FITNESS INHERITANCE IN MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

Margarita Reyes-Sierra and Carlos A. Coello Coello †

ABSTRACT

In this paper, we propose to incorporate the concept of fitness inheritance into a Multi-Objective Particle Swarm Optimizer previously proposed by the authors, in order to reduce the number of function evaluations performed. Four well-known test functions taken from the multi-objective optimization literature are used to evaluate the performance of the proposed approach. The results indicate a very good performance of the fitness inheritance technique, mainly when it is applied with a low probability, in which case the quality of the obtained results is even improved.

1. INTRODUCTION

In many real world applications of Evolutionary Algorithms (EAs), the fitness evaluation of the individuals in a population is computationally expensive. Fitness Inheritance is an enhancement technique that has been proposed to improve the performance of EAs [8]. In fitness inheritance, 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 computational cost is reduced. In this paper, we propose the first attempt (to the best of our knowledge) to incorporate the concept of fitness inheritance to a real-coded Particle Swarm Optimizer (PSO) that has been proposed previously by the authors to solve multi-objective problems [7]. Our approach with fitness inheritance is compared against the same approach without fitness inheritance and, also, against two other PSO-based multi-objective algorithms representative of the state-of-the-art. For our comparative study, we used four well-known test functions taken from the multi-objective optimization literature and three different measures of performance. This paper is organized as follows. An introduction to Fitness Inheritance is given in Section 2. Section 3 introduces the fitness inheritance technique proposed in this paper and the multi-objective PSO-based algorithm in which it is incorporated. In Section 4 and 5 we present the obtained results and their discussion, respectively. Finally, the conclusions and future work are described in Section 6.

†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

2. FITNESS INHERITANCE

The use of fitness inheritance to improve the performance of GAs was originally proposed by Smith et al. [8]. 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) [8] 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.

Sastry et al. [6] provide some theoretical foundations for fitness inheritance. They investigated convergence times, population sizing and the optimal proportion of inheritance for the OneMax problem. Chen et al. [1] investigate 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.

Ducheyne et al. [3] tested the performance of average and weighted average fitness inheritance on a well-known test suite of multi-objective optimization problems ([10]), using a binary GA. They concluded that the fitness inheritance efficiency enhancement techniques can be used in order to reduce the number of fitness evaluations provided that the Pareto front is convex and continuous. They also concluded that if the Pareto surface is not convex or if it is discontinuous, the fitness inheritance strategies fail to reach the true Pareto front.

3. DESCRIPTION OF OUR WORK

We propose to incorporate fitness inheritance into a Multi-Objective Particle Swarm Optimizer (MOPSO) that was previously proposed by us [7]. The MOPSO proposed in [7] is based on Pareto dominance, since it considers every non-dominated solution as a new leader. Additionally, the ap-

```
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 our algorithm.

proach 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 by means of a binary tournament based on the crowding value of the available 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 incorporated in the proposed approach the use of different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. We also 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 [4] to fix the size of the set of final solutions produced by the algorithm. Figure 1 shows the pseudo-code of our proposed approach.

Since leader selection is a key component in PSO, we have updated the version of our MOPSO algorithm published in [7] by including a new leader selection technique. In order to speed up the convergence to the true Pareto front of a multi-objective optimization problem, we have incorporated a new technique that is based only on Pareto dominance: given a particle x, we assign as its leader a particle y, randomly chosen, but *if and only if* y dominates x. If it is the case that x is a non-dominated particle, the leader will be randomly chosen. Since this new technique is based only on Pareto dominance, we have preserved the selection technique that our original MOPSO includes, in order to avoid losing diversity within the swarm. After an extensive series of experiments, we determined that our approach reached its best performance when using the new selection technique during 97% of the time (the old selection scheme is adopted during the remaining 3% of the time).

However, our main contribution in this paper is the incorporation of fitness inheritance within our MOPSO. The aim in this case is to reduce the number of fitness function evaluations required by our approach. Since PSO has no recombination operator, we adopted as "parents" of a particle the previous position of the particle and its leader. We performed experiments using the weighted average inheritance over real numbers encoding [8]:

Given a particle x_{old}, 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): $d_1 = d(x_{new}, x_{old})$, $d_2 = d(x_{new}, x_{ld})$, where d is an Euclidean distance in the decision variable space. Later, we calculate how near is the new particle from these two previous particles and proceed to inherit the corresponding objective function values:

$$r = \frac{d_1}{d_1 + d_2}$$

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

where f_i is the value of the objective function i and n is the number of objective functions.

Ducheyne [3] noted that the previous form of fitness inheritance has problems when the Pareto front is not convex. We came across this same limitation and after some analysis, we concluded that the problem arises in our MOPSO only when the leader chosen does not dominate the current particle. So, we changed the fitness inheritance mechanism when the above situation arises. 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 2.

In Figure 1, the symbol (⇒) indicates the line in which the concept of fitness inheritance is incorporated. The *inheritance proportion*, p_i, is 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 **can not** enter into the final Pareto front.

4. PRELIMINARY RESULTS

We have tested our approach using four test functions proposed in [10]: 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 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). Also, we compared our approach against other two PSO-based multi-objective approaches representative of the state-of-the-art: the Sigma-MOPSO [5] and the Cluster-MOPSO [9]. The approaches will be identified with the following labels: sMOPSO refers to [5], cMOPSO refers to [9], and oMOPSO is our MOPSO [7].

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. In this way, 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. The parameters adopted for oMOPSO were: 100 particles and 200 generations. cMOPSO used 40 particles, 4 swarms, 5 iterations per swarm and a total number of iterations of 100. In the case of sMOPSO, 200 particles were used through 100 iterations (as suggested by the author of this method). As recommended in [5], the sMOPSO used a turbulence probability of 0.01 for all functions, except for ZDT4 in which the turbulence probability used was 0.05. Our proposed approach used a probability mutation of $1/codesize$. All the algorithms were set such that they provided Pareto fronts with 100 points. For our comparative study, we implemented two unary and one binary 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 nearest member of the set of nondominated vectors found by the algorithm.

Two Set Coverage (SC) [11]: $SC(X', X'') \in [0, 1]$ gives the ratio of points in the set X'' that are dominated by at least one point in the set X'. In general, $SC(X', X'')$ and $SC(X'', X')$ both have to be considered due to set intersections not being empty. If $SC(X', X'') = 0$ and $SC(X'', X') = 1$, we say that X'' is better than X'.

Table 1 summarizes the results obtained with respect to the unary measures and Table 2 summarizes the results obtained for the binary measure. The Pareto fronts shown in Figures 3 and 4 correspond to the nondominated vectors obtained from the union of the 20 Pareto fronts produced by each approach. It should be noted that the Pareto fronts shown were also used to apply the binary measure of performance.

5. DISCUSSION OF RESULTS

As we can see in Table 1, our algorithm oMOPSO without fitness inheritance, has the best results in all cases with respect to the two the unary measures, compared with the other two PSO-based approaches. sMOPSO had problems to reach the true Pareto front of functions ZDT2 and ZDT4 and, although it obtained approximations to the true Pareto front of functions ZDT1 and ZDT3, sMOPSO was unable to cover the whole true Pareto front in both cases, as we can see in Figures 3 and 4. For this reason, sMOPSO has high values in the IGD measure. The cMOPSO approach had very similar problems with functions ZDT2 and ZDT4, however, in function ZDT2 it found a lot of points but all concentrated in the superior part of the true Pareto front. In functions ZDT1 and ZDT3, although cMOPSO could not reach the true Pareto front, it obtained good approximations to the whole front. For this reason, cMOPSO obtained better values than sMOPSO with respect to the IGD measure.

Table 1 shows that, with respect to the SCC measure, the use of fitness inheritance decreases the quality of the results as we increase p_i. However, with respect to the IGD measure all the approaches with fitness inheritance have almost the same median and mean values that the approach without inheritance, except in function ZDT3. In this way, we can conclude that the approaches with fitness inheritance obtained as good approximations to the corresponding true Pareto front, as the approach without inheritance. In fact, it is very interesting to note that, in some cases, when the value of p_i is low, the quality of the results with respect to the SCC measure is better than the algorithm without fitness inheritance. This is the case of functions ZDT1 (p_i = 0.1), ZDT3 (p_i = 0.1) and ZDT4 (p_i = 0.1,0.2). This effect in the results may be a product of the diversity that it is introduced by a particle that has inherited the objective values of a leader. On the other hand, the reduction in the number of function evaluations is always greater than the degradation in the quality of results, on average, with respect to the SCC measure. For example, in functions ZDT1, ZDT2 and ZDT3, the worst results were obtained by the approach with p_i = 0.4, reducing the quality of the results in a 14.1%, 13.5% and 13.2%, respectively. In function ZDT4, the worst results were obtained by the approach with p_i = 0.3 reducing the quality of the results in a 25%. In fact, even in the worst case with a saving of 40% of evaluations, the results of the approaches with fitness inheritance are better than the results of the other PSO-based approaches.

Since, in general, the performance with respect to the IGD of all the approaches with inheritance in very similar, we choose the approach with the worst results with respect

Test Function ZDT1								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	best	93	37	84	94	94	78	86
	median	58	7	74	79	72	69	65
	worst	23	1	22	44	27	25	36
	mean	59	8	71	**77**	64	62	61
	std. dev.	24.2	7.9	13.6	14.5	19.8	16.4	14.8
IGD	best	0.0031	0.0016	0.0009	0.0009	0.0009	0.0009	0.0009
	median	0.0260	0.0029	0.0010	0.0009	0.0010	0.0010	0.0010
	worst	0.0448	0.0041	0.0011	0.0010	0.0011	0.0012	0.0087
	mean	0.0269	0.0030	0.0010	**0.0009**	0.0010	0.0010	0.0014
	std. dev.	0.0095	0.0007	0.00005	0.00003	0.00005	0.00008	0.0017

Test Function ZDT2								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	best	1	94	100	100	100	100	97
	median	1	0	91	93	92	88	84
	worst	1	0	60	6	54	23	33
	mean	1	29	**89**	83	86	79	77
	std. dev.	0	38.9	10	22.5	13.1	22.4	20.7
IGD	best	0.0723	0.0030	0.0006	0.0006	0.0006	0.0006	0.0006
	median	0.0723	0.0723	0.0007	0.0007	0.0007	0.0007	0.0007
	worst	0.0723	0.0852	0.0008	0.0011	0.0009	0.0010	0.0009
	mean	0.0723	0.0680	**0.0007**	0.0007	0.0007	0.0007	0.0007
	std. dev.	0.0000	0.0152	0.00005	0.0001	0.00007	0.0001	0.00008

Test Function ZDT3								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	best	89	0	90	88	82	85	92
	median	15	0	72	77	66	67	67
	worst	0	0	18	51	30	27	8
	mean	26	0	68	**73**	65	64	59
	std. dev.	25.4	0	18.2	11.2	13.9	14.5	21.2
IGD	best	0.0023	0.0028	0.0008	0.0008	0.0008	0.0008	0.0008
	median	0.0249	0.0054	0.0008	0.0008	0.0009	0.0008	0.0009
	worst	0.0374	0.0096	0.0021	0.0106	0.0103	0.0014	0.0049
	mean	0.0245	0.0062	0.0009	0.0015	0.0014	**0.0009**	0.0016
	std. dev.	0.0095	0.0020	0.0003	0.0022	0.0021	0.0002	0.0012

Test Function ZDT4								
		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	best	0	0	96	96	96	89	94
	median	0	0	88	83	81	63	77
	worst	0	0	35	50	55	27	11
	mean	0	0	80	81	**81**	60	68
	std. dev.	0	0	16.3	13	12.8	22.7	25.4
IGD	best	0.1541	0.4203	0.0009	0.0009	0.0009	0.0009	0.0009
	median	0.7393	1.6404	0.0010	0.0010	0.0009	0.0010	0.0009
	worst	1.2865	4.1864	0.0010	0.0010	0.0010	0.0010	0.0013
	mean	0.7591	1.8621	0.0010	0.0010	**0.0009**	0.0010	0.0010
	std. dev.	0.3147	0.9357	0.00003	0.00003	0.00003	0.00004	0.00009

Table 1. Obtained results for all the test functions, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with fitness inheritance (p_i=0.1,0.2,0.3,0.4).

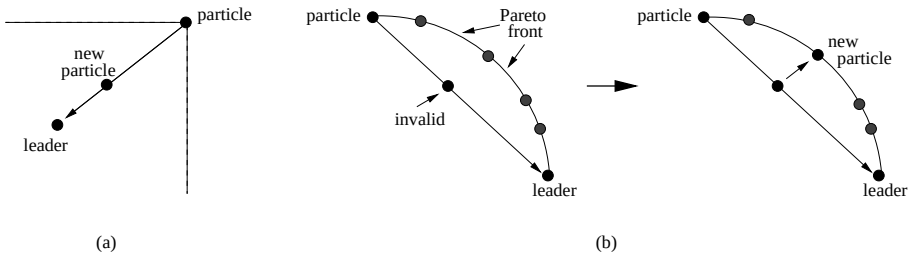

Figure 2. The two possible cases of fitness inheritance: (a) when the leader dominates the particle and (b) when the leader does not dominate the particle.

Test Function ZDT1							
SC(X,	sMOPSO)	cMOPSO)	oMOPSO)	0.1)	0.2)	0.3)	0.4)
sMOPSO	0.00	0.90	0.35	0.25	0.35	0.38	0.41
cMOPSO	0.00	0.00	0.00	0.00	0.00	0.00	0.00
oMOPSO	0.08	0.95	0.00	0.22	0.36	0.46	0.41
0.1	0.08	0.94	0.45	0.00	0.46	0.51	0.52
0.2	0.06	0.94	0.29	0.24	0.00	0.38	0.40
0.3	0.05	0.95	0.21	0.16	0.28	0.00	0.32
0.4	0.05	0.95	0.28	0.16	0.31	0.36	0.00
Test Function ZDT2							
SC(X,	sMOPSO)	cMOPSO)	oMOPSO)	0.1)	0.2)	0.3)	0.4)
sMOPSO	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cMOPSO	0.00	0.00	0.00	0.02	0.01	0.01	0.01
oMOPSO	0.00	0.21	0.00	0.28	0.34	0.28	0.47
0.1	0.00	0.21	0.31	0.00	0.39	0.29	0.51
0.2	0.00	0.21	0.26	0.25	0.00	0.24	0.44
0.3	0.00	0.21	0.35	0.32	0.37	0.00	0.48
0.4	0.00	0.21	0.18	0.15	0.22	0.17	0.00
Test Function ZDT3							
SC(X,	sMOPSO)	cMOPSO)	oMOPSO)	0.1)	0.2)	0.3)	0.4)
sMOPSO	0.00	0.75	0.13	0.14	0.19	0.20	0.18
cMOPSO	0.01	0.00	0.00	0.00	0.00	0.00	0.00
oMOPSO	0.23	0.92	0.00	0.28	0.40	0.44	0.38
0.1	0.25	0.92	0.41	0.00	0.49	0.53	0.45
0.2	0.22	0.92	0.26	0.22	0.00	0.38	0.31
0.3	0.22	0.92	0.20	0.17	0.29	0.00	0.26
0.4	0.23	0.92	0.26	0.22	0.34	0.39	0.00
Test Function ZDT4							
SC(X,	sMOPSO)	cMOPSO)	oMOPSO)	0.1)	0.2)	0.3)	0.4)
sMOPSO	0.00	1.00	0.00	0.00	0.00	0.00	0.00
cMOPSO	0.00	0.00	0.00	0.00	0.00	0.00	0.00
oMOPSO	0.00	0.00	0.00	0.38	0.32	0.54	0.42
0.1	0.00	0.00	0.21	0.00	0.24	0.48	0.33
0.2	0.00	0.00	0.25	0.36	0.00	0.53	0.39
0.3	0.00	0.00	0.11	0.15	0.12	0.00	0.20
0.4	0.00	0.00	0.17	0.24	0.21	0.40	0.00

Table 2. Obtained results for the test function ZDT4, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with fitness inheritance (p_i=0.1,0.2,0.3,0.4).

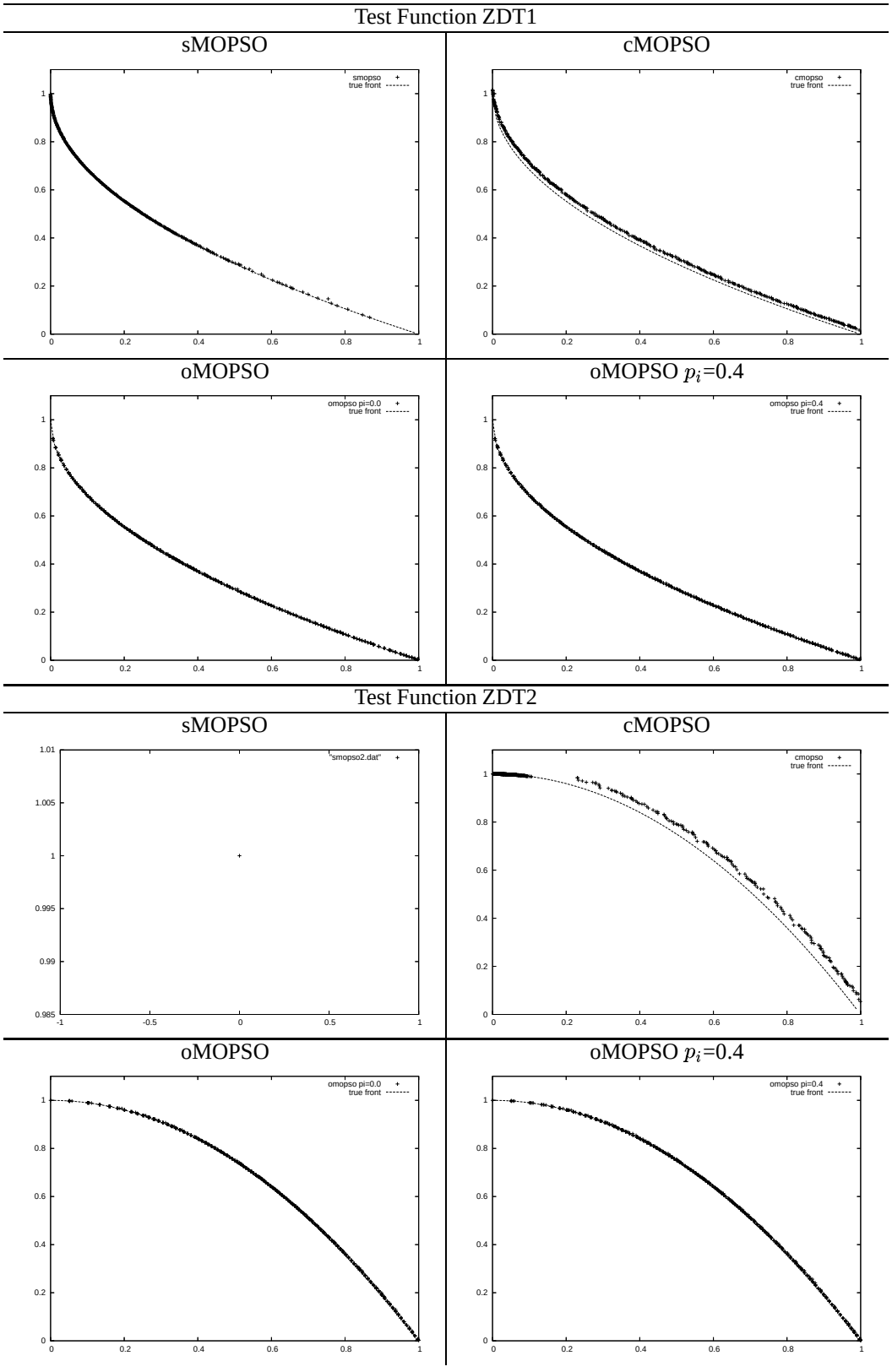

Figure 3. Pareto fronts obtained by sMOPSO, cMOPSO, oMOPSO, and oMOPSO with inheritance proportion of 0.4, for functions ZDT1 y ZDT2.

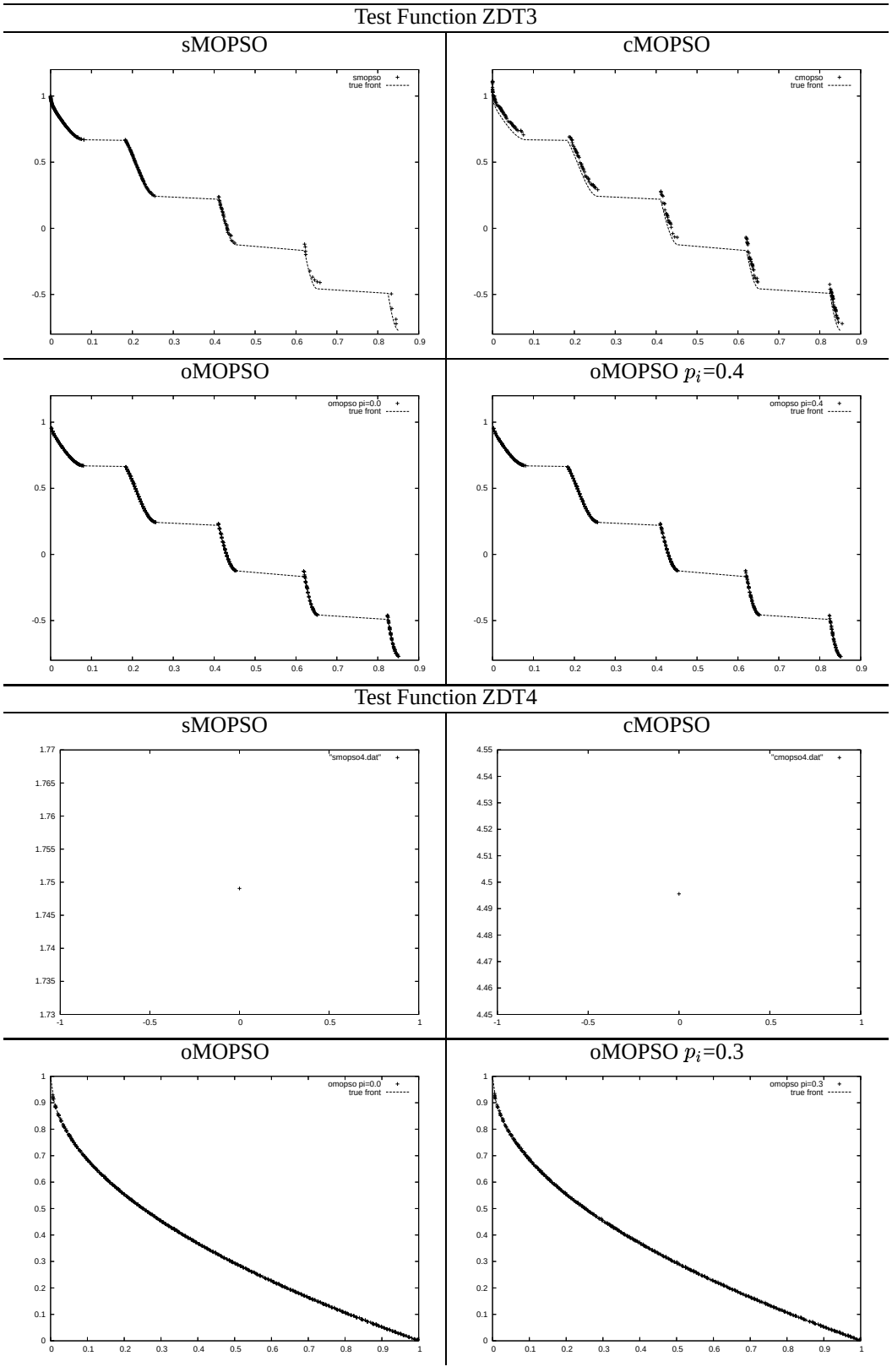

Figure 4. Pareto fronts obtained by sMOPSO, cMOPSO, oMOPSO, and oMOPSO with inheritance proportion of 0.4, for function ZDT3, and with inheritance proportion of 0.3, for function ZDT4.

to the SCC measure to be represented by means of its corresponding Pareto front in Figures 3 and 4. Thus, in functions ZDT1, ZDT2 and ZDT3, we show the Pareto front corresponding to the approach with p_i = 0.4 and, in the case of function ZDT4, the Pareto front corresponding to the approach with p_i = 0.3. We can see in Figures 3 and 4 that the approaches with inheritance do not lose the Pareto front even in cases where the true Pareto front is not convex or discontinuous. As we described in Section 3, we implemented a simple mechanism (shown in Figure 2) in order to avoid producing invalid particles in the cases of Pareto fronts non-convex or discontinuous, this is the main reason why our algorithm does not have problems with these types of test functions.

Given the definition of the binary Set Coverage (*SC*) measure, we will conclude that an approach X is *relatively better* than the approach Y, when $SC(Y, X) > SC(X, Y)$. In this way, from the results shown in Table 2, we can conclude that sMOPSO is relatively better than oMOPSO only in function ZDT1, and that oMOPSO is relatively better than sMOPSO in function ZDT3 and than cMOPSO in functions ZDT1, ZDT2 and ZDT3. Since our algorithm has some problems losing the extreme points of the Pareto fronts (due to the use of the ϵ-dominance in external archive), it is unable to dominate the points generated by sMOPSO and cMOPSO in functions ZDT2 and ZDT4.

On the other hand, our algorithm without inheritance (oMOPSO) is relatively better than almost all the approaches with fitness inheritance in all functions, except for the case when p_i = 0.1. The approach with p_i = 0.1 is relatively better than the approach without inheritance in three of the four functions: ZDT1, ZDT2 and ZDT3. This conclusion agrees with the results obtained with the unary measures. The only function in which the algorithm without inheritance is relatively better than all the approaches with fitness inheritance is ZDT4. These results seem indicate that the fitness inheritance approach is more useful when the decision space of the problem has a high dimension.

6. CONCLUSIONS AND FUTURE WORK

We have proposed an approach to incorporate the concept of fitness inheritance to a real-coded MOPSO that we proposed in some of our previous work. The proposed technique was tested using several multi-objective test functions and compared against two other PSO-based multi-objective algorithms. From our results, we conclude that the efficiency enhancement technique proposed in this paper reduces the computational cost without decreasing the quality of the results in a significant way. Also, the fitness inheritance technique used in our approach is able to generate non-convex and discontinuous Pareto fronts. On the other hand, the obtained results seem indicate that the proposed enhancement technique is more useful, that is, the quality of the results is less affected, when the decision space is high-dimensional. As part of our future work, we plan to explore alternative ways to incorporate fitness inheritance so that we can have a more important reduction in the number of evaluations at the expense of a lower degradation in the quality of the results.

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

7. REFERENCES

[1] Jian-Jung Chen, David E. Goldberg, Shinn-Ying Ho, and Kumara Sastry. Fitness Inheritance in Multi-Objective Optimization. In *GECCO'2002*, pages 319–326. Morgan Kaufmann Publishers, 2002.

[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] 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.

[4] 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.

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

[6] Kumara Sastry, David E. Goldberg, and Martin Pelikan. Don't Evaluate, Inherit. In *GECCO 2001*, pages 551–558. Morgan Kaufmann, 7-11 2001.

[7] 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.

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

[9] 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. Springer-Verlag, LNCS Vol. 3102, 2004.

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

[11] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In *PPSN V*, pages 292–301. Springer-Verlag, 1998.