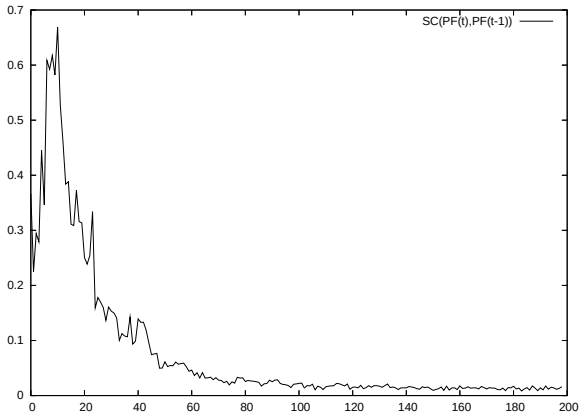

Figure 2: Average value of $SC(PF(t), PF(t-1))$ at each generation.

and 200 generations, and we obtained the average of $SC(PF(t), PF(t-1))$ at each generation. Figure 2 shows the obtained results. As we can see, the most important improvement takes place during the first quarter of the total of generations (the first 50 generations in this case). 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 propose to set the value of the parameter p_i dynamically with respect to the current generation number. The main idea is to increase the use of fitness inheritance through the evolutionary process. In this way, we propose six different functions to adapt the value of the inheritance proportion with respect to the number of the current generation: 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: $p_i(gen) = \left(\frac{gen}{Gmax}\right)^{\frac{1}{4}}$

The six previous functions (that we will call *adaptive functions*) are designed in such a way that the value of inheritance proportion increases through

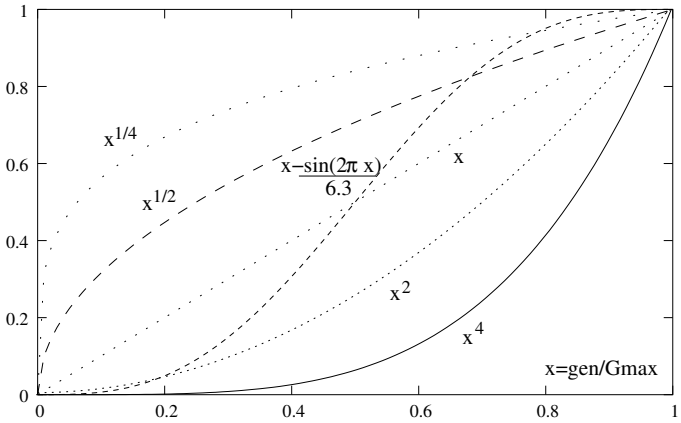

Figure 3: Plot of the six different functions proposed to adapt the value of the inheritance proportion (p_i) through the evolutionary process.

the evolutionary process. Nevertheless, it is important to note that $gen = 0, ..., Gmax - 1$, and that $p_i = 0$ when $gen = 0$, since fitness inheritance must not be applied when $gen = 0$. Figure 3 presents a plot of the six adaptive functions.

The adaptive functions are numbered following an ascending order with respect to the savings on the total number of evaluations that they define. In this way, the adaptive function nonlinear1 is the one that defines the least savings on the total number of evaluations. On the other hand, the adaptive function nonlinear5 is the one that defines the greatest savings on the total number of evaluations. See Figure 3.

7 Results

In order to test the mechanism proposed to set the value of the parameter p_i, we proceeded to incorporate it into the MOPSO approach and we performed the following experiments. We ran the algorithm 30 times using functions ZDT1, ZDT2, ZDT3 and ZDT4:

- Test Function ZDT1 [12]:

  Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
  $f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
  $g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$, $h(f1, g) = 1 - \sqrt{f_1/g}$
  where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT2 [12]:

Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
$f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
$g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$, $h(f1, g) = 1 - (f_1/g)^2$
where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT3 [12]:

  Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
  $f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
  $g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$, $h(f1, g) = 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1)$
  where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT4 [12]:

  Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
  $f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
  $g(\vec{x}) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 - 10cos(4\pi x_i))$, $h(f1, g) = 1 - \sqrt{f_1/g}$
  where $m = 10$, $x_1 \in$[0,1] and $x_i \in$[-5,5], $i = 2, ..., m$.

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.

The parameters used were 200 particles, 100 generations and 100 points in the final Pareto Front. In this way, the total number of evaluations is 20200 in the absence of fitness inheritance. As we said before, in all the experiments performed we used the fitness inheritance technique FI5, since it was found to be the best among the inheritance techniques proposed in [8]. For our comparative study, we implemented two unary measures of performance:

Success Counting (SCC): This measure counts the number of vectors, in the current set of nondominated vectors available, that are members of the Pareto optimal set:

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

where n is the number of vectors in the current set of nondominated vectors available; $s_i = 1$ if vector i is a member of the Pareto optimal set, and $s_i = 0$ otherwise.

Inverted Generational Distance (IGD): This measure indicates how far is the true Pareto front from the obtained Pareto front:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

where n is the number of elements in the true Pareto front and d_i is the Euclidean distance (measured in objective space) between each of these and the

Table 1: Obtained results for all the adaptive functions, for test function ZDT1.

Function ZDT1					
		no-inherit	nonlinear1	nonlinear2	nonlinear3
SCC	best	99	99	94	97
	median	93	87	79	76
	worst	47	38	16	29
	mean	87	84	74	71
	st. dev.	12.5	12.6	21	18.6
IGD	best	0.00091	0.00091	0.00091	0.00091
	median	0.00097	0.00097	0.00097	0.00096
	worst	0.00100	0.00104	0.00198	0.04922
	mean	0.00096	0.00096	0.00103	0.00313
	st. dev.	0.00003	0.00003	0.00024	0.00890
evaluations		20200	16306	13640	10295
savings		0%	19.3%	32.5%	49%
		no-inherit	linear	nonlinear4	nonlinear5
SCC	best	99	95	81	62
	median	93	75	55	17
	worst	47	12	10	1
	mean	87	68	53	21
	st. dev.	12.5	22.7	21.6	13.5
IGD	best	0.00091	0.00092	0.00092	0.00099
	median	0.00097	0.00096	0.00103	0.00141
	worst	0.00100	0.03182	0.04922	0.08305
	mean	0.00096	0.00280	0.00388	0.00838
	st. dev.	0.00003	0.00693	0.00979	0.01803
evaluations		20200	10303	6966	4319
savings		0%	49%	65.5%	78.6%

nearest member of the set of nondominated vectors found by the algorithm.

Tables 1, 2, 3 and 4 present the results obtained using the unary measures. For each test function, we present first the results obtained by the MOPSO approach without inheritance and then, the results of the approach with inheritance with each one of the adaptive functions, from function nonlinear1 to function nonlinear5. For each approach, we present the best, median, worst, mean and standard deviation values with respect to the two unary measures implemented. Also, we show the average number of evaluations performed on each case and the corresponding percentage of savings obtained.

On the other hand, for each test function, we selected the Pareto fronts corresponding to the median value with respect to the SCC measure to represent each approach. In this way, Table 5 presents the results obtained when ap-

Table 2: Obtained results for all the adaptive functions, for test function ZDT2.

Function ZDT2					
		no-inherit	nonlinear1	nonlinear2	nonlinear3
SCC	best	100	100	100	99
	median	96	94	94	91
	worst	35	69	47	0
	mean	92	93	89	83
	st. dev.	12.9	6.1	12.2	21.7
IGD	best	0.00063	0.00063	0.00063	0.00063
	median	0.00065	0.00065	0.00066	0.00067
	worst	0.00096	0.00076	0.00081	0.00587
	mean	0.00067	0.00066	0.00067	0.00092
	st. dev.	0.00006	0.00002	0.00004	0.00100
evaluations		20200	16304	13641	10295
savings		0%	19.3%	32.5%	49%
		no-inherit	linear	nonlinear4	nonlinear5
SCC	best	100	100	98	95
	median	96	93	79	46
	worst	35	2	0	0
	mean	92	84	69	45
	st. dev.	12.9	22.9	26.6	34.2
IGD	best	0.00063	0.00063	0.00063	0.00067
	median	0.00065	0.00067	0.00070	0.00111
	worst	0.00096	0.00353	0.06777	0.04557
	mean	0.00067	0.00078	0.00516	0.00378
	st. dev.	0.00006	0.00053	0.01390	0.00904
evaluations		20200	10298	6968	4316
savings		0%	49%	65.5%	78.6%

Table 3: Obtained results for all the adaptive functions, for test function ZDT3.

Function ZDT3					
		no-inherit	nonlinear1	nonlinear2	nonlinear3
SCC	best	91	91	93	86
	median	78	74	74	57
	worst	42	38	9	4
	mean	76	73	72	53
	st. dev.	12.7	11.6	15.9	21.5
IGD	best	0.00076	0.00081	0.00079	0.00085
	median	0.00086	0.00090	0.00093	0.00186
	worst	0.00134	0.00198	0.01586	0.05416
	mean	0.00090	0.00101	0.00200	0.00742
	st. dev.	0.00014	0.00027	0.00322	0.01375
evaluations		20200	16312	13622	10290
savings		0%	19.2%	32.6%	49.1%
		no-inherit	linear	nonlinear4	nonlinear5
SCC	best	91	89	69	48
	median	78	60	37	11
	worst	42	17	10	2
	mean	76	59	37	16
	st. dev.	12.7	16.2	18	12.6
IGD	best	0.00076	0.00082	0.00083	0.00167
	median	0.00086	0.00112	0.00282	0.01291
	worst	0.00134	0.02734	0.04911	0.05052
	mean	0.00090	0.00232	0.01085	0.01779
	st. dev.	0.00014	0.00490	0.01371	0.01473
evaluations		20200	10304	6966	4336
savings		0%	49%	65.5%	78.5%

Table 4: Obtained results for all the adaptive functions, for test function ZDT4.

Function ZDT4					
		no-inherit	nonlinear1	nonlinear2	nonlinear3
SCC	best	100	99	98	99
	median	97	97	96	94
	worst	78	69	74	49
	mean	96	94	93	89
	st. dev.	4.8	6.6	6.0	12.6
IGD	best	0.00090	0.00090	0.00090	0.00091
	median	0.00097	0.00097	0.00097	0.00096
	worst	0.00098	0.00098	0.00098	0.00100
	mean	0.00096	0.00096	0.00096	0.00095
	st. dev.	0.00002	0.00002	0.00002	0.00003
evaluations		20200	16287	13626	10315
savings		0%	19.4%	32.5%	48.9%
		no-inherit	linear	nonlinear4	nonlinear5
SCC	best	100	99	95	81
	median	97	95	83	48
	worst	78	28	3	2
	mean	96	90	77	47
	st. dev.	4.8	14.2	18.1	22.6
IGD	best	0.00090	0.00090	0.00092	0.00096
	median	0.00097	0.00100	0.00096	0.00110
	worst	0.00098	0.00100	0.00139	0.00212
	mean	0.00096	0.00100	0.00098	0.00124
	st. dev.	0.00002	0.00003	0.00008	0.00032
evaluations		20200	10304	6958	4315
savings		0%	49%	65.6%	78.6%

Table 5: Success Counting measure.

Function ZDT1						
X	nl1	nl2	nl3	lin	nl4	nl5
SC(no-inh,X)	0.33	0.23	0.25	0.25	0.23	0.66
SC(X,no-inh)	0.04	0.02	0.00	0.00	0.00	0.00
Function ZDT2						
X	nl1	nl2	nl3	lin	nl4	nl5
SC(no-inh,X)	0.37	0.21	0.29	0.20	0.39	0.30
SC(X,no-inh)	0.03	0.07	0.05	0.04	0.00	0.00
Function ZDT3						
X	nl1	nl2	nl3	lin	nl4	nl5
SC(no-inh,X)	0.27	0.20	0.34	0.37	0.40	0.67
SC(X,no-inh)	0.14	0.08	0.02	0.01	0.00	0.00
Function ZDT4						
X	nl1	nl2	nl3	lin	nl4	nl5
SC(no-inh,X)	0.21	0.12	0.19	0.13	0.11	0.30
SC(X,no-inh)	0.05	0.14	0.06	0.01	0.01	0.01

plying the SC measure to the Pareto fronts that represent each approach. For each case, we compare the Pareto front corresponding to the approach without inheritance against the Pareto fronts corresponding to the approaches with inheritance, using adaptive functions nonlinear1 to nonlinear5. Also, for each case we compare the Pareto fronts corresponding to the approaches with inheritance against the Pareto front of the approach without inheritance.

Finally, Figures 4, 5, 6 and 7 show the Pareto fronts that represent each approach. For each test function, we present three plots. The first one, shows the Pareto fronts from the approach without inheritance and the approaches with inheritance and adaptive functions nonlinear1 and nonlinear2. The second plot shows the Pareto fronts from approaches with inheritance and adaptive functions nonlinear3 and linear. Finally, the third plot shows the Pareto fronts from approaches with inheritance and adaptive functions nonlinear4 and nonlinear5.

8 Discussion of Results

As we can see in Tables 1, 2, 3 and 4, for all the test functions, when the application of the inheritance increases, the quality on the results is more affected. This behavior can be observed more dramatically in functions ZDT1 and ZDT3. In functions ZDT2 and ZDT4, the quality of the results is less affected by the application of fitness inheritance. In fact, in function ZDT4, we can see that the results with respect to the IGD measure are almost of the same quality, even when a 78% of evaluations are saved. Function ZDT4 is the one with the lowest number of variables. In this way, we may argue that the use of fitness

inheritance is less harmful when the decision variable space has low dimensionality. Actually, in function ZDT4 we argue that, considering both measures of performance, it is possible to save even a 65% of function evaluations without affecting the quality of the results.

Taking into account both unary measures of performance, we consider that the quality of the results is maintained when at least one of the measures indicates so. In this way, we can conclude that in general, it is possible to save even a 32% of evaluations without affecting the quality of the obtained solutions. That is, adaptive functions nonlinear1 and nonlinear2, seem to be good options to save evaluations without deteriorating the quality of the results.

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. In general, the results when using function linear are better than the corresponding results using function nonlinear3. Since both adaptive functions (nonlinear3 and linear) provide the same amount of savings, we conclude that given their corresponding definition, the obtained results indicate that 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.

As we can see, adaptive functions that save more than a 50% of the total number of evaluations, nonlinear4 and nonlinear5, affect the results more dramatically. That is, the damage on the quality of the results increase more rapidly when the savings are greater than a 50% of the evaluations.

Very similar conclusions can be obtained from the results obtained using measure SC. In Table 5, we can see again that the percentage of dominated solutions by the Pareto front corresponding to the approach without inheritance grows when we increase the use of fitness inheritance. In functions ZDT2 and ZDT4, the quality of the results is less affected by the use of inheritance, specially in the case of function ZDT4. In the case of adaptive functions nonlinear1 and nonlinear2, the results in Table 5 indicate that the Pareto fronts obtained using function nonlinear2 are better. On the other hand, results from adaptive functions nonlinear3 and linear are very similar, but we can conclude again that function linear provides better results that function nonlinear3. Finally, it is very clear that the use of adaptive functions nonlinear4 and nonlinear5 affects dramatically the obtained results, specially in the case of functions ZDT1 and ZDT3.

From Table 5, we can conclude that it may be possible to save a 65% of the total number of evaluations expecting to lose a 40% of quality, in the worst case. Also, it seems very harmful to save a 78% of evaluations, since in the worst case the quality was reduced by 67%.

Finally, as we can see in Figures 4 to 7, the Pareto fronts generated by the approaches with inheritance are very similar to the Pareto front from the approach without inheritance, even when a 49% of the total number of evaluations is saved. It is only in the case of the approaches with inheritance and adaptive functions nonlinear4 and nonlinear5 in which we can observe Pareto fronts of relatively low quality. However, even in those cases, we can see that the obtained Pareto fronts are very good approximations of the true Pareto front.

In general, from the obtained results shown in Tables 1, 2, 3, 4 and 5, we can conclude that it is possible to save a 32% of the total number of evaluations without significantly affecting the quality of the obtained solutions. Also, the quality of the results when having savings of 49% of the evaluations, is very acceptable. Finally, in the case of reducing the total number of evaluations by more than a 50%, the quality of the results is more affected. However, as we can see in the Pareto fronts shown, even with a 78% of savings, the approach with inheritance still provides 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 the real world application is very expensive to evaluate and we are interested only on a few optimal solutions, the proposed approach may be a suitable choice.

9 Conclusions and Future Work

Fitness inheritance is applied using a parameter called inheritance proportion. Such parameter is the probability with which an individual inherits its fitness value from its parents. Otherwise, the individual is evaluated using the true fitness function. On the other hand, when a fitness inheritance technique is incorporated into an evolutionary algorithm, the inheritance proportion also determines the number of evaluations (approximately) that are going to be saved.

We have proposed a mechanism to adapt the value of the inheritance proportion in a dynamical way, through the evolutionary process. Six different functions to adapt the inheritance proportion were proposed, each one defining a different percentage of savings, from 19% to 78% of the total number of evaluations. The proposed approach was incorporated into a MOPSO algorithm that was previously proposed and using an inheritance technique that is also part of previous work. The mechanism was tested using four different functions taken from the specialized literature of multi-objective optimization.

From the obtained results, we conclude that, in general, the use of fitness inheritance affects the quality of the provided Pareto fronts as we increase the number of true evaluations saved. However, such effect was less noticeable in the test function with the decision space of lowest dimensionality. On the other hand, we conclude that it is possible to save until a 32% of the total number of evaluations without significantly deteriorating the quality of the results. When the proposed approaches save a 49% of evaluations, the effect on the quality of the results is more noticeable. However, the provided results are of very good quality. Finally, 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.

As part of our future work, we plan to test the proposed approaches on different test functions and also to design new adaptive functions in order to analyze two possibilities: (1) to increase the savings obtained without deterio-

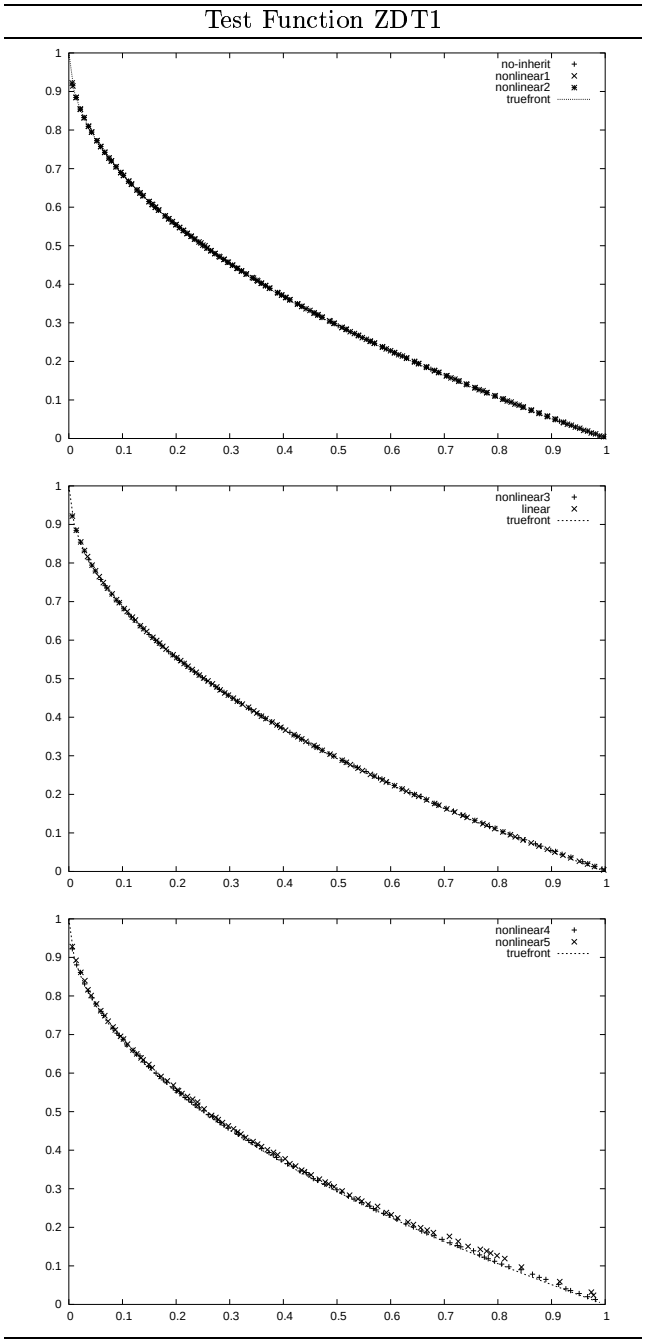

Figure 4: Pareto fronts obtained for function ZDT1.

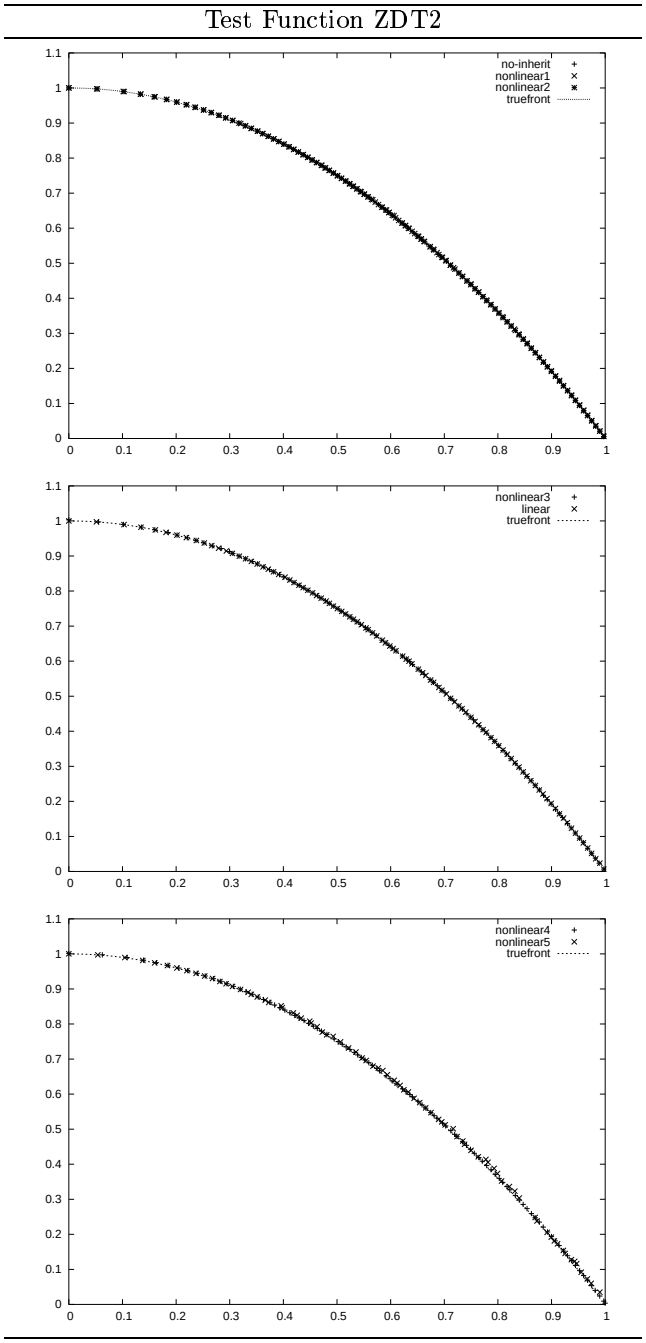

Figure 5: Pareto fronts obtained for function ZDT2.

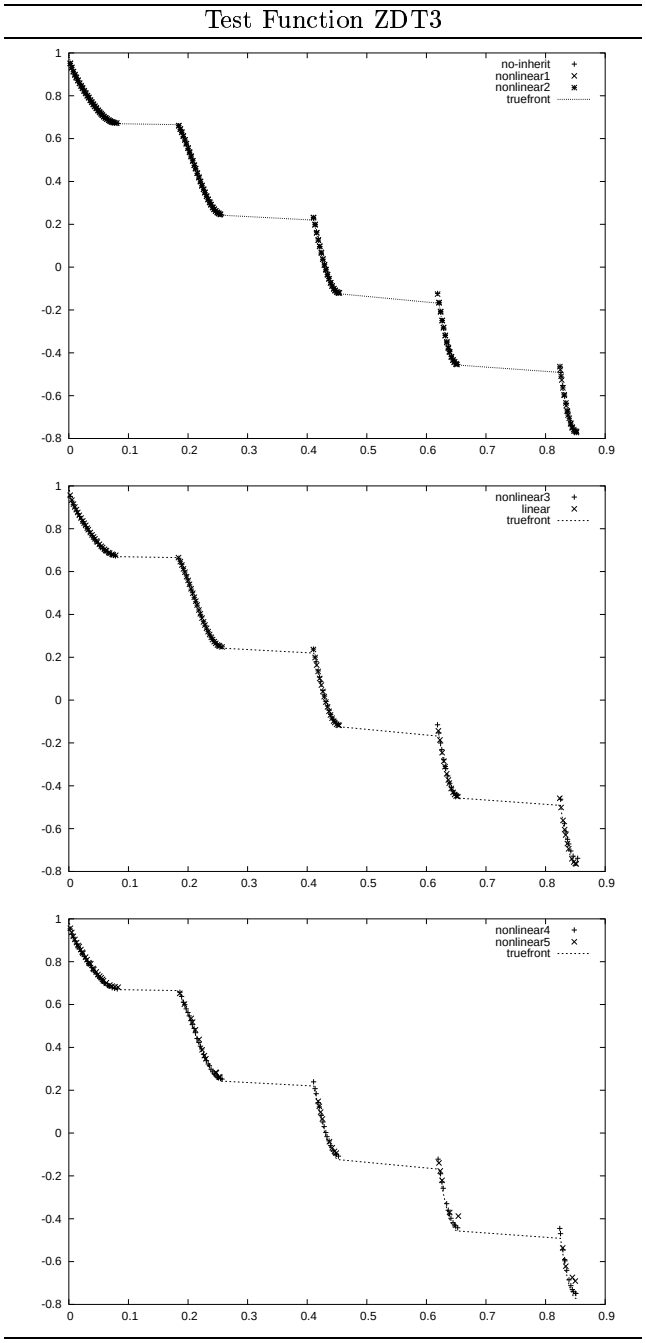

Figure 6: Pareto fronts obtained for function ZDT3.

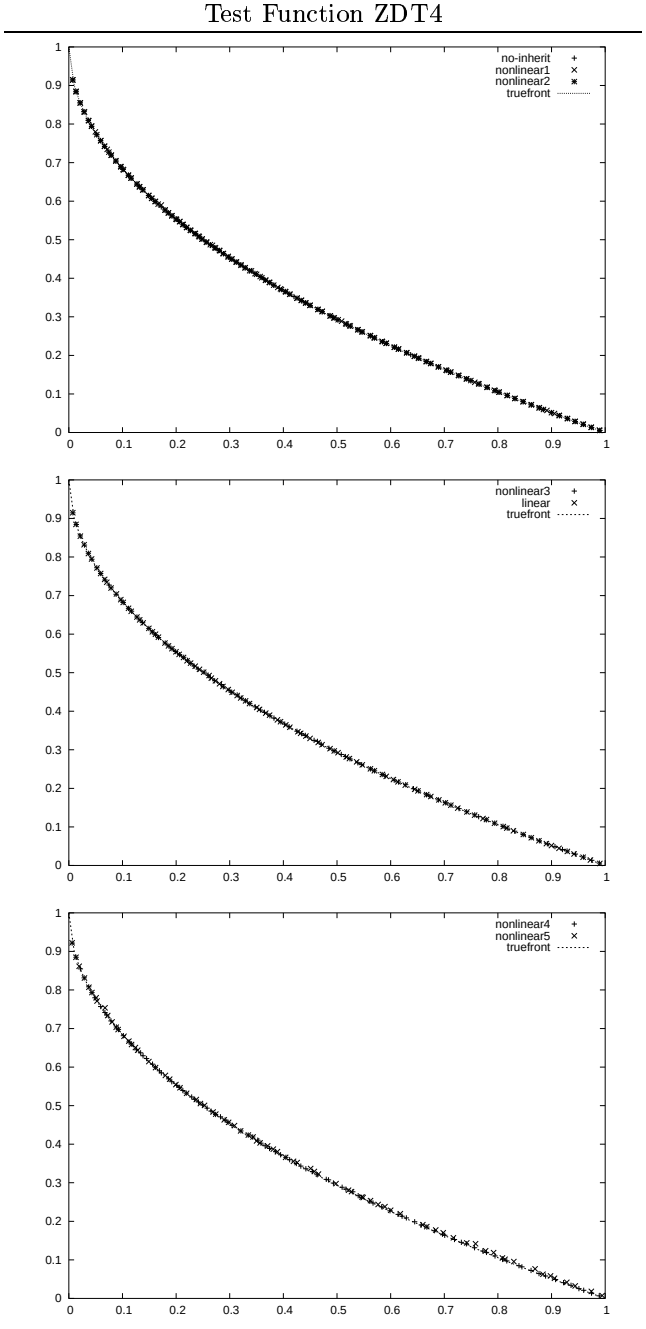

Figure 7: Pareto fronts obtained for function ZDT4.

rating more the quality of the results (2) to improve the quality of the obtained results while keeping constant the percentage of savings. The last possibility is related to the design of functions with the same integral value. 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.

References

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

[2] Jian-Jung Chen, David E. Goldberg, Shinn-Ying Ho, and Kumara Sastry. Fitness Inheritance in Multi-Objective Optimization. In W.B. Langdon et.al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 319–326, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[3] 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, 2002.

[4] Els I. Ducheyne, Bernard De Baets, and Robert De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In C. M. Fonseca et.al., editor, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[5] 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.

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

[7] Margarita Reyes-Sierra and Carlos 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.

[8] Margarita Reyes-Sierra and Carlos 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.

[9] Mehrdad Salami and Tim Hendtlass. A Fast Evaluation Estrategy for Evolutionary Algorithms. *Applied Soft Computing*, 2(3):156–173, 2003.

[10] Kumara Sastry, David E. Goldberg, and Martin Pelikan. Don't Evaluate, Inherit. In Lee Spector et.al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 551–558, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

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

[12] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.