A New Proposal for Multiobjective Optimization using Particle Swarm Optimization and Rough Sets Theory

Luis V. Santana-Quintero[1], Noel Ramírez-Santiago[1], Carlos A. Coello Coello[1], Julián Molina Luque[2], and Alfredo García Hernández-Díaz[3]

[1] CINVESTAV-IPN,Electrical Engineering Department, Computer Science Section,México D.F. 07300, MÉXICO,
lvspenny@hotmail.com, santiago@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

[2] University of Malaga, Department of Applied Economics (Mathematics), SPAIN
julian.molina@uma.es

[3] Pablo de Olavide University, Department of Quantitative Methods, Seville, SPAIN
agarher@upo.es

Abstract. This paper presents a new multi-objective evolutionary algorithm which consists of a hybrid between a particle swarm optimization approach and some concepts from rough sets theory. The main idea of the approach is to combine the high convergence rate of the particle swarm optimization algorithm with a local search approach based on rough sets that is able to spread the nondominated solutions found, so that a good distribution along the Pareto front is achieved. Our proposed approach is able to converge in several test functions of 10 to 30 decision variables with only 4,000 fitness function evaluations. This is a very low number of evaluations if compared with today's standards in the specialized literature. Our proposed approach was validated using nine standard test functions commonly adopted in the specialized literature. Our results were compared with respect to a multi-objective evolutionary algorithm that is representative of the state-of-the-art in the area: the NSGA-II.

1 Introduction

In this paper, we propose a new multi-objective evolutionary algorithm (MOEA) which consists of a hybrid between a particle swarm optimization (PSO) approach and rough sets theory. The main aim of this work is to design a MOEA that can produce a reasonably good approximation of the true Pareto front of a problem with a relatively low number of fitness function evaluations (no more than 5000 fitness function evaluations). PSO is a bio-inspired metaheuristic that was proposed by James Kennedy and Russell Eberhart in the mid-1990s [1], and which is inspired on the choreography of a bird flock. In PSO, each solution is represented by a particle. Particles group in "swarms" (there can be either one swarm or several in one population) and the evolution of the swarm to the optimal solutions is achieved by a velocity equation. This equation is composed of

three elements: a velocity inertia, a cognitive component and a social component. Depending on the topology adopted (i.e., one swarm or multiple swarms), each particle can be affected by either the best local and/or the best global particle in its swarm. PSO has been found to be a very successful optimization approach both in single-objective and in multi-objective problems [1, 2]. However, so far, the high convergence rate of PSO has not been properly exploited by researchers, since most of the current multi-objective PSOs (MOPSOs) perform 20,000 fitness function evaluations or more in test functions such as the ones adopted in this paper. The main reason for this is that despite its high convergence rate, PSO normally has difficulties to achieve a good distribution of solutions with a low number of evaluations. That is why we adopted rough sets theory (which can be useful at finding solutions within the neighborhood of a reference set) in this paper in order to have a local optimizer whose computational cost is low.

2 Particle Swarm Optimization

In the PSO algorithm, the particles (including the *pbest*) are randomly initialized at the beginning of the search process. Next, the fittest particle from the swarm is identified and assigned to the *gbest* solution (i.e., the global best, or best particle found so far). After that, the swarm flies through the search space (in k dimensions, in the general case). The flight function adopted by PSO is determined by equation (1), which updates the position and fitness of the particle (see equation (2)). The new fitness is compared with respect to the particle's *pbest* position. If it is better, then it replaces the *pbest* (i.e., the personal best, or the best value that has been found for this particle so far). This procedure is repeated for every particle in the swarm until the termination criteria is reached.

$$v_{i,k} = w \cdot v_{i,k} + c_1 \cdot U(0,1)(pbest_{i,k} - x_{i,k}) + c_2 \cdot U(0,1)(gbest_k - x_{i,k}); \quad (1)$$

$$x_{i,k} = x_{i,k} + v_{i,k} \quad (2)$$

where c_1 and c_2 are constants that indicate the attraction from the *pbest* or *gbest* position, respectively; w refers to the velocity inertia of the previous movement; $x_i = (x_{i1}, x_{i2}, ..., x_{ik})$ represents the $i-th$ particle. U(0,1) denotes a uniformly random number generated in the range (0,1).

There are plenty of proposals to extend PSO for dealing with multiple objectives (see for example [2, 3]). A survey of MOPSOs is beyond the scope of this paper, but interested readers may refer to [4].

3 Rough Sets Theory

Rough sets theory is a new mathematical approach to imperfect knowledge that was originally proposed by Pawlak [5]. The main idea of this approach is explained next. Let's assume that we are given a set of objects S called the universe and an indiscernibility relation $R \subseteq S \times S$, representing our lack of knowledge

about elements of S (in our case, R is simply an equivalence relation based on a grid over the feasible set; this is, just a division of the feasible set in (hyper)-rectangles). Let X be a subset of S. We want to characterize the set X with respect to R. The way rough sets theory expresses vagueness is employing a boundary region of the set X. If the boundary region of a set is empty it means that the set is *crisp*; otherwise, the set is *rough* (inexact). A nonempty boundary region of a set means that our knowledge about the set is not enough to define the set precisely. Then, each element in S is classified as *certainly* inside X if it belongs to the lower approximation or *partially* (*probably*) inside X if it belongs to the upper approximation. The *boundary* is the difference of these two sets, and the bigger the boundary the worse the knowledge we have of set X. On the other hand, the more precise is the grid implicity used to define the indiscernibility relation R, the smaller the boundary regions are. But, the more precise is the grid, the bigger the number of elements in S, and then, the more complex the problem becomes. Our aim is to use rough sets to explore the neighborhood of a set of reference solutions (the nondominated solutions found by our PSO-based MOEA), so that we can spread such solutions along the Pareto front. For this sake, it is required to design a grid and decide which elements of S (that we will call *atoms* and will be just rectangular portions of decision variable space) are inside the Pareto optimal set and which are not. Once we have the *efficient atoms*, we will intensify the search over these atoms. Note however, that the precision of the grid has an impact on both the computational cost (the more precise the grid, the higher its cost) and on effectiveness (the less precise the grid, the less knowledge we can obtain from it). Evidently, in our approach, we will try to generate a grid that is not so computationally expensive but that offers a reasonably good knowledge about the Pareto optimal set. Once this grid is built, it becomes relatively straightforward to generate more points on the *efficient atoms*, as these atoms are built in decision variable space. Note however, that the use of rough sets requires not only a set of nondominated solutions, but also another one of dominated solutions that are close to being nondominated. This second set is required in order to intensify the search. Thus, our MOEA will be modified in order to keep this second set, which is not normally required in evolutionary multiobjective optimization.

4 Pareto-adaptive ϵ-dominance

Our approach also adopts a variant of ε-dominance [6] that we call *pa*ε-dominance. The details of *pa*ε-dominance are omitted due to space constraints (see [7] for further information), but its main difference with respect to the original proposal is that in this case the hyper-grid generated adapts the sizes of the boxes to certain geometrical characteristics of the Pareto front (e.g., almost horizontal or vertical portions of the Pareto front) as to increase the number of solutions retained in the grid. Thus, this scheme maintains the good properties of ε-dominance but improves on its main weaknesses.

5 Our Proposed Approach

Our proposed approach, called PSOMORSA (Particle Swarm Optimization for Multiobjective Optimization with Rough Sets), is divided in two phases, and each of them consumes a fixed number of fitness function evaluations. During Phase I, our PSO-based MOEA is applied for 2000 fitness function evaluations. During Phase II, a local search procedure based on rough sets theory is applied for another 2000 fitness function evaluations, in order to improve the solutions (i.e., spread them along the Pareto front) produced at the previous phase. Each of these two phases is described next in more detail.

5.1 Phase I : Particle Swarm Optimization

Our proposed PSO-based approach adopts a very small population size ($P = 5$ particles). The leader is determined using a very simple criterion: the first N particles (N is the number of objectives of the problem) are guided by the best particle in each objective, considered separately. The remainder $P - N$ particles are adopted to build an approximation of the ideal vector. Then, we identify the individual which is closest to this ideal vector and such individual becomes the leader for the remainder $P - N$ particles. The purpose of these selection criteria is twofold: first, we aim to approximate the optimum for each separate objective, by exploiting the high convergence rate of PSO in single-objective optimization. The second purpose of our selection rules is to encourage convergence towards the "knee" of the Pareto front (considering the bi-objective case). We found that the use of rough sets can generate the entire Pareto front even if only one nondominated solution is available in the Pareto front, and in disconnected Pareto fronts it is required only one nondominated solution per each discontinuos segment of the Pareto front.

Algorithm 1 shows the pseudocode of Phase I from our proposed approach. First, we randomly generate 5 individuals. In the $getLeaders()$ function, we identify the best particles in each objective and the closest particle to the ideal vector. Those particles (the leaders) are stored in the set L. Then the $getLeader(\boldsymbol{x})$ function returns the position of the leader from the set L. Then, we perform the flight in order to obtain a new particle. If this solution is beyond the allowable bounds for a decision variable, then we adopt the $BLX - \alpha$ recombination operator [8], and a new vector solution $Z = (z_1, z_2, ..., z_d)$ is generated, where $z_i \in [c_{min} - I\alpha, c_{max} + I\alpha]$; $c_{max} = max(a, b)$, $c_{min} = min(a, b)$, $I = c_{max} - c_{min}$, $\alpha = 0.5$, $a = L_g$ (the leader of the particle) and $b = pbest$ (i.e., the personal best of the particle). Note that the use of a recombination operator is not a common practice in PSO, and some people may consider our approach as a PSO-variant because of that. PSO does not use a specific mutation operator (the variation of the factors of the flight equation may compensate for that). However, it has become common practice in MOPSOs to adopt some sort of mutation (or turbulence) operator that improves the exploration capabilities of PSO [2, 3]. The use of a mutation operator is normally simpler (and easier) than varying the factors

Algorithm 1: Algorithm for the Phase I of our approach.

```
1  begin
2      Initialize Population (P) with randomly generated solutions;
3      getLeaders();
4      repeat
5          for i = 1 to P do
6              g = getLeader(i);
7              for d = 1 to k do
8                  /* L_{g,d} is the leader of particle i */;
9                  v_{i,d} = w · v_{i,d} + c1 · U(0,1)(p_{i,d} − x_{i,d}) + c2 · U(0,1)(L_{g,d} − x_{i,d});
10                 x_{i,d} = x_{i,d} + v_{i,d};
11             end
12             if x_i ∉ search space then
13                 x_i = BLX − α(x_i);
14             end
15             if U(0,1) < p_m then
16                 x_i =Mutate(x_i);
17             end
18             if x_i is nondominated then
19                 for d = 1 to k do
20                     p_{i,d} = x_{i,d};;
21                 end
22             end
23         end
24         getLeaders();
25         Add nondominated solutions into secondary population
26     until MaxIter ;
27 end
```

of the flight equation and therefore its extended use. We adopted Parameter-Based Mutation [9] in our approach with $p_m = 1/n$. Our proposed approach also uses an external archive (also called secondary population). In order to include a solution into this external archive, it is compared with respect to each member already contained in the archive using the $pa\epsilon$-dominance grid [7]. And a third population (called DS) stores the dominated points needed for Phase II. Every removed point from the secondary population (also called ES) is included into the third population. If this third population reaches a size of 100 points, a $pa\epsilon$-dominance grid will be created in order to ensure a good distribution of dominated points.

5.2 Phase II : Local Search using Rough Sets

The Rough Sets Phase departs from the two sets obtained from Phase I (ES, which contains the nondominated solutions, and DS, which contains the dominated solutions). The main loop of the second phase is the following:

1. From the set ES, we choose $NumEff$ points previously unselected. If we do not have enough unselected points, we choose the rest randomly from the set ES.
2. We choose from the set DS, $NumDom$ points previously unselected (complete randomly as before).
3. Do a Rough Sets iteration, to approximate the boundary between the Pareto front and the rest of the feasible set. This information is used to intensify

the search in the area where the nondominated points reside, while refusing the finding of more points in the dominated area.

The dominated and nondominated points are both stored in the set $Items$ and the rough sets iteration is the following:

1. **Range Initialization:** For each decision variable i, we compute and sort (from the smallest to the highest) the different values contained in $Items$. Then, we have the set $Range_i$, for each i. By combining all these sets we produce a (non-uniform) grid in decision variable space.
2. **Compute Atoms:** We compute $NumEff$ rectangular atoms centered in the $NumEff$ efficient points selected. To build a rectangular atom associated to a nondominated point $x^e \in Items$ we compute the following upper and lower bounds for each decision variable i:
   - Lower Bound i: Middle point between x_i^e and the previous value in the set $Range_i$.
   - Upper Bound i: Middle point between x_i^e and the following value in the set $Range_i$.

   In both cases, if there are no previous or subsequent values in $Range_i$, we consider the absolute lower or upper bound of variable i. This setting allows the method to explore closer to the feasible set boundaries.
3. **Generate Offspring:** Inside each atom we randomly generate $Offspring$ new points. Each of these points is sent to the set ES (we use the $pa\epsilon$-dominance grid for that sake) to check if it must be included as a new nondominated point. If any point in ES is dominated by this new point, it is sent to the set DS.

6 Analysis of Results

In order to validate our proposed approach, we compare results with respect to the NSGA-II [9], which is a MOEA representative of the state-of-the-art in the area. The first phase of our approach uses three parameters: population size (P), leaders number (N), mutation probability (P_m), plus the traditional PSO parameters (w, c_1, c_2). On the other hand, the second phase uses three more parameters: number of points randomly generated inside each atom ($Offspring$), number of atoms per generations ($NumEff$) and the number of dominated points considered to generate the atoms ($NumDom$). Finally, the minimum number of nondominated points needed to generate the $pa\epsilon$-dominance grid is set to 100 for all problems. Our approach was validated using 9 test problems: 5 problems from the **ZDT** set [10] and 4 from the **DTLZ** set [11]. The detailed description of these test functions was omitted due to space restrictions (see [10, 11] for further information). However, all of these test functions are unconstrained, minimization and have between 10 and 30 decision variables. In all cases, the parameters of our approach were set as follows: $P = 5$, $N = k + 1$ (k = number of objective functions), $P_m = 1/n$ (n = number of decision variables), $w = 0.3$, $c_1 = 0.1$, $c_2 = 1.4$, $Offspring = 1$, $NumEff = 2$ and $NumDom = 10$. The NSGA-II

used the following parameters: crossover rate = 0.9, mutation rate = 1/n, η_c = 15, η_m = 20, population size = 100 and maximum number of generations = 40. The population size of the NSGA-II is the same as the size of the grid of our approach. In order to allow a fair comparison of results, both approaches adopted real-numbers encoding and performed 4,000 fitness function evaluations per run. Three performance measures were adopted in order to allow a quantitative assessment of our results: (1) Inverted Generational Distance (**IGD**), which is a variation of a metric proposed by Van Veldhuizen [12] in which the true Pareto is used as a reference; Spread (**S**), proposed by Deb et al. [13], which measures both progress towards the Pareto-optimal front and the extent of spread; and (3) Two Set Coverage (**SC**), proposed by Zitzler et al. [10], which performs a relative coverage comparison of two sets. For each test problem, 30 independent runs were performed and the results reported in Table 1 correspond to the mean and standard deviation of the performance metrics (IGD, S and SC). We show in boldface the best mean values per test function. It can be observed that in the ZDT's test problems our approach produced the best results with respect to both IGD and SC in all cases. Remarkably, our approach also outperformed the NSGA-II with respect to the spread metric in all but one case (ZDT3). In the DTLZ's test problems, the NSGA-II outperformed our approach in one case with respect to IGD, in all cases with respect to Spread and in 3 (out of 4) cases with respect to SC. Figures 1 and 2 show the graphical results produced by the PSOMORSA and NSGA-II for all the test problems adopted. The solutions displayed correspond to the median result with respect to the IGD metric. The true Pareto front (obtained by enumeration) is shown with a continuous line and the approximation produced by each algorithm is shown with circles. In Figures 1 and 2, we can clearly see that in problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, the NSGA-II is very far from the true Pareto front, whereas our PSOMORSA is very close to the true Pareto front after only 4,000 fitness function evaluations (except for ZDT4). Graphically, the results are not entirely clear for the DTLZ test problems. However, if we pay attention to the scale, it will be evident that, in most cases, our approach has several points closer to the true Pareto front than the NSGA-II. Nevertheless, due to the better spread of the NSGA-II, there are a few points that dominate several of the solutions produced by our approach and therefore the superiority of the NSGA-II with respect to the SC and S metrics. The poor performance of our approach in the DTLZ's test problems is caused in the PSO selection process because we select one particle that helps to optimize the third objective function and the ideal vector is optimized with two other particles, causing that the convergence rate gets lower than expected in problems with three or more objectives.

7 Conclusions and Future Work

We have introduced a new hybrid between a MOEA based on PSO and a local search mechanism based on rough sets theory. This hybrid aims to combine the high convergence rate of PSO with the good neighborhood exploration per-

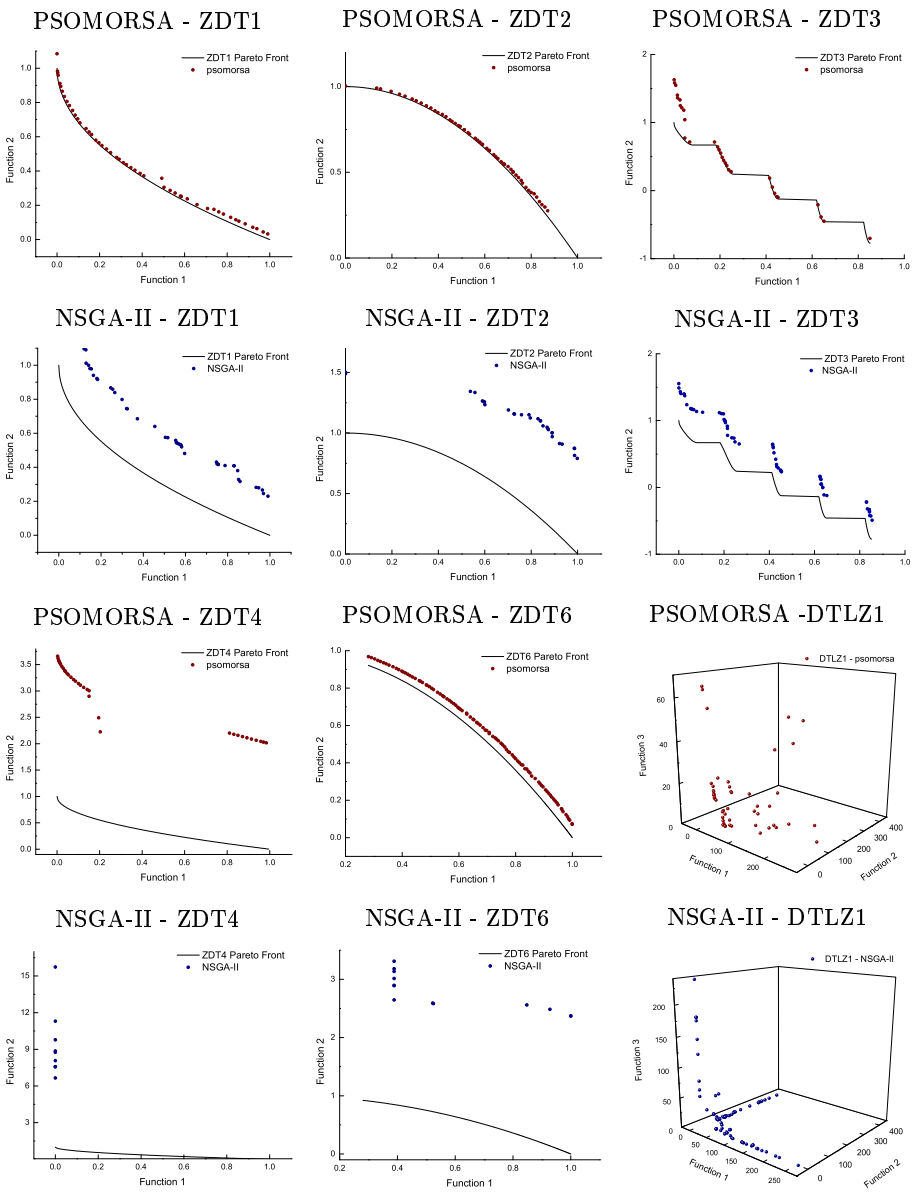

Fig. 1. Pareto fronts generated by PSOMORSA and NSGA-II for ZDT's and DTLZ1.

	IGD				S				SC			
Function	PSOMORSA		NSGA-II		PSOMORSA		NSGA-II		PSOMORSA		NSGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
ZDT1	**0.0009**	0.0005	0.0097	0.0019	**0.4827**	0.1306	0.5603	0.0483	**0.0222**	0.0312	0.9332	0.0355
ZDT2	**0.0036**	0.0057	0.0223	0.0064	**0.6176**	0.2199	0.7130	0.1114	**0.0038**	0.0127	0.8784	0.1645
ZDT3	**0.0043**	0.0014	0.0155	0.0020	0.7761	0.0656	**0.7441**	0.0456	**0.0608**	0.0656	0.9062	0.0555
ZDT4	**0.1265**	0.0371	0.4297	0.1304	**0.9590**	0.0424	0.9718	0.0412	**0.0342**	0.0557	0.3065	0.1560
ZDT6	**0.0009**	0.0003	0.0420	0.0041	**0.7336**	0.1271	0.8706	0.0817	**0.0012**	0.0066	0.9333	0.2034
DTLZ1	**0.5157**	0.1217	0.7318	0.2062	0.9983	0.0015	**0.9972**	0.0011	0.3208	0.1945	**0.3142**	0.1839
DTLZ2	**0.0004**	0.0001	**0.0004**	0.0000	0.5676	0.0747	**0.3188**	0.0440	**0.1418**	0.1485	0.1913	0.1131
DTLZ3	**1.1681**	0.3063	1.4228	0.2690	0.9990	0.0012	**0.9986**	0.0012	0.5220	0.2600	**0.1545**	0.1019
DTLZ4	0.0221	0.0038	**0.0096**	0.0025	0.7682	0.1055	**0.6676**	0.1250	0.8537	0.1626	**0.0084**	0.0272

Table 1. Comparison of results between our approach (called PSOMORSA) and the NSGA-II for the nine test problems adopted.

formed by the rough sets algorithm. Our proposed approach produced results that are competitive with respect to the NSGA-II in problems whose dimensionality goes from 10 up to 30 decision variables, while performing only 4,000 fitness function evaluations. Although our results are still preliminary, they are very encouraging, since they seem to indicate that our proposed approach could be a viable alternative for solving real-world problems in which the cost of a single fitness function evaluation is very high (e.g., in aeronautics). As part of our future work, we intend to improve the performance of the PSO approach adopted. Particularly, the selection of the appropriate leader is an issue that deserves further study.

Acknowledgments: The third author acknowledges support from CONACyT through project number 45683-Y.

References

1. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, California, USA (2001)
2. Coello Coello, C.A., Toscano Pulido, G., Salazar Lechuga, M.: Handling Multiple Objectives With Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation **8**(3) (2004) 256–279
3. Mostaghim, S., Teich, J.: Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 26–33
4. Reyes-Sierra, M., Coello Coello, C.A.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research **2**(3) (2006) 287–308
5. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences **11**(1) (1982) 341–356
6. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation **10**(3) (2002) 263–282
7. Hernández-Díaz, A.G., Santana-Quintero, L.V., Coello, C.A.C., Molina, J.: Pareto-adaptive ε-dominance. Technical Report EVOCINV-02-2006, Evolutionary Com-

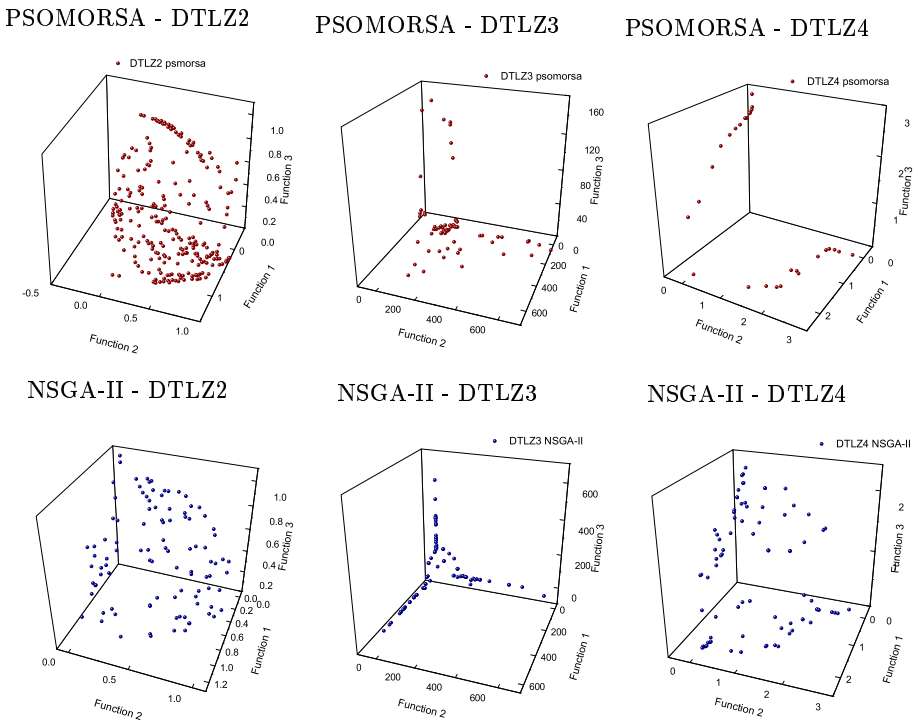

Fig. 2. Pareto fronts generated by PSOMORSA and NSGA-II for DTLZ2, DTLZ3 and DTLZ4.

putation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México (2006)
8. Eshelman, L.J., Schaffer, J.D.: Real-coded Genetic Algorithms and Interval-Schemata. In Whitley, L.D., ed.: Foundations of Genetic Algorithms 2. Morgan Kaufmann Publishers, San Mateo, California (1993) 187–202
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
10. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms:Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
11. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., Jain, L., Goldberg, R., eds.: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications. Springer, USA (2005) 105–145
12. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
13. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001) ISBN 0-471-87339-X.