A Multi-objective Particle Swarm Optimizer Hybridized with Scatter Search

Luis V. Santana-Quintero, Noel Ramírez, Carlos Coello Coello

CINVESTAV-IPN, Electrical Engineering Department, Computer Science Section,
Av. IPN No. 2508, México D.F. 07360, MÉXICO,
lsantana@computacion.cs.cinvestav.mx
santiago@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. This paper presents a new multi-objective evolutionary algorithm which consists of a hybrid between a particle swarm optimization (PSO) approach and scatter search. 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 scatter search. We propose a new leader selection scheme for PSO, which aims to accelerate convergence. Upon applying PSO, scatter search acts as a local search scheme, improving the spread of the nondominated solutions found so far. Thus, the hybrid constitutes an efficient multi-objective evolutionary algorithm, which can produce reasonably good approximations of the Pareto fronts of multi-objective problems of high dimensionality, while only performing 4,000 fitness function evaluations. Our proposed approach is validated using ten standard test functions commonly adopted in the specialized literature. Our results are 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

Most real world problems involve the simultaneous optimization of two or more (often conflicting) objectives. The solution of such problems (called "multi-objective") is different from that of a single-objective optimization problem. The main difference is that multi-objective optimization problems normally have not one but a set of solutions which are all equally good. 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. In the past, a wide variety of evolutionary algorithms (EA's) have been used to solve multi-objective optimization problems [2]. In this paper, we propose a new hybrid multi-objective evolutionary algorithm based on particle swarm optimization (PSO) and scatter search (SS). PSO is a bio-inspired optimization algorithm that was proposed by James Kennedy and Russell Eberhart in the mid-1990s [10], and which is inspired on the choreography of a bird flock. PSO has been found to be a very successful optimization approach both

in single-objective and in multi-objective problems [1, 10]. 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 "*pbest*" and a social component "*gbest*". 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 normally has difficulties to achieve a good distribution of solutions with a low number of evaluations. That is why we adopted scatter search (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. SS is an evolutionary method that was originally proposed in the 1970s by Fred Glover [7] for combining decision rules and problem constraints. This method uses strategies for combining solution vectors that have been found effective during the search (the so called "reference set") [12]. SS has been successfully applied to hard optimization problems, and it constitutes a very flexible heuristic, since it can be implemented in a variety of ways, offering numerous alternatives for exploiting its fundamental ideas. The remainder of this paper is organized as follows. Section 2 provides a brief introduction to particle swarm optimization. In Section 3 we analyze the scatter search components. Section 4 describes our proposed approach. Our comparison of results is provided in Section 5. Our conclusions and some possible paths for future research are provided in Section 6.

2 Particle Swarm Optimization (PSO)

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 the 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 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 within the range (0,1); $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$ represents the

rate change (velocity) of particle i. The equation (1) describes the velocity that is constantly updated by each particle and equation (2) updates the position of the particle in each decision variable. There are plenty of proposals to extend PSO for dealing with multiple objectives (see for example [1]).

3 Scatter Search

As indicated before, Scatter Search was first introduced in 1977 by Fred Glover [7] as a method that uses a succession of coordinated initializations to generate solutions. In 1994 [8], the range of applications of SS was expanded to nonlinear optimization problems, binary and permutation problems. Finally, in 1998 a new publication [9] on scatter search triggered the interest of researchers and practitioners, who translated these ideas into different computer implementations to solve a variety of problems.

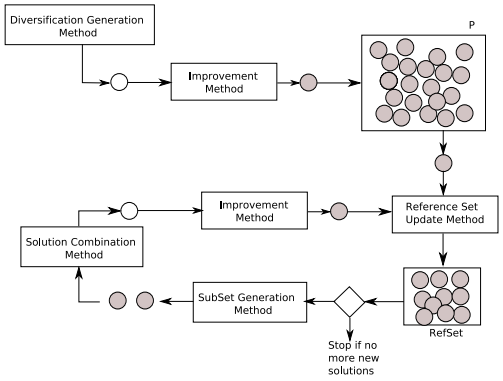

Fig. 1. Scatter Search Scheme

In Figure 1, the *Diversification Generation Method* (generates a scatter solutions set) and *Improvement Method* (makes a local search, and aims to improve the solutions) are initially applied to all the solutions in the P set. A $RefSet$ set is generated based on the P set, in $RefSet$ is contained some of the best solutions and diversity solutions founded at the moment. The *Subset Generation Method* takes reference solutions as its input to produce solution subsets to be combined, the solution subsets contain two or more solutions from $RefSet$. Then, the *Combination Method* is applied to the solution subsets to get new solutions. We try to improve the generated solutions with the *Improvement Method* and the result of the improvement is handed by the *Reference Set Update Method*. This method applies rules regarding the admission to the reference set of solutions called $RefSet$.

4 Our Proposed Approach

Our proposed approach, called MOPSOSS (Multi-objective Optimization using Particle Swarm Optimization with Scatter Search), 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 scatter search 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.

4.1 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. The ideal vector $(f_1^*, f_2^*, \ldots, f_n^*)$ where f_i^* is the best solution found at the i function at the moment. Then, we identify the individual which is closest to this ideal vector (using an euclidian distance) 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).

Algorithm 1 shows the pseudocode of Phase I from our proposed approach. First, we randomly generate the first population, but in the population we need at least the same number of individuals as the number of objectives plus one these last individual is needed to become the ideal vector, for this purpose we choose 5 individuals to perform the tests. 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 for a particle x. 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 [6], and a new vector solution $Z = (z_1, z_2, ..., z_d)$ is generated, where $z_i \in [c_{min} - I\alpha, c_{max} + I\alpha]$; $c_{max} = max(a_i, b_i)$, $c_{min} = min(a_i, b_i)$, $I = c_{max} - c_{min}$, $\alpha = 0.5$, $a = L_g$ (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 [1, 14]. The use of a mutation operator is normally simpler

(and easier) than varying the factors of the flight equation and therefore its extended use. We adopted Parameter-Based Mutation [4] 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 ϵ-dominance grid [13]. Every solution in the archive is assigned an identification array ($\mathbf{B} = (B_1, B_2, \ldots, B_d)^T$, where d is the total number of objectives) as follows:

$$B_j(f) = \begin{cases} (\lfloor (f_j - f_j^{min})/\epsilon_j \rfloor), \text{ for minimizing } f_j; \\ (\lceil (f_j - f_j^{min})/\epsilon_j \rceil), \text{ for maximizing } f_j. \end{cases}$$

where: f_j^{min} is the minimum possible value of the j-th objective and ϵ_j is the allowable tolerance in the j-th objective [13]. The identification array divides the whole objective space into hyper-boxes, each having ϵ_j size in the j-th objective. Using this procedure, we can guarantee the generation of a well-distributed set of nondominated solutions. Also, the value of ϵ adopted (defined by the user) regulates the size of the external archive.

Algorithm 1: Phase I - PSO Algorithm

```
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 nVariables 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(L_g, p_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 nVariables do
20                     p_{i,d} = x_{i,d}
21                 end
22             end
23         end
24         getLeaders()
25     until MaxIter
26 end
```

Any member that is removed from the secondary population is included in the third population. The third population stores the dominated points needed for the Phase II.

4.2 Phase II: Scatter Search

Upon termination of Phase I (2000 fitness function evaluations), we start Phase II, which departs from the nondominated set generated in Phase I. This set is contained within the secondary population. We also have the dominated set, which is contained within the third population. From the nondominated set we choose $MaxScatterSolutions$ points. These particles have to be scattered in the nondominated set, so we choose them based on a distance L_α, which is determined by equation 3:

$$L_\alpha(x) = \underset{i=1,\ldots,p}{Max} \left\{ \frac{f_i^{max}(x) - f_i(x)}{f_i^{max}(x) - f_i^{min}(x)} \right\} \tag{3}$$

Generalizing, to obtain the scatter solutions set among the nondominated set, we use equation 4:

$$L_{set} = \underset{\forall u \in U}{Max} \left\{ \underset{\forall v \in V}{Min} \left\{ \underset{i=1,\ldots,p}{Max} \left\{ \frac{|f_{vi}(x) - f_{ui}(x)|}{f_i^{max}(x) - f_i^{min}(x)} \right\} \right\} \right\} \tag{4}$$

where L_{set} is the Leaders set, U is the nondominated set and V contains the scatter solutions set, f_i^{max} and f_i^{min} are the upper and lower bound of the i-th objective function in the secondary population.

Algorithm 2 describes the scatter search elements. The **getScatterSolution()** function returns the scatter solutions set in the nondominated set V, **getScatterSolution(n)** function returns the $n-th$ scatter solution and stores it in pl. **CreateRefSet(pl)** creates the reference set of the pl scatter solution. This function returns a set of solutions C_n Regarding the Solution Combination Method required by SS, we used the $BLX-\alpha$ recombination operator [6] with $\alpha = 0.5$. This operator combines the $i-th$ particle and $j-th$ particle from the C_n set. Finally, we used a Parameter-Based mutation as the Improvement Method with $p_m = \frac{1}{nVariables}$.

5 Results

In order to validate our proposed approach, we compare results with respect to the NSGA-II [4], which is a MOEA representative of the state-of-the-art in the area. The first phase of our approach uses four parameters: population size (P), leaders number (N), mutation probability (P_m), recombination parameter α, plus the traditional PSO parameters (w, c_1, c_2). On the other hand, the second phase uses two more parameters: reference set size ($RefSetSize$) and number of scatter solutions ($MaxScatterSolutions$). Finally, the $\epsilon - vector$ used to generate the ϵ-dominance grid was set to 0.05 in Kursawe's function, and to 0.02 in

Algorithm 2: Phase II - Scatter Search Algorithm

```
1  begin
2      repeat
3          getScatterSolutions()
4          for n = 0 to MaxScatterSolutions do
5              pl = getScatterSolution(n)
6              //Reference Set Update and Create Method
7              CreateRefSet(pl)
8              for i = 0 to SizeRefSet do
9                  for j = i + 1 to RefSetSize do
10                     //Solution Combination Method
11                     x = BLX − α(popRefSet(i), popRefSet(j))
12                     //Improvement Method
13                     x = Mutate(x)
14                     if x is nondominated then
15                         Add Particule x into secondary population
16                     end
17                 end
18             end
19         end
20     until MaxIter
21 end
```

the ZDT and the DTLZ test functions. Our approach was validated using 10 test problems: Kursawe's function [11], 5 problems from the **ZDT** set [16] and 4 from the **DTLZ** set [5]. The detailed description of these test functions was omitted due to space restrictions, but can be found in their original sources. However, all of these test functions are unconstrained and have between 3 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), $P_m = 1/n$, $w = 0.3$, $c_1 = 0.1$, $c_2 = 1.4$, $RefSetSize = 4$, $MaxScatterSolutions = 7$ and $\alpha = 0.5$. The NSGA-II used the following parameters: crossover rate = 0.9, mutation rate = $1/n$ (n = number of decision variables), $\eta_c = 15$, $\eta_m = 20$, 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 because with our approach we only need 4,000 fitness evaluations to converge to the real Pareto front in most of the test problems. Three performance measures were adopted in order to allow a quantitative assessment of our results: (1) Two Set Coverage (**SC**), proposed by Zitzler et al. [16], which performs a relative coverage comparison of two sets; (2) Inverted Generational Distance (**IGD**), which is a variation of a metric proposed by Van Veldhuizen [15] in which the true Pareto is used as a reference; and (3) Spread (**S**), proposed by Deb et al. [3], which measures both progress towards the Pareto-optimal front and the extent of spread. 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 (SC, IGD and S). We show in boldface the best mean values per test function. It can be observed that in the ZDT test problems, our approach produced the best results with respect to SC, IGD and S in all cases. Our approach also outperformed the NSGA-II with respect to the set coverage metric in the DTLZ1, DTLZ2 and DTLZ3 test problems. The NSGA-II outperformed our approach in three cases with respect to the IGD, and S metrics. Figures 2 and 3 shows the graphical results produced by the MOPSOSS and the 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 2 and 3, we can clearly see that in problems Kursawe, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, the NSGA-II is very far from the true Pareto front, whereas our MOPSOSS 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. Our results indicate that the NSGA-II, despite being a highly competitive MOEA is not able to converge to the true Pareto front in most of the test problems adopted when performing only 4000 fitness function evaluations. If we perform a higher number of evaluations, the NSGA-II would certainly produce a very good (and well-distributed) approximation of the Pareto front. However, our aim was precisely to provide an alternative approach that could require a lower number of evaluations than a state-of-the-art MOEA while still providing a highly competitive performance. Such an approach could be useful in real-world applications with objective functions requiring a very high evaluation cost (computationally speaking).

	SC				**IGD**				**S**			
Function	MOPSOSS		NSGA-II		MOPSOSS		NSGA-II		MOPSOSS		NSGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
KURSAWE	**0.1834**	0.0568	0.2130	0.0669	0.0056	0.0004	**0.0036**	0.0002	**0.4030**	0.0298	0.4325	0.0379
ZDT1	**0.0000**	0.0000	0.8622	0.0343	**0.0018**	0.0009	0.0097	0.0019	**0.4288**	0.0533	0.5515	0.0345
ZDT2	**0.0000**	0.0000	0.9515	0.0520	**0.0040**	0.0050	0.0223	0.0064	**0.5121**	0.0811	0.7135	0.1126
ZDT3	**0.0397**	0.0978	0.8811	0.0905	**0.0072**	0.0046	0.0155	0.0020	**0.6955**	0.0641	0.7446	0.0401
ZDT4	**0.0139**	0.0750	0.2331	0.1293	**0.1097**	0.0395	0.4247	0.1304	**0.9417**	0.0271	0.9813	0.0236
ZDT6	**0.0000**	0.0000	0.5417	0.1539	**0.0008**	0.0003	0.0420	0.0041	**0.7502**	0.0699	0.8713	0.0802
DTLZ1	**0.0403**	0.0598	0.6900	0.1942	**0.4100**	0.1131	0.7318	0.2062	0.9986	0.0010	**0.9976**	0.0011
DTLZ2	**0.0484**	0.0528	0.1856	0.0736	0.0005	0.0001	**0.0004**	0.0000	0.7488	0.1012	**0.2246**	0.0250
DTLZ3	**0.0207**	0.0540	0.4473	0.1893	**0.9331**	0.2631	1.4228	0.2690	**0.9991**	0.0004	**0.9991**	0.0002
DTLZ4	0.3262	0.3417	**0.0874**	0.1123	0.0216	0.0041	**0.0096**	0.0025	0.7605	0.1553	**0.7136**	0.1104

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

6 Conclusions and Future Work

We have introduced a new hybrid between a MOEA based on PSO and a local search mechanism based on scatter search. This hybrid aims to combine the high convergence rate of PSO with the good neighborhood exploration performed by the scatter search algorithm. In PSO, the main problem is the leader selection, the social parameter (p_best) is very important to get the high convergence rate required by our approach. With SS we observe that the selection of closer solutions to the Pareto front generates smooth moves that give us more solutions closer to the true Pareto front of the problem being solved. Our proposed approach produced results that are competitive with respect to the NSGA-II in problems whose dimensionality goes from 3 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 gratefully acknowledges support from CONACyT through project 45683-Y.

References

1. Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.
2. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
3. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001. ISBN 0-471-87339-X.
4. 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.
5. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.
6. Larry J. Eshelman and J. Davis Schaffer. Real-coded Genetic Algorithms and Interval-Schemata. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann Publishers, California, 1993.
7. Fred Glover. Heuristic for integer programming using surrogate constraints. *Decision Sciences 8*, pages 156–166, 1977.
8. Fred Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49(1-3):231–255, 1994.

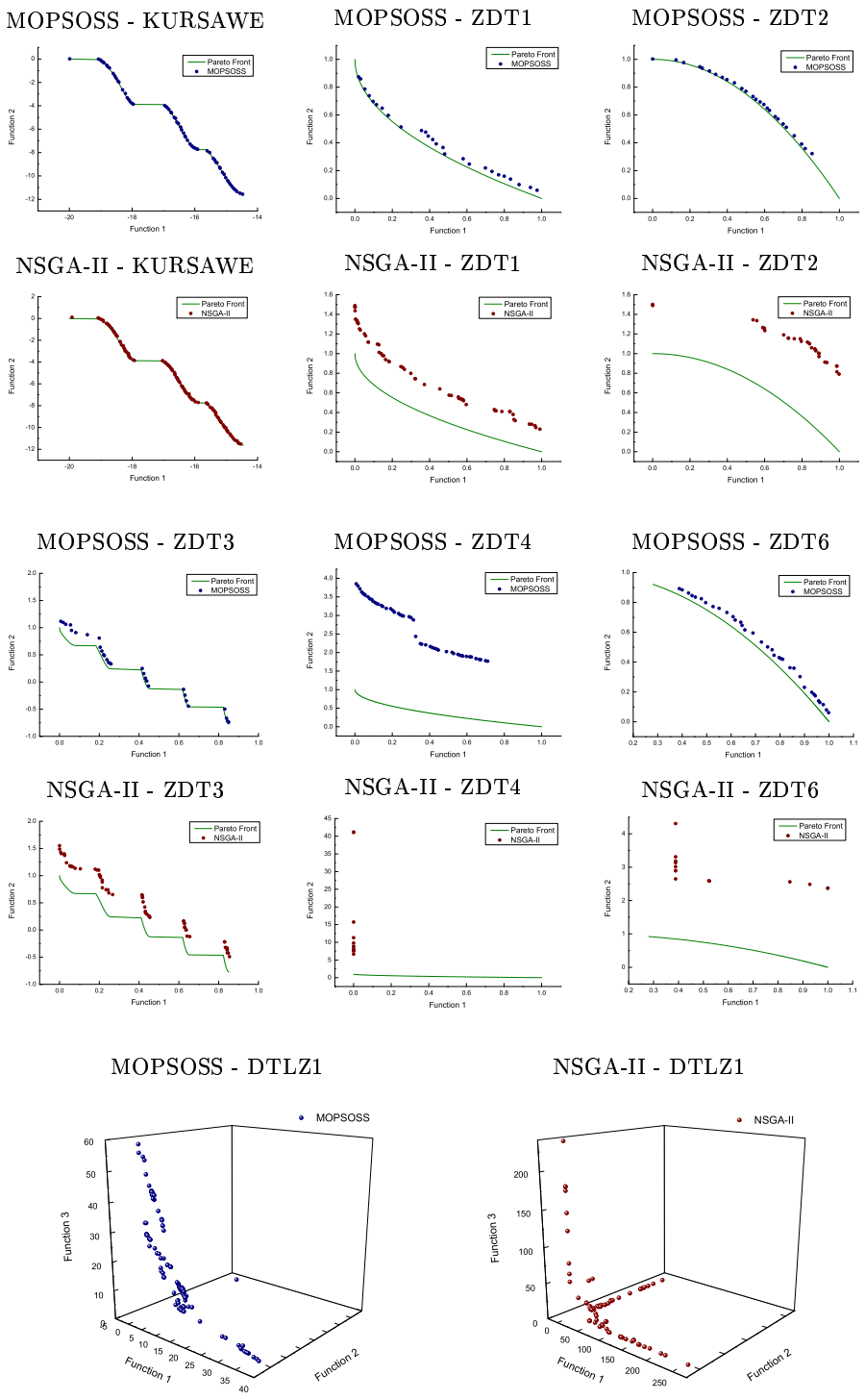

Fig. 2. Pareto fronts generated by MOPSOSS and NSGA-II for Kursawe's, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 and DTLZ1 test functions.

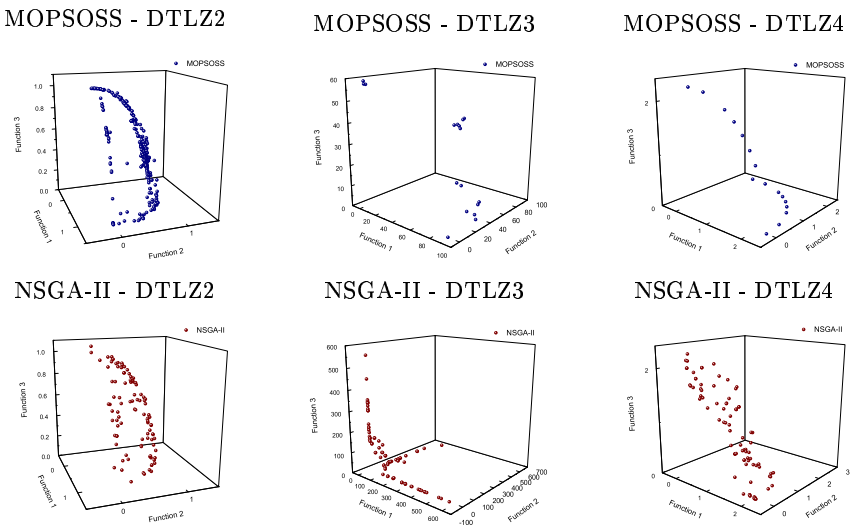

Fig. 3. Pareto fronts generated by MOPSOSS and NSGA-II for the DTLZ2, DTLZ3 and DTLZ4 test functions.

9. Fred Glover. A template for scatter search and path relinking. In *AE '97: Selected Papers from the Third European Conference on Artificial Evolution*, pages 13–54, London, UK, 1998. Springer-Verlag.
10. James Kennedy and Russell C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann Publishers, California, USA, 2001.
11. Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science Vol. 496*, pages 193–197, Berlin, Germany, October 1991. Springer-Verlag.
12. Manuel Laguna and Rafael Martí. *Scatter Search: Methodology and Implementations in C.* Kluwer Academic Publishers, 2003.
13. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.
14. Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *2003 IEEE SIS Proceedings*, pages 26–33, Indianapolis, USA, April 2003. IEEE Service Center.
15. David A. Van Veldhuizen. *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, May 1999.
16. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.