Towards a More Efficient Multi-Objective Particle Swarm Optimizer

Luis V. Santana-Quintero, Noel Ramírez-Santiago
and Carlos A. Coello Coello
CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508
Col. San Pedro Zacatenco
México, D.F. 07360
MEXICO
Phone: +52 55 5061 3800 x 6564
Fax: +52 55 5061 3757
email: lvspenny@hotmail.com,
santiago@computacion.cs.cinvestav.mx,
ccoello@cs.cinvestav.mx

Introduction

Despite the considerable volume of research developed on evolutionary multi-objective optimization during the last 20 years, certain topics such as algorithmic design emphasizing efficiency have become popular only within the last few years (Coello Coello, Van Veldhuizen, & Lamont, 2002).

In this chapter, we propose a new hybrid multi-objective evolutionary algorithm (MOEA) based on particle swarm optimization (PSO) and scatter search (SS). The main motivation of this work was to design a MOEA that could produce a reasonably good approximation of the true Pareto front of a problem with a relatively low number of fitness function evaluations. The core idea of our proposal is to combine the high convergence rate of PSO with the use of SS as a local search mechanism that compensates for the diversity lost during the search (due to the high selection pressure generated by the leader selection scheme that we propose for our PSO). As we show later, our proposed hybrid scheme constitutes an efficient MOEA, which can produce reasonably good approximations of the Pareto fronts of both constrained and unconstrained multi-objective problems of high dimensionality, performing a relatively low number of fitness function evaluations. Our results are compared with respect to a MOEA that is representative of the state-of-the-art in the area: the NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002).

The third author acknowledges support from CONACyT project no. 45683-Y.

The remainder of this chapter is organized as follows. In Section , we provide some basic concepts from multi-objective optimization required to make the chapter self-contained. This includes an introduction to the PSO strategy (subsection) and to SS (subsection). We also present a brief review of several multi-objective particle swarm optimizers previously proposed in the specialized literature (subsection). Section provides the details of our proposed approach and the mechanism adopted to maintain diversity. Our comparison of results is provided in Section . Our conclusions are presented in Section , and some promising paths for future research are briefly described in Section . Finally, for those interested in gaining more in-depth knowledge about the topics covered in this chapter, a list of additional readings is provided in Section .

Basic Concepts

In this section, we introduce some basic definitions which aim to make the paper self-contained. We introduce some basic terminology from the multi-objective optimization literature (assuming minimization) and we provide an introduction to both particle swarm optimization and scatter search.

Multi-objective optimization

The multi-objective optimization problem can be formally defined as the problem of finding:
$\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which satisfies the m inequality constraints:

$$g_i(\overrightarrow{x}) \leq 0; i = 1, \ldots, m$$

the p equality constraints:

$$h_i(\overrightarrow{x}) = 0; i = 1, \ldots, p$$

and optimizes the vector function:

$$\overrightarrow{f}(\vec{x}) = f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x})$$

In other words, we aim to determine from among the set $\mathcal{F}$ of all vectors (points) which satisfy the constraints those that yield the optimum values for all the k-objective functions simultaneously. The constraints define the feasible region $\mathcal{F}$ and any point $\overrightarrow{x}$ in the feasible region is called a feasible point.

Pareto Dominance.
Pareto dominance is formally defined as follows:
A vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\overrightarrow{v} = (v_1, \ldots, v_k)$ *if and only if* $\overrightarrow{u}$ *is partially less than* $\overrightarrow{v}$*, i.e.,* $\forall i \in (1, \ldots, k), u_i \leq v_i \wedge \exists i \in (1, \ldots, k) : u_i < v_i$.

In words, this definition says that a solution dominates another one, only if it's strictly better in at least one objective, and not worse in any of them. So, when we are comparing two different solutions A and B, there are 3 possibilities:

- A dominates B
- A is dominated by B
- A and B are nondominated

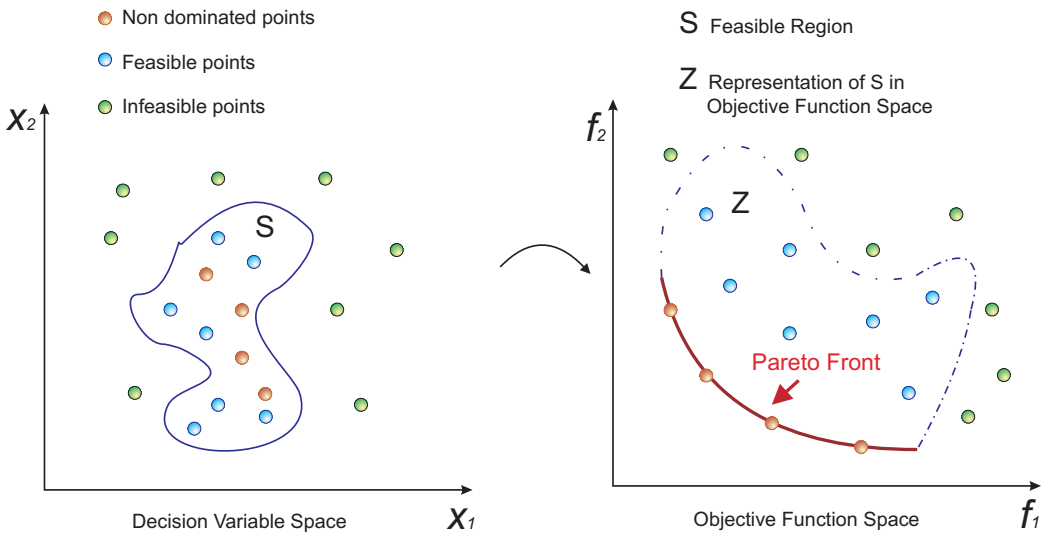

Figure 1. Multi-objective Optimization Problem

Pareto Optimality. The formal definition of *Pareto optimality* is provided next:

A solution $\overrightarrow{x_u} \in \mathcal{F}$ *(where* $\mathcal{F}$ *is the feasible region) is said to be Pareto optimal if and only if there is no* $\overrightarrow{x_v} \in \mathcal{F}$ *for which* $v = f(x_v) = (v_1, \ldots, v_k)$ *dominates* $u = f(x_u) = (u_1, \ldots, u_k)$, where k is the number of objectives.

In words, this definition says that $\overrightarrow{x}_u$ is Pareto optimal if there exists no feasible vector $\overrightarrow{x_v}$ which would decrease some objective without causing a simultaneous increase in at least one other objective.

This does not provide us a single solution (in decision variable space), but a set of solutions which form the so-called *Pareto Optimal Set*. The vectors that correspond to the solutions included in the Pareto optimal set are *nondominated*.

Pareto Front.

When all nondominated solutions are plotted in objective space, the nondominated vectors are collectively known as the *Pareto Front*. Formally:

For a given MOP $\overrightarrow{f}(x)$ and Pareto optimal set P^*, the Pareto front (PF^*) is defined as:

$$PF^* := \{\overrightarrow{f} = [f_1(x), \ldots, f_k(x)] | x \in P^*\}$$

The previous definitions are graphically depicted in Figure 1, showing the *Pareto front*, the *Pareto Optimal Set* and *dominance* relations among solutions.

Particle Swarm Optimization (PSO)

PSO is a bio-inspired optimization algorithm that was proposed by James Kennedy and Russell Eberhart in the mid-1990s (Kennedy & Eberhart, 2001), 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 (Kennedy & Eberhart, 2001; Reyes-Sierra & Coello Coello, 2006).

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, each particle can be affected by either the best local and/or the best global particle in its swarm. In some of our previous work, we noticed that PSO normally has difficulties to achieve a good distribution of solutions with a low number of fitness function evaluations (Coello Coello, Toscano Pulido, & Salazar Lechuga, 2004). That is why we adopted scatter search (which can be useful at finding solutions within the neighborhood of a reference set) in this work in order to have a local optimizer whose computational cost is low.

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 criterion is reached.

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

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

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; $j = 1, 2, \ldots, k$ and k represents the dimension; 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. 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.

Multi-objective Algorithms based on PSO.

There are plenty of proposals to extend PSO for dealing with multiple objectives (see for example (Alvarez-Benitez, Everson, & Fieldsend, 2005; Bartz-Beielstein et al., 2003; Baumgartner, Magele, & Renhart, 2004; Coello Coello & Salazar Lechuga, 2002; Mahfouf, Chen, & Linkens, 2004)) and the survey by Reyes-Sierra and Coello Coello (Reyes-Sierra & Coello Coello, 2006). A brief description of the most representative proposals is provided next and a summary of their features is presented in Table 1:

Alvarez-Benitez et al. (Alvarez-Benitez et al., 2005): The authors propose methods based exclusively on Pareto dominance for selecting leaders from an unconstrained nondominated (external) archive. The authors propose and evaluate four mechanisms for confining particles to the feasible region, that is, constraint-handling methods. The

authors show that a probabilistic selection favoring archival particles that dominate few particles provides good convergence towards the Pareto front while properly covering it at the same time. Also, they conclude that allowing particles to explore regions close to the constraint boundaries is important to ensure convergence to the Pareto front. This approach uses a turbulence factor that is added to the position of the particles with certain probability.

Bartz et al. (Bartz-Beielstein et al., 2003): This approach starts from the idea of introducing elitism (through the use of an external archive) into PSO. Different methods for selecting and deleting particles (leaders) from the archive are analyzed to generate a satisfactory approximation of the Pareto front. Selecting methods are either inversely related to the fitness value or based on the previous success of each particle. The authors provide some statistical analysis in order to assess the impact of each of the parameters used by their approach.

Baumgartner et al. (Baumgartner et al., 2004): This approach, based on the *fully connected* topology, uses linear aggregating functions. In this case, the swarm is equally partitioned into n subswarms, each of which uses a different set of weights and evolves into the direction of its own swarm leader. The approach adopts a gradient technique to identify the Pareto optimal solutions.

Coello Coello et al. (Coello Coello & Salazar Lechuga, 2002; Coello Coello et al., 2004): This proposal is based on the idea of having an external archive in which every particle deposits its flight experiences after each flight cycle. The search space explored is divided in hypercubes. Each hypercube receives a fitness value based on the number of particles it contains. Once a hypercube has been selected, the leader is randomly chosen. This approach also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved.

Fieldsend and Singh (Fieldsend & Singh, 2002): This approach uses an unconstrained elite external archive (in which a special data structure called "dominated tree" is adopted) to store the nondominated individuals. The archive interacts with the primary population in order to define leaders. The selection of the gbest for a particle in the swarm is based on the structure defined by the dominated tree. First, a composite point of the tree is located based on dominance relations, and then the closest member (in objective function space) of the composite point is chosen as the leader. On the other hand, a set of personal best particles found (nondominated) is also maintained for each swarm member, and the selection is performed uniformly. This approach also uses a "turbulence" operator that is basically a mutation operator that acts on the velocity value used by the PSO algorithm.

Mahfouf et al. (Mahfouf et al., 2004): The authors propose an Adaptive Weighted PSO (AWPSO) algorithm, in which the velocity is modified by including an acceleration term that increases as the number of iterations increases. This aims to enhance the global search ability at the end of the run and to help the algorithm to jump out of local optima. The authors use dynamic weights to generate Pareto optimal solutions. When the population is losing diversity, a mutation operator is applied to the positions

of certain particles and the best of them are retained. Finally, the authors include a nondominated sorting algorithm to select the particles from one iteration to the next.

Moore and Chapman (Moore & Chapman, 1999): This was the first attempt to produce a multi-objective particle swarm optimizer. In this approach, the personal best (*pbest*) of a particle is a list of all the nondominated solutions it has found in its trajectory. When selecting a *pbest*, a particle from the list is randomly chosen. Since the ring topology is used, when selecting the best particle of the neighborhood, the solutions contained in the *pbest* lists are compared, and a nondominated solution with respect to the neighborhood is chosen. The authors do not indicate how they choose the *lbest* particle when more than one nondominated solution is found in the neighborhood.

Parsopoulos et al. (Parsopoulos, Tasoulis, & Vrahatis, 2004): studied a parallel version of the Vector Evaluated Particle Swarm (VEPSO) method for multi-objective problems. VEPSO is a multi-swarm variant of PSO, which is inspired on the Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985, 1985). In VEPSO, each swarm is evaluated using only one of the objective functions of the problem under consideration, and the information it possesses for this objective function is communicated to the other swarms through the exchange of their best experience (*gbest* particle). The authors argue that this process can lead to Pareto optimal solutions.

Reyes-Sierra and Coello Coello (Reyes Sierra & Coello Coello, 2005): This approach is based on Pareto dominance and the use of a nearest neighbor density estimator for the selection of leaders (by means of a binary tournament). This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions. On the other hand, the concept of ϵ-dominance is used to select the particles that remain in the archive of final solutions. Additionally, the authors propose a scheme in which they subdivide the population (or swarm) into three different subsets. A different mutation operator is applied to each subset. Finally, this approach incorporates fitness inheritance (Smith, Dike, & Stegmann, 1995) in order to reduce the total number of fitness function evaluations performed. However, this approach performs 20,000 fitness function evaluations, which is considerably higher than the number of evaluations performed by the algorithm proposed in this chapter.

Scatter Search

Scatter search (SS) is an evolutionary method that was originally proposed in the 1970s by Fred Glover (Glover, 1977) 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") (Laguna & Martí, 2003). 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.

In 1994 (Glover, 1994), the range of applications of SS was expanded to nonlinear optimization problems, binary and permutation problems. Finally, in 1998 a new publication on scatter search (Glover, 1998) triggered the interest of researchers and practitioners,

Author(s) / (reference)	Selection
Alvarez-Benitez et al. (Alvarez-Benitez et al., 2005)	Pareto Ranking
Bartz et al. (Bartz-Beielstein et al., 2003)	Population-based
Baumgartner et al. (Baumgartner et al., 2004)	Linear Aggregation
Coello et al. (Coello Coello & Salazar Lechuga, 2002; Coello Coello et al., 2004)	Pareto Ranking
Fieldsend and Singh (Fieldsend & Singh, 2002)	Dominated Tree based
Mahfouf et al. (Mahfouf et al., 2004)	Population-based
Moore and Chapman (Moore & Chapman, 1999)	Population-based
Parsopoulos et al. (Parsopoulos et al., 2004)	Population-based
Reyes and Coello (Reyes Sierra & Coello Coello, 2005)	Pareto Ranking

Table 1: Characteristics of different multi-objective particle swarm optimizers

who translated these ideas into different computer implementations to solve a variety of problems.

Scatter search operates on a set of solutions (the reference set) by combining these solutions to create new ones. When the main mechanism for combining solutions is such that a new solution is created from the linear combination of two other solutions, the reference starts to evolve and the new solutions generated become part of the reference set.

Unlike a population in genetic algorithms, the reference set of solutions in scatter search is relatively small. In genetic algorithms, two solutions are randomly chosen from the population and a crossover or combination mechanism is applied to generate one or more offspring. A typical population size in a genetic algorithm consists of 100 elements, which are randomly sampled to create combinations. In contrast, scatter search chooses two or more elements of the reference set in a systematic way with the purpose of creating new solutions. Typically, the reference set in scatter search has 20 solutions or less. The scatter search algorithm contains five different methods, that are illustrated in Figure 2 and are briefly explained next:

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. $RefSet$ contains the best solutions in terms of quality and diversity found so far. The *Subset Generation Method* takes the 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 handled by the *Reference Set Update Method*. This method applies rules regarding the admission of solutions to the reference set $RefSet$.

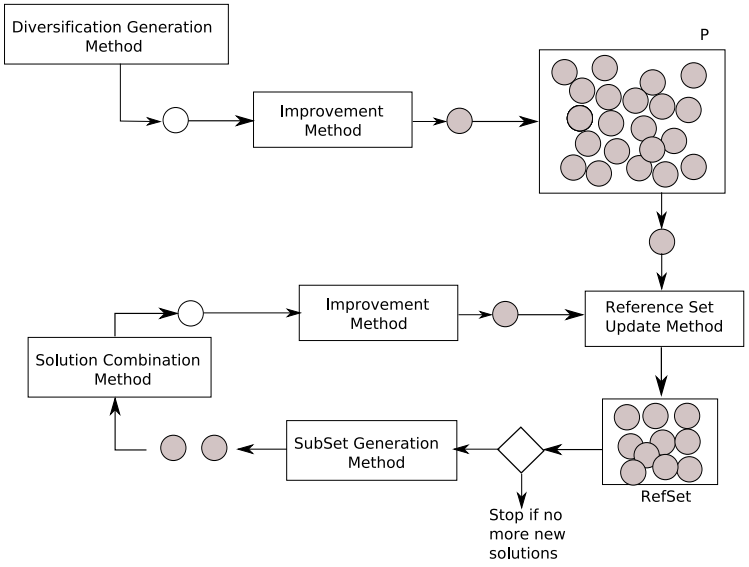

Figure 2. Scatter Search Scheme

Handling well-distributed solutions

Researchers have proposed several mechanisms to reduce the number of nondominated solutions generated by a MOEA (most of them applicable to external archives): clusters (Zitzler & Thiele, 1999), adaptive grids (Knowles & Corne, 2000, 2000), crowding (Deb et al., 2002) and relaxed forms of Pareto dominance (Laumanns, Thiele, Deb, & Zitzler, 2002). However, we only mention those that need an external archive to store nondominated solutions, such as:

• **Adaptive Grid**: Proposed by Knowles and Corne (Knowles & Corne, 2000, 2000), the adaptive grid is really a space formed by hypercubes. Such hypercubes have as many components as objective functions has the problem to be solved. Each hypercube can be interpreted as a geographical region that contains an n number of individuals. The adaptive grid allows us to store nondominated solutions and to redistribute them when its maximum capacity is reached.

• **ϵ-dominance**: This is a relaxed form of dominance proposed by Laumanns et al. (Laumanns et al., 2002). The so-called ϵ-Pareto set is an archiving strategy that maintains a subset of generated solutions. It guarantees convergence and diversity according to well-defined criteria, namely the value of the ϵ parameter, which defines the resolution of the grid to be adopted for the secondary population. The general idea of this mechanism is to divide objective function space into boxes of size ϵ. In Figure 3 it is shown an ϵ-dominance example in which it can be seen the main difference between Pareto dominance and ϵ-dominance. Each box can be interpreted as a geographical region that contains a single solution. This algorithm is very attractive both from a theoretical and from a practical point of view. However, in order to achieve the best performance, it is necessary to provide the size of the box (the ϵ parameter) which is problem-dependent, and it's normally not known before

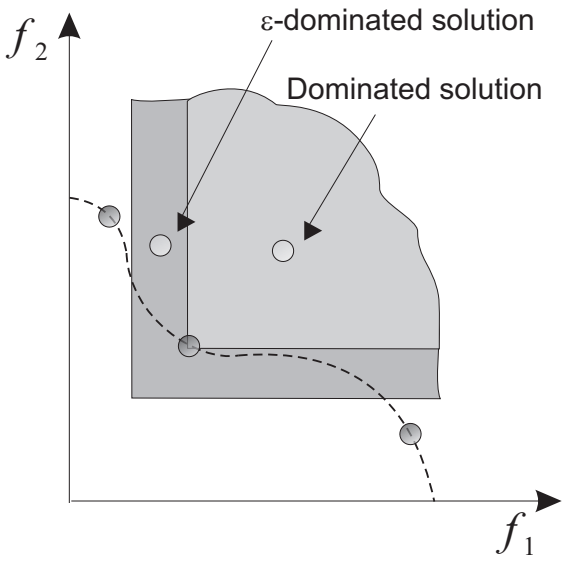

Figure 3. ϵ-dominance relation example

executing a MOEA.

Our Proposed Approach

Our proposed approach, called **Constrained Multi-objective Optimization using Particle Swarm Optimization with Scatter Search (C-MOPSOSS)**, 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 for unconstrained problems and 2500 evaluations for constrained problems. During Phase II, a local search procedure based on scatter search is applied (for the same number of fitness function evaluations as in Phase I), 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.

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 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 is formed with $(f_1^*, f_2^*, \ldots, f_N^*)$ where f_i^* is the best solution found so far for the ith objective function. 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 initial population, but in the population we need at least the same number of individuals as the number of objectives plus one. This last individual is needed to form the ideal vector; for this purpose, we chose 5 individuals to perform the experiments reported in this work. 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 $getLeader(\vec{x})$ 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 $BLX - \alpha$[1] recombination operator (Eshelman & Schaffer, 1993, 1993), and a new vector solution $Z = (z_1, z_2, ..., z_d)$ is generated, where $z_i \in [c_{min} - I \cdot \alpha, c_{max} + I \cdot \alpha]$; $c_{max} = max(a_i, b_i)$, $c_{min} = min(a_i, b_i)$, $I = c_{max} - c_{min}$, $\alpha = 0.5$, $a = L_g$ (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 either (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 (Reyes-Sierra & Coello Coello, 2006; Mostaghim & Teich, 2003). 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 (Deb et al., 2002) in our approach with $p_m = \frac{1}{nVariables}$. Our proposed approach uses an external archive (also called secondary population) and a selection procedure which is described in detail in Section .

The second phase of our algorithm is based on Scatter Search, and it requires a set of dominated points to work properly. For this purpose, we decided to include a third population that stores the dominated points that are being removed constantly from the secondary population as the search progresses. Thus, this third population contains solutions that are (globally) dominated, but that are close from being nondominated (as shown in Figure 4).

Phase II: Scatter Search

Phase II 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):

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

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

[1] $BLX - \alpha$ crossover is commonly used for real-numbers encoding. This operator generates an offspring between two real numbers a and b, randomly selected from an interval of width $2 \cdot |a - b|$ centered on the mean of the a and b values.

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} + c_1 · U(0,1)(p_{i,d} − x_{i,d}) + c_2 · 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 and x_i is feasible 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
```

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

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.

This selection process is shown in Figure 5, where the black points are the scatter particles that we use to obtain the reference set. Algorithm 2 describes the scatter search process used in the second phase of our approach. This process is graphically depicted in Figure 6. The **getScatterSolution()** function returns the scatter solutions set in the nondominated set V. The **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 as it is shown in Figure 5 (those which are enclosed by the continuous black line). Regarding the Solution Combination Method required by SS, we used the $BLX - \alpha$ recombination operator (Eshelman & Schaffer, 1993,

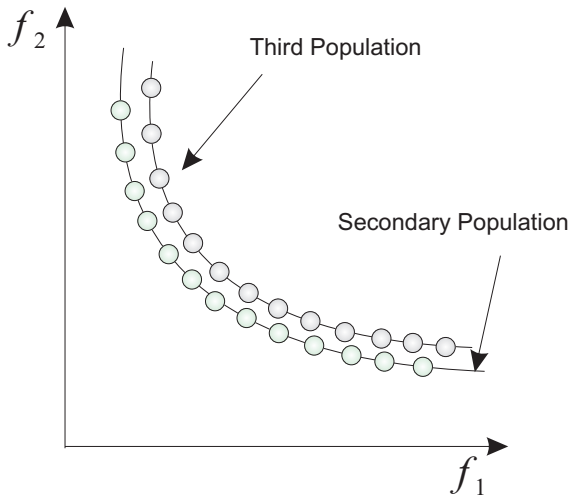

Figure 4. Secondary and Third Population

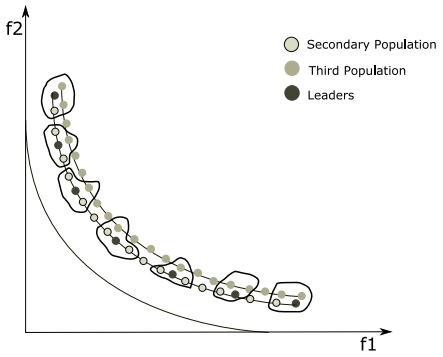

Figure 5. Solutions selected to create the reference set used in scatter search.

1993) with $\alpha = 0.5$. This operator combines the $i-th$ particle with the $j-th$ particle from the C_n set. Finally, we used a Parameter-Based mutation as the Improvement Method with $p_m = \frac{1}{nVariables}$.

Use of ϵ-dominance

As indicated before, our proposed approach uses an external archive (also called secondary population) together with the ϵ-dominance concept proposed by Laumanns (Laumanns et al., 2002). In order to include a solution into this archive, it is compared with respect to each member already contained in the archive using ϵ-dominance. The procedure is described next.

Every solution in the archive is assigned an identification array ($\mathbf{B}$ =

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 and x is feasible then
15                         Add Particle x into secondary population
16                     end
17                 end
18             end
19         end
20     until MaxIter
21 end
```

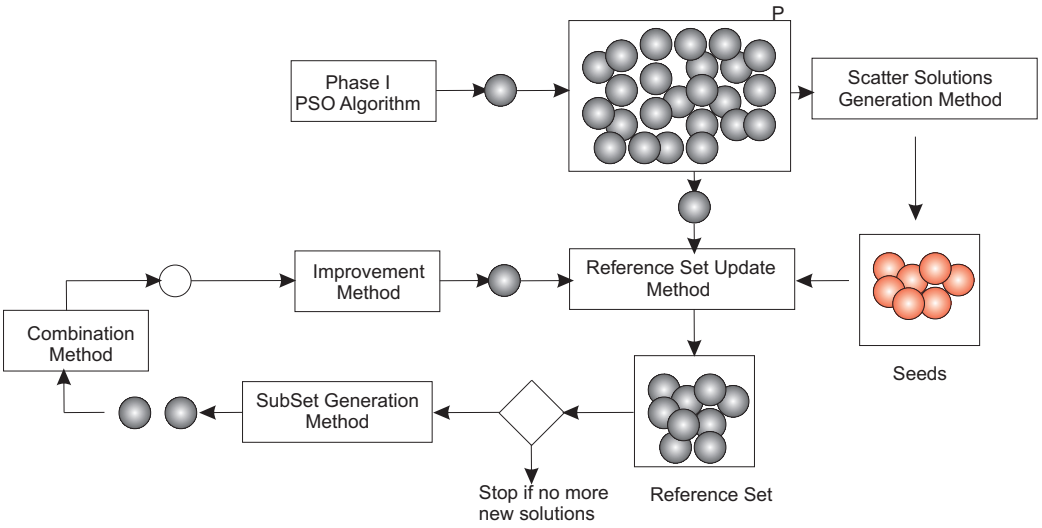

Figure 6. Multi-objective Optimization Problem

$(B_1, B_2, \ldots, B_k)^T$, where k 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, $\lceil\ \rceil$, $\lfloor\ \rfloor$ are the ceiling and floor functions, respectively, and ϵ_j is the allowable tolerance in the j-th objective (Laumanns et al., 2002). The identification array divides the whole objective space into hyper-boxes, each having ϵ_j size in the j-th objective. With the identification arrays calculated for the offspring c_1 and each archive member a, we use the procedure illustrated in Figure 7 and described next:

1. If the identification array B_a of any archive member a dominates that of the offspring c_i, then it means that the offspring is ϵ-dominated by this archive member and so the offspring *is not accepted*. This is case (a) in Figure 7.

2. If B_{c_i} of the offspring dominates the B_a of any archive member a, the archive member is deleted and the offspring *is accepted*. This is case (b) in Figure 7.

If neither of the two above cases occur, then it means that the offspring is ϵ-nondominated with respect to the archive contents. There are two further possibilities in this case:
1. If the offspring shares the same **B** vector with an archive member (meaning that they belong to the same hyper-box), then they are first checked for the usual nondomination. If the offspring dominates the archive member or the offspring is nondominated with respect to the archive member but is closer to the **B** vector (in terms of the Euclidian distance) than the archive member, then the offspring *is retained*. This is case (c) in Figure 7.
2. In the event of an offspring not sharing the same **B** vector with any archive member, the offspring *is accepted*. This is case (d) in Figure 7.

Using the above 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. Thus, there is no need to pre-fix an upper limit on the size of the archive as done in most traditional multi-objective evolutionary algorithms.

Comparison of Results

We chose thirteen test problems with different geometrical characteristics for our experimental study (ten of them are unconstrained and three with constraints). The problems selected are the following: **Kursawe**'s problem (Kursawe, 1991, 1991) (the Pareto front is disconnected); 5 problems from the **ZDT** set (multimodal problems) (Zitzler, Deb, & Thiele, 2000); 4 from the **DTLZ** set (Deb, Thiele, Laumanns, & Zitzler, 2005) (three-objective problems); and 4 more functions have constraints: **Osyczka** (Osyczka and Kundu, 1995, 1995), **Kita** (Kita et al., 1996, 1996), **Welded Beam** (Ragsdell & Phillips, 1975) and **Speed Reducer** (J., 1970). The full description of **Welded Beam** and **Speed Reducer** is presented in Table 6. The description of the other test functions is available from their original sources, and was omitted due to space limitations.

In order to allow a quantitative comparison of results, we adopted two performance measures that are used to determine if our approach has converged to the true Pareto front (Two Set Coverage and Inverted Generational Distance) and a third one was adopted to

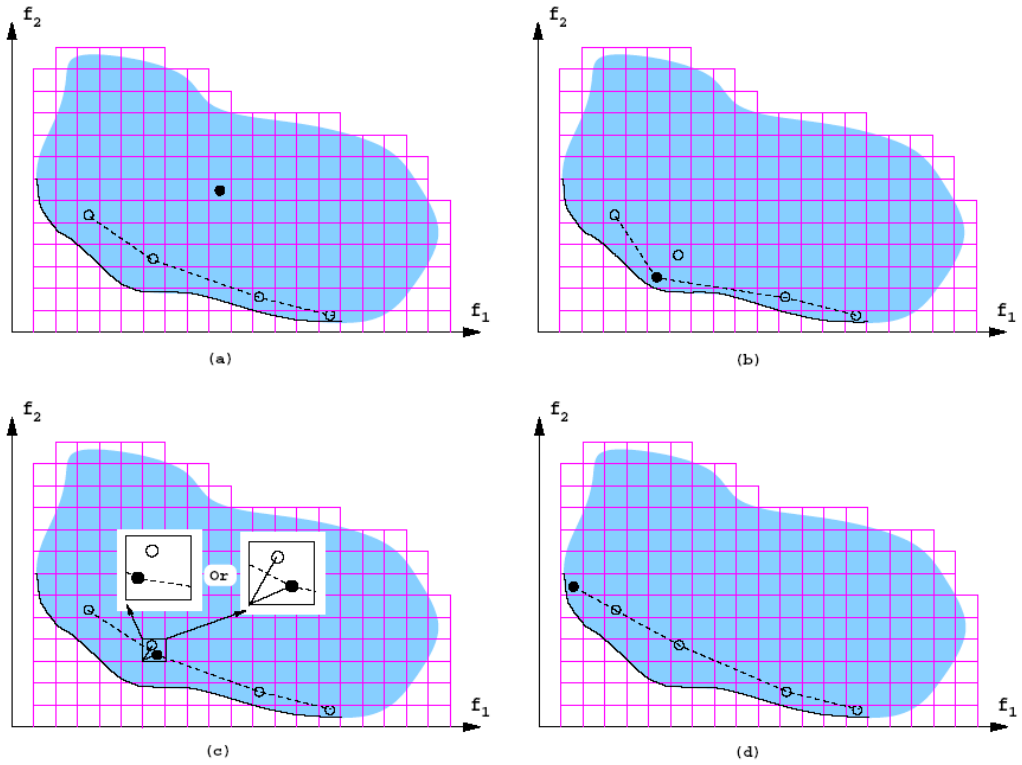

Figure 7. Four cases of inserting a child into the external archive

determine if the solutions are uniformly distributed along the Pareto front (Spread). A brief description of each of them is provided next:

Two Set coverage (SC) Zitzler (Zitzler et al., 2000) proposed this metric that can be termed relative coverage comparison of two sets. SC is defined as the mapping of the order pair (X', X'') to the interval [0,1] as follows:

$$SC(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a \succeq a''\}|}{|X''|}$$

Consider $X', X'' \subseteq X'$as two sets of decision vectors. If all points in X' dominate or are equal to all points in X", then by definition SC = 1. SC = 0 implies the opposite. Observe that this is not a distance measure that allows to determine how close these sets are from each other.

Inverted Generational Distance (IGD): Van Veldhuizen and Lamont (Van Veldhuizen and Lamont, 1998, 1998; Veldhuizen & Lamont, 2000b, 2000) proposed the Generational Distance (GD) metric as a way of estimating how far are the elements in the Pareto front produced by an algorithm from those in the true Pareto front of the problem. Mathematically:

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

where n is the number of nondominated solutions found by the algorithm being analyzed and d_i is the Euclidean distance between each of these and the nearest member of the true Pareto front. A value of zero for this metric indicates that our algorithm has converged to the true Pareto front. Therefore, any other value indicates how "far" we are from the true Pareto front. We adopted a variation of this metric, called Inverted Generational Distance (IGD) in which we use as a reference the true Pareto front, and we compare each of its elements with respect to the approximation produced by an algorithm. This provides a better estimate and avoids some of the pathological cases of poor convergence that could arise (see (Coello Coello & Cruz Cortés, 2005) for a more extended discussion on this metric).

Spread metric (S) Deb (Deb, 2001) proposed the metric Δ with the idea of measuring both progress towards the Pareto-optimal front and the extent of spread. To this end, if P is a subset of the Pareto-optimal front, Δ is defined as follows

$$\Delta = \frac{\sum_{i=1}^{m} d_i^e + \sum_{i=1}^{|F|} |d_i - \overline{d}|}{\sum_{i=1}^{m} d_i^e + |F|\overline{d}}.$$

where d_i^e denotes the distance between the i-th coordinate for both extreme points in P and F, and d_i measures the distance of each point in F to its closer point in F.

From the above definition, it is easy to conclude that $0 \leq \Delta \leq 1$ and the lower the Δ value, the better the distribution of solutions. A perfect distribution, that is $\Delta = 0$, means that the extreme points of the Pareto-optimal front have been found and d_i is constant for all i.

Discussion of Results

This section is divided in four parts: 1) in the first one, we compare our approach with respect to the NSGA-II, which is a MOEA representative of the state-of-the-art in the area. 2) In the second part we compare our proposed approach against the MOPSO proposed by Coello Coello et al. (Coello Coello et al., 2004). 3) In the third part we compare our approach without scatter search (i.e., using only the first phase) with respect to the full approach (i.e., using the two phases) in order to assess the usefulness of the scatter search algorithm. Finally, 4) the fourth part is dedicated to show that the C-MOPSOSS is capable of converging to the true Pareto front of all the test functions adopted in our study, if allowed a sufficiently large number of fitness function evaluations.

It is important to mention the parameters that we use for all the different test functions in order to make all the comparisons in this section. 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 search solutions ($MaxScatterSolutions$). Finally, the $\epsilon - vector$ used to generate the ϵ-dominance grid was set to 0.05 in Kursawe's function, 0.02 in the ZDT and the DTLZ test functions, and to 0.1 for the constrained functions. 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 $\alpha = 0.5$. These values were empirically derived after numerous experiments.

C-MOPSOSS vs NSGA-II comparison.

In order to validate our proposed approach, we compare results with respect to the NSGA-II (Deb et al., 2002), which is a MOEA representative of the state-of-the-art in the area.

The NSGA-II was used with the following parameters[2]: crossover rate = 0.9, mutation rate = $1/n$ (n = number of decision variables), $\eta_c = 15$, $\eta_m = 20$[3] (as suggested in (Deb et al., 2002)), population size = 100 and maximum number of generations = 40 or 50 for constrained problems. 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 the same number of fitness function evaluations per run because with our approach we only need 4,000 (and 5,000 for constrained problems) fitness function evaluations to produce a reasonably good approximation of the true Pareto front in most of the test problems adopted in our study. For each test problem, 30 independent runs were performed. Our results reported in Table 2 correspond to the mean and standard deviation of the performance metrics (SC, IGD and S). We show in boldface the best mean values for each test function.

In Table 2, it can be observed that in the ZDT test problems, our approach produced the best results with respect to the SC, IGD and S metrics, in all cases. Our approach also outperformed the NSGA-II with respect to the SC metric in the DTLZ1, DTLZ2 and DTLZ3 test problems. The NSGA-II found better results in three cases with respect to the IGD, and S metrics. In constrained problems, our C-MOPSOSS obtained better results with respect to the SC and Spread metrics in all three cases, and the NSGA-II outperformed it with respect to the IGD metric for Osyczka's and the Welded beam problems. Based on these results, we can conclude that our approach obtained better solutions than the NSGA-II with the same number of function evaluations in almost all the problems except for DTLZ4. With respect to the IGD metric, we obtained better results than the NSGA-II in 10 of the 14 test functions adopted. We also concluded that the use of ϵ-dominance helped our approach to maintain a well-distributed set of nondominated solutions because we were able to outperform the NSGA-II with respect to the S metric in 12 of the 14 test functions, in spite of the fact that the NSGA-II uses a very powerful density estimator (the crowded-comparison operator).

Figures 8 and 9 show the graphical results produced by our C-MOPSOSS and the NSGA-II for all the test problems adopted. The solutions displayed correspond to the me-

[2]The NSGA-II used in this work was obtained from `http://www.iitk.ac.in/kangal/codes.shtml` using the parameters suggested by its authors (Deb et al., 2002).

[3]The NSGA-II adopts Simulated Binary Crossover and Parameter-Based mutation as defined in (Deb & Agrawal, 1995).

	SC				IGD				Spread			
Function	**C**-MOPSOSS		**N**SGA-II		**C**-MOPSOSS		**N**SGA-II		**C**-MOPSOSS		**N**SGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
KURSAWE	**0.1738**	0.0635	0.2166	0.0664	**0.0005**	0.0004	0.0036	0.0002	**0.3978**	0.0257	0.4383	0.0370
ZDT1	**0.0000**	0.0000	0.8662	0.0418	**0.0018**	0.0011	0.0097	0.0019	**0.3190**	0.0634	0.5444	0.0380
ZDT2	**0.0000**	0.0000	0.8894	0.1718	**0.0052**	0.0064	0.0223	0.0064	**0.3984**	0.1108	0.7213	0.1102
ZDT3	**0.0117**	0.0391	0.9011	0.0483	**0.0047**	0.0033	0.0155	0.0020	**0.6462**	0.0635	0.7492	0.0353
ZDT4	**0.0000**	0.0000	0.2415	0.1316	**0.1065**	0.0421	0.4297	0.1304	**0.6992**	0.0771	0.9840	0.0193
ZDT6	**0.0000**	0.0000	0.5375	0.1426	**0.0011**	0.0006	0.0420	0.0041	**0.4261**	0.1557	0.8747	0.0778
DTLZ1	**0.0551**	0.0993	0.6826	0.1502	**0.3942**	0.1181	0.7318	0.2062	**0.9939**	0.0040	0.9981	0.0008
DTLZ2	**0.0656**	0.0800	0.1806	0.0800	**0.0006**	0.0001	0.0004	0.0000	0.7600	0.0686	**0.2155**	0.0202
DTLZ3	**0.0036**	0.0140	0.5165	0.2214	**0.8367**	0.2348	1.4228	0.2690	**0.9963**	0.0029	0.9992	0.0002
DTLZ4	0.4329	0.3815	**0.0815**	0.1760	0.0222	0.0043	**0.0096**	0.0025	0.7345	0.1168	**0.7251**	0.0898
OSYCZKA	**0.1613**	0.2248	0.6233	0.2363	0.0020	0.0011	**0.0012**	0.0006	**0.9667**	0.0173	0.9848	0.0251
KITA	**0.2663**	0.0696	0.4033	0.0675	**0.0009**	0.0001	**0.0009**	0.0001	**0.5620**	0.0964	0.6400	0.1861
WELBEAM	**0.3694**	0.3399	0.3812	0.2049	0.0047	0.0059	**0.0024**	0.0019	**0.7050**	0.0992	0.8601	0.0998
SPD RED	**0.2247**	0.1139	0.4430	0.1006	0.0027	0.0002	**0.0026**	0.0001	**0.9996**	0.0001	0.9990	0.0001

Table 2: Comparison of results between our hybrid (called C-MOPSOSS) and the NSGA-II with 4000 (and 5000 for constrained problems) fitness function evaluations.

dian result with respect to the IGD metric. The true Pareto front (obtained by enumeration or using the corresponding analytical expressions, where applicable) is shown with a continuous line and the approximation produced by each algorithm is shown with circles. In Figures 8 and 9, 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 C-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, its evident that, in most cases, our approach has several points closer to the true Pareto front than the NSGA-II. In constrained problems, we can see that the performance of both approaches is very similar in Kita's problem, but in Osyczka's problem our approach gets closer to the true Pareto front with a good spread of solutions. In the Welded beam problem, the NSGA-II gets a well-distributed set of solutions along the Pareto front.

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 a low number of fitness function evaluations. Note however that 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 requires 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 having objective functions that require a very high computational cost.

Comparison between our C-MOPSOSS and MOPSO:.

We now compare the results produced by our C-MOPSOSS with respect to the MOPSO proposed in (Coello Coello et al., 2004)[4]: number of Particles = 50, number

[4]The MOPSO used was obtained from `http://www.cs.cinvestav.mx/~EVOCINV/ software.html` using

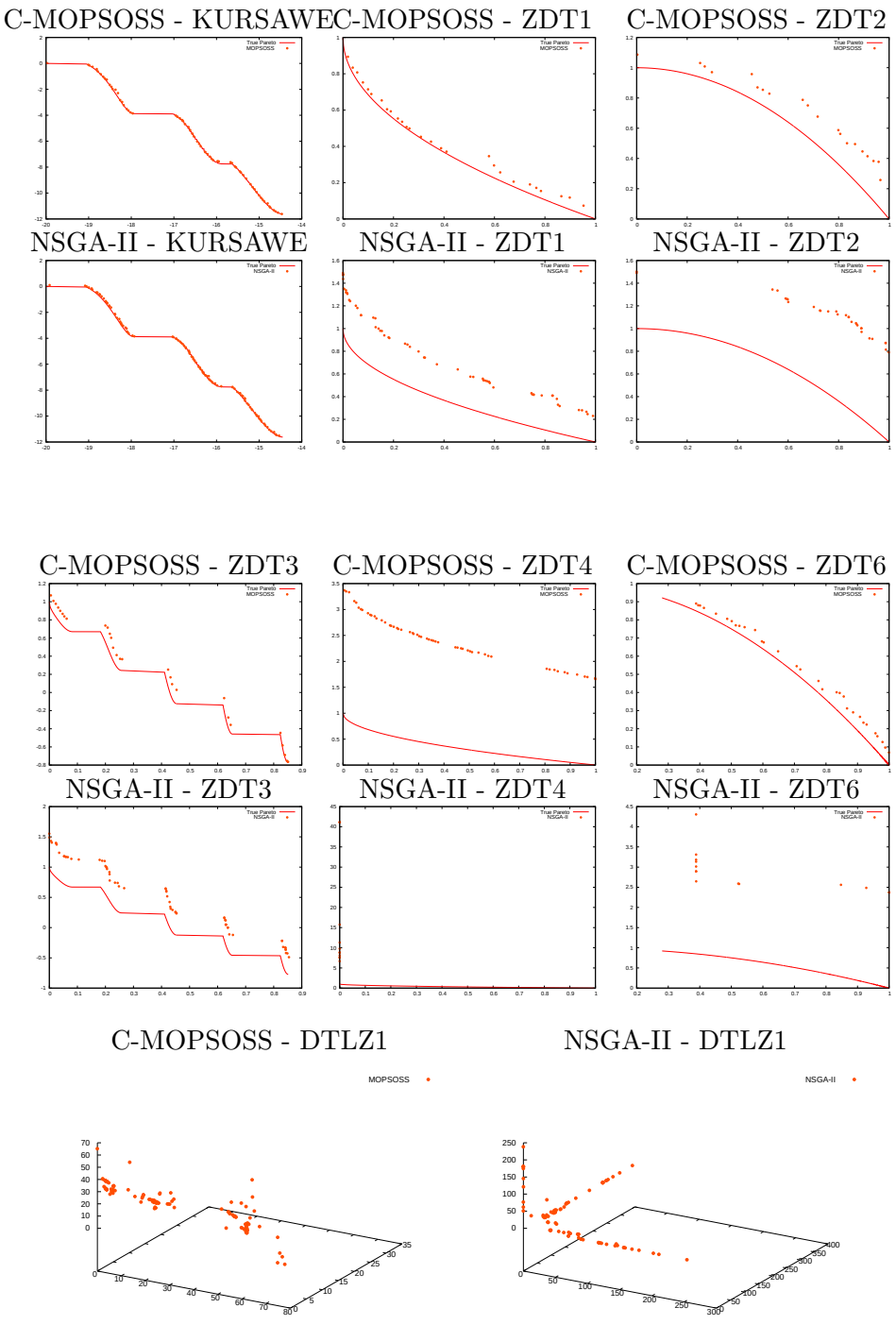

Figure 8. Pareto fronts generated by C-MOPSOSS and NSGA-II for Kursawe's, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 and DTLZ1 test functions.

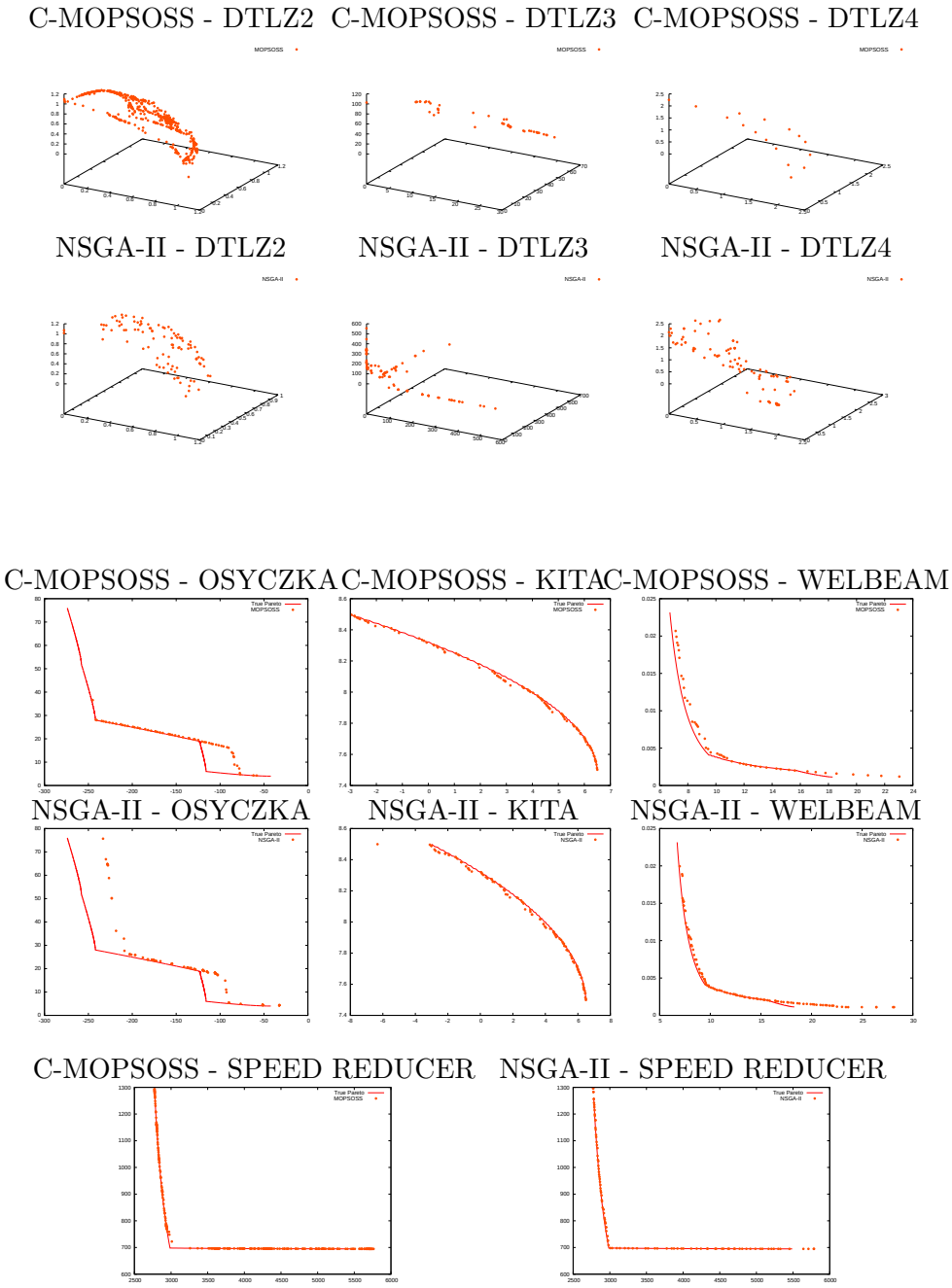

Figure 9. Pareto fronts generated by our C-MOPSOSS and the NSGA-II for the DTLZ2, DTLZ3, DTLZ4, Osyczka, Kita, Welded beam and Speed Reducer test functions.

	SC				IGD				Spread			
Function	**C**-MOPSOSS		MOPSO		**C**-MOPSOSS		MOPSO		**C**-MOPSOSS		MOPSO	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
KURSAWE	**0.1421**	0.3083	0.0852	0.0664	**0.0005**	0.0004	0.0014	0.0027	**0.3978**	0.0257	0.5715	0.0766
ZDT1	**0.0000**	0.0000	0.9374	0.0149	**0.0018**	0.0011	0.0507	0.0043	**0.3190**	0.0634	0.6362	0.0211
ZDT2	**0.0000**	0.0000	0.4662	0.4984	**0.0052**	0.0064	0.1162	0.0203	**0.3984**	0.1108	0.9287	0.0749
ZDT3	**0.0000**	0.0000	0.9530	0.0159	**0.0047**	0.0033	0.0580	0.0045	**0.6462**	0.0635	0.8396	0.0221
ZDT4	**0.0000**	0.0000	**0.0000**	0.0000	**0.1065**	0.0421	0.8777	0.6170	**0.6992**	0.0771	0.9698	0.0327
ZDT6	**0.0000**	0.0000	0.0150	0.0584	**0.0011**	0.0006	0.4041	0.0115	**0.4261**	0.1557	0.9878	0.0109
DTLZ1	**0.1680**	0.3386	0.7300	0.3157	**0.3942**	0.1181	2.6280	1.3218	**0.9939**	0.0040	0.9960	0.0038
DTLZ2	**0.0000**	0.0000	0.9730	0.0586	**0.0006**	0.0001	0.0079	0.0006	0.7600	0.0686	**0.6609**	0.0398
DTLZ3	**0.0000**	0.0000	0.7321	0.2087	**0.8367**	0.2348	3.5866	0.0699	**0.9963**	0.0029	0.9983	0.0009
DTLZ4	**0.0000**	0.0000	0.6893	0.4189	**0.0222**	0.0043	0.0431	0.0009	**0.7345**	0.1168	0.7699	0.0255

Table 3: Comparison of results between our hybrid (called C-MOPSOSS) and MOPSO with 4000 (and 5000 for constrained problems) fitness function evaluations.

of cycles = 80, mutation rate = 0.05, $w = 0.4$, $c_1 = 2.0$, $c_2 = 2.0$. For each test problem, 30 independent runs were performed. The results obtained are shown in Table 3 and correspond to the mean and standard deviation of the performance metrics (SC, IGD and S). We show in boldface the best mean values for each test function. Due to space limitations, we do not include any graphical comparison of results, and we only limited our comparison to the unconstrained test problems.

In Table 3, it can be observed that in all the test problems, our C-MOPSOSS obtained the best results with respect to the SC and IGD metrics, except for ZDT4, in which there was a tie (both obtain a zero value).[5] With respect to the S metric, we obtained the best results in all the test problems, except for DTLZ2.

From these results, we can conclude that our approach clearly outperforms MOPSO (Coello Coello et al., 2004), which is one of the few multi-objective particle swarm optimizers published in a specialized journal.

Comparing the Two Stages of our Approach:.

The intention of this subsection is to illustrate the effect of the second phase of our approach. For that sake, we adopted a version of our algorithm that only uses the first phase (i.e., without scatter search) and which we call C-MOPSO. This approach is compared to C-MOPSOSS (i.e., the version with the two phases) performing the same number of fitness function evaluations as before (4000 for unconstrained and 5000 for constrained problems) and using the same test functions. The parameters used for C-MOPSO are exactly the same as the ones described before: $P = 5$, $N = k + 1$ (k = number of objective functions), $P_m = 1/n$, $w = 0.3$, $c_1 = 0.1$, $c_2 = 1.4$. For each test problem, 30 independent runs were performed. Our results are reported in Table 4. Such results correspond to the mean and standard deviation of the performance metrics (SC, IGD and S); we show in boldface the best mean values for each test function. Figure 10 shows the graphical results produced by C-MOPSO for all the test problems adopted. The solutions displayed correspond to the

the parameters suggested by its authors in (Coello Coello et al., 2004).

[5] In ZDT4, MOPSO only generates a few solutions, but since they lie on the true Pareto front, the solutions produced by our C-MOPSOSS cannot dominate them, and viceversa. This explains the zero value obtained.

median result with respect to the IGD metric.

In Table 4, it can be observed that, with respect to the SC metric, our C-MOPSOSS obtains better results only in 8 from the 14 test functions. This result was anticipated, since allowing that the search engine performs more evaluations should produce a better approximation of the Pareto front. With respect to the IGD metric, we obtained better results with our C-MOPSOSS in all, but two cases, because this metric penalizes a poor spread of solutions, even when an algorithm has converged to the true Pareto front. With respect to the S metric, we obtained better results with our C-MOPSOSS in 13 of the 14 test functions (except for DTLZ2). In Figure 10, we can observe the Pareto fronts obtained by our C-MOPSO algorithm. In this figure, it can be observed that very few solutions are obtained in each case, and that they present a poor distribution along the Pareto front.

Based on these results, we conclude that the use of the second phase of our approach is beneficial to improve the spread of solutions along the Pareto front. It is worth noting, however, that such an improvement in spread is obtained by sacrificing a slight improvement in terms of convergence.

	SC				**IGD**				**Spread**			
Function	**C**-MOPSOSS		**C**-MOPSO		**C**-MOPSOSS		**C**-MOPSO		**C**-MOPSOSS		**C**-MOPSO	
	Mean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ
KURSAWE	**0.0719**	0.0319	0.3745	0.0685	**0.0005**	0.0000	0.0009	0.0001	**0.3978**	0.0257	0.6830	0.0652
ZDT1	0.4155	0.1699	**0.1071**	0.0737	**0.0018**	0.0011	**0.0018**	0.0006	**0.3190**	0.0634	0.7102	0.0841
ZDT2	0.3049	0.2377	**0.1197**	0.1704	0.0052	0.0064	**0.0037**	0.0048	**0.3984**	0.1108	0.5374	0.0795
ZDT3	0.3043	0.1511	**0.1404**	0.0839	**0.0047**	0.0033	0.0110	0.0045	**0.6462**	0.0635	1.1051	0.1057
ZDT4	**0.0000**	0.0000	0.7444	0.2291	**0.1065**	0.0421	0.8452	0.4025	**0.6992**	0.0771	0.9741	0.0201
ZDT6	0.7702	0.2254	**0.0465**	0.0894	0.0011	0.0006	**0.0007**	0.0007	**0.4261**	0.1557	0.6280	0.2688
DTLZ1	**0.1299**	0.1699	0.4067	0.2445	**0.3942**	0.2432	0.9439	0.2790	**0.9930**	0.0040	0.9939	0.0050
DTLZ2	0.1391	0.0947	**0.0627**	0.0672	**0.0006**	0.0001	0.0008	0.0003	0.7600	0.0686	**0.4643**	0.0688
DTLZ3	**0.0735**	0.1139	0.3521	0.2449	**0.8367**	0.2348	1.0328	0.3063	**0.9963**	0.0029	0.9967	0.0025
DTLZ4	**0.1604**	0.2792	0.1843	0.3085	**0.0222**	0.0043	0.0246	0.0033	**0.7345**	0.1168	0.7517	0.1175
OSYCZKA	**0.2077**	0.2261	0.4020	0.2560	**0.0020**	0.0011	0.0026	0.0013	**0.9667**	0.0173	0.9793	0.0156
KITA	0.3857	0.1014	**0.2055**	0.0512	**0.0009**	0.0001	0.0029	0.0015	**0.5620**	0.0964	0.8585	0.1252
WELBEAM	**0.2131**	0.2377	0.5533	0.2366	**0.0047**	0.0059	0.0085	0.0048	**0.7050**	0.0992	0.7226	0.0969
SPD RED	**0.0203**	0.0970	0.3297	0.1459	**0.0027**	0.0002	0.0034	0.0013	**0.9996**	0.0000	0.9999	0.0001

Table 4: Comparison of results between our hybrid (called C-MOPSOSS) and a version that only uses the first phase (called C-MOPSO) with 4000 (and 5000 for constrained problems) fitness function evaluations.

Convergence Capability of our C-MOPSOSS:.

Here, we show that our proposed hybrid approach, C-MOPSOSS, is capable of converging to the true Pareto front of all the test functions adopted in our study, if allowed a sufficiently large number of fitness function evaluations. Table 5 shows the total number of fitness function evaluations required to reach the true Pareto front for each test function. Figure 11 shows these results in graphical form. No metrics are adopted in this case, since we only aimed to illustrate the convergence capabilities of our proposed approach.

Conclusions

We have presented a hybrid between a MOEA based on PSO and a local search mechanism based on scatter search to solve both unconstrained and constrained multi-

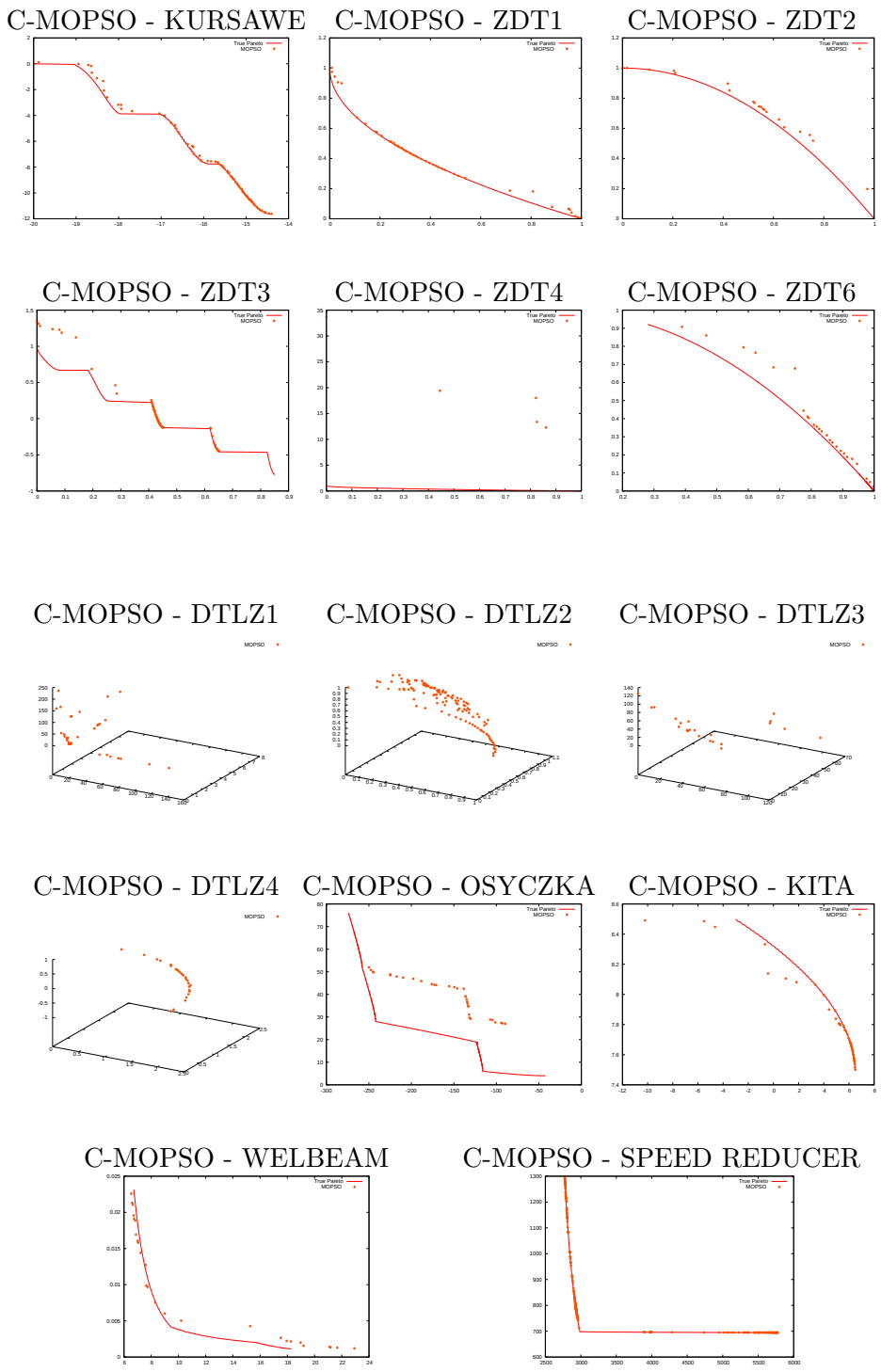

Figure 10. Pareto fronts generated by C-MOPSO for all the 13 test functions adopted.

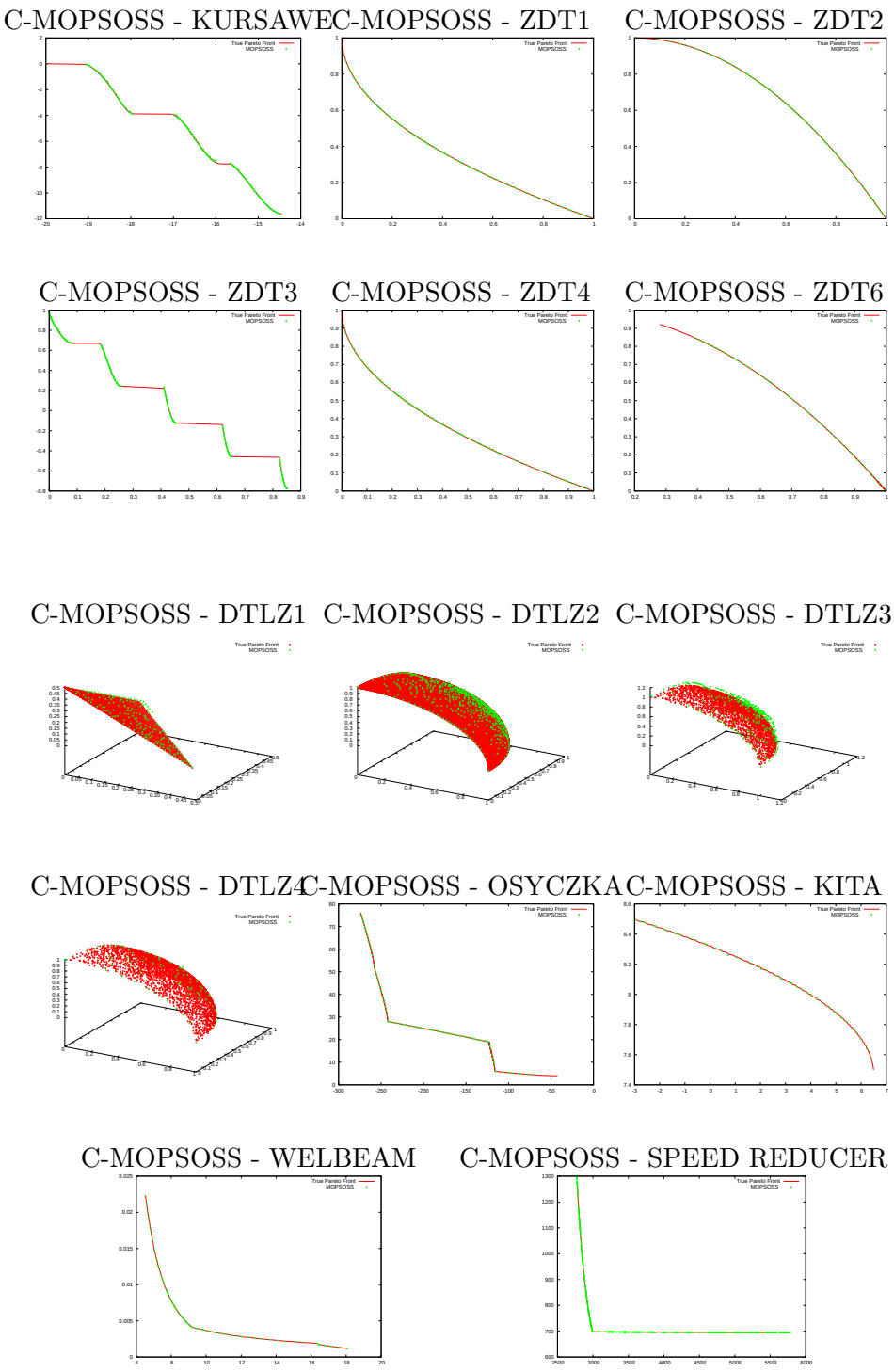

Figure 11. Pareto fronts generated by C-MOPSO for all the 14 test functions adopted.

Function	# eval	Function	# eval
Kursawe	7,000	DTLZ2	7,000
ZDT1	7,000	DTLZ3	150,000
ZDT2	7,000	DTLZ4	150,000
ZDT3	7,000	OSYCZKA	30,000
ZDT4	30,000	KITA	10,000
ZDT6	7,000	WELDED BEAM	10,000
DTLZ1	150,000	SPEED REDUCER	7,000

Table 5: Number of Fitness Function Evaluations required to reach the true Pareto front in all the test problems adopted.

objective optimization problems. This hybrid combines the high convergence rate of PSO with the good neighborhood exploration capabilities of the scatter search algorithm. We have also found out that the leader selection scheme adopted in PSO is a key element of our algorithm, since it is responsible of the high convergence that our approach requires. With SS we observe that the selection of solutions closer to the Pareto front generates smooth moves that give us more solutions closer to the true Pareto front of the problem being solved. Our approach also adopts the concept of ϵ-dominance to produce well-distributed sets of nondominated solutions. The results obtained are very competitive with respect to the NSGA-II in problems whose dimensionality goes from 3 up to 30 decision variables.

The main aim of this chapter has been to present practitioners a tool that they can use in applications in which each fitness function evaluation consumes a considerable amount of CPU time (such applications are common in areas such as aeronautical engineering). In those cases, the approach proposed in this chapter may provide reasonably good approximations of the true Pareto front at an affordable computational cost.

Future Research Directions

The design of highly efficient multi-objective particle swarm optimizers is an area that certainly deserves more research. Next, some of the most promising future research directions within this topic are briefly described.

One interesting path for future work is to improve the performance of the PSO approach proposed in this chapter. The leader selection mechanism is of particular interest, since it is known to be a critical element in any multi-objective particle swarm optimizer (Branke & Mostaghim, 2006). Additionally, the implementation adopted for the ϵ-dominance mechanism needs to be improved, since it currently loses the extremes of the Pareto front when dealing with segments that are almost horizontal or vertical. A variation of ϵ-dominance such as the one proposed in (Hernández-Díaz, Santana-Quintero, Coello, & Molina, 2006) could be adopted to deal with this problem. Additionally, the ideas presented in this chapter may also serve as a basis to design entirely new multi-objective particle swarm optimizers whose main design emphasis is efficiency.

Another interesting path for future research is the exploration of self-adaptation and

online adaptation schemes that allow the design of parameterless multi-objective particle swarm optimizers. In order to reach this goal, it is necessary to have an in-depth knowledge of the behavior of multi-objective particle swarm optimizers when dealing with a wide variety of different test problems (see for example (Toscano Pulido, 2005)). However, a proper setting of the parameters of a multi-objective particle swarm optimizer is not an easy task, and a lot of research is still needed in this direction. Nevertheless, progress in this research area would allow the design of mechanisms to properly deal with the (potentially wide) variety of situations that could arise in practice.

Although we propose in this chapter the use of scatter search as a diversification mechanism, other techniques are also possible (see for example (Hernández-Díaz, Santana-Quintero, Coello Coello, Caballero, & Molina, 2006), in which rough sets are adopted for this sake). It is worth noting, however, that attention must be paid to the computational cost of the diversification technique adopted, since such cost must be as low as possible in order to be of practical use.

A more extended use of multi-objective particle swarm optimizers in real-world applications is expected to happen within the next few years. The relative simplicity of implementation and ease of use of multi-objective particle swarm optimizers makes them a suitable candidate for practitioners to use. Thus, an important number of real-world applications of this sort of heuristic are expected to appear in the next few years.

Table 6:: MOP Test Functions

Function	**Definition**	**Bounds**
WELDED BEAM (Ragsdell & Phillips, 1975)	$F = (f_1(\vec{x}), f_2(\vec{x}))$, where: $f_1(\vec{x}) = 1.10471h^2l + 0.04811tb(14.0 + l)$, $f_2(\vec{x}) = \delta(\vec{x})$ subject to: $g_1(\vec{x}) \equiv 13600 - \tau(\vec{x}) \geq 0$, $g_2(\vec{x}) \equiv 30000 - \sigma(\vec{x}) \geq 0$, $g_3(\vec{x}) \equiv b - h \geq 0$, $g_4(\vec{x}) \equiv P_c - 6000 \geq 0$. where: $\tau(\vec{x}) = \sqrt{\frac{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')}{\sqrt{0.25(l^2 + (h+t)^2)}}}$, $\tau' = \frac{6000}{\sqrt{2}hl}$, $\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h+t)^2)\}}$, $\sigma(\vec{x}) = \frac{504000}{t^2b}$, $P_c(\vec{x}) = 64746.022(1 - 0.0282346t)tb^3$.	$0.125 \leq h, b \leq 5.0$, $0.1 \leq l, t \leq 10.0$.

Table 6:: (continued)

Function	Definition	Bounds
SPEED REDUCER (J., 1970)	$F = (f_1(\vec{x}), f_2(\vec{x}))$, where: $f_1(\vec{x}) = 0.7854x_1 \cdot x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.5079(x_6^2 + x_7^2)x_1 + 7.477(x_6^3 + x_7^3) + 0.7854(x_4 \cdot x_6^2 + x_5 \cdot x_7^2)$ $f_2(\vec{x}) = \frac{\sqrt{(\frac{745 \cdot x_4}{x_2 x_3})^2 + 1.69e^7}}{0.1x_6^3}$ subject to: $g_1(\vec{x}) \equiv \frac{1}{x_1 \cdot x_2^2 \cdot x_3} - \frac{1}{27} \leq 0,$ $g_2(\vec{x}) \equiv \frac{1}{x_1 \cdot x_2^2 \cdot x_3^2} - \frac{1}{397.5} \leq 0,$ $g_3(\vec{x}) \equiv \frac{x_3^3}{x_2 \cdot x_3 \cdot x_6^4} - \frac{1}{1.93} \leq 0,$ $g_4(\vec{x}) \equiv \frac{x_5^3}{x_2 \cdot x_3 \cdot x_7^4} - \frac{1}{1.93} \leq 0,$ $g_5(\vec{x}) \equiv x_2 \cdot x_3 - 40 \leq 0,$ $g_6(\vec{x}) \equiv \frac{x_1}{x_2} - 12 \leq 0,$ $g_7(\vec{x}) \equiv 5 - \frac{x_1}{x_2} \leq 0,$ $g_8(\vec{x}) \equiv 1.9 - x_4 + 1.5x_6 \leq 0,$ $g_9(\vec{x}) \equiv 1.9 - x_5 + 1.1x_7 \leq 0,$ $g_{10}(\vec{x}) \equiv f_2 - 1300 \leq 0,$ $g_{11}(\vec{x}) \equiv \frac{\sqrt{(\frac{745 \cdot x_5}{x_1 \cdot x_2})^2 + 1.575e^8}}{0.1 \cdot x_6^3} - 1100 \leq 0,$	$2.6 \leq x_1 \leq 3.6,$ $0.7 \leq x_2 \leq 0.8,$ $17 \leq x_3 \leq 28$ $7.3 \leq x_4, x_5 \leq 8.3,$ $2.9 \leq x_6 \leq 3.9,$ $5.0 \leq x_7 \leq 5.5$

References

Alvarez-Benitez, J. E., Everson, R. M., & Fieldsend, J. E. (2005, March). A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In C. A. C. Coello, A. H. Aguirre, & E. Zitzler (Eds.), *Evolutionary multi-criterion optimization. third international conference, emo 2005* (pp. 459–473). Guanajuato, México: Springer. LNCS Vol. 3410.

Bartz-Beielstein, T., Limbourg, P., Parsopoulos, K. E., Vrahatis, M. N., Mehnen, J., & Schmitt, K. (2003, December). Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 congress on evolutionary computation (cec'2003)* (Vol. 3, pp. 1780–1787). Canberra, Australia: IEEE Press.

Baumgartner, U., Magele, C., & Renhart, W.(2004, March). Pareto Optimality and Particle Swarm Optimization. *IEEE Transactions on Magnetics*, *40*(2), 1172–1175.

Branke, J., & Mostaghim, S. (2006, September). About Selecting the Personal Best in Multi-Objective Particle Swarm Optimization. In T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, & X. Yao (Eds.), *Parallel problem solving from nature - ppsn ix, 9th international conference* (pp. 523–532). Reykjavik, Iceland: Springer. Lecture Notes in Computer Science Vol. 4193.

Coello Coello, C. A., & Cruz Cortés, N. (2005, June). Solving Multiobjective Optimization Problems using an Artificial Immune System. *Genetic Programming and Evolvable Machines*, *6*(2), 163–190.

Coello Coello, C. A., & Salazar Lechuga, M. (2002, May). MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on evolutionary computation (cec'2002)* (Vol. 2, pp. 1051–1056). Piscataway, New Jersey: IEEE Service Center.

Coello Coello, C. A., Toscano Pulido, G., & Salazar Lechuga, M. (2004, June). Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers. (ISBN 0-3064-6762-3)

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons. (ISBN 0-471-87339-X)

Deb, K., & Agrawal, R. B. (1995). Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, *9*, 115–148.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002, April). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multiobjective Optimization. In A. Abraham, L. Jain, & R. Goldberg (Eds.), *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications* (pp. 105–145). USA: Springer.

Eshelman, L. J., & Schaffer, J. D. (1993). Real-coded Genetic Algorithms and Interval-Schemata. In L. D. Whitley (Ed.), *Foundations of genetic algorithms 2* (pp. 187–202). California: Morgan Kaufmann Publishers.

Fieldsend, J. E., & Singh, S. (2002, September). A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 u.k. workshop on computational intelligence* (pp. 37–44). Birmingham, UK.

Glover, F. (1977). Heuristic for integer programming using surrogate constraints. *Decision Sciences 8*, 156-166.

Glover, F. (1994). Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, *49*(1-3), 231–255.

Glover, F. (1998). A template for scatter search and path relinking. In *Ae '97: Selected papers from the third european conference on artificial evolution* (p. 13-54). London, UK: Springer-Verlag.

Hernández-Díaz, A. G., Santana-Quintero, L. V., Coello, C. A. C., & Molina, J. (2006, March). *Pareto-adaptive ε-dominance* (Tech. Rep. No. EVOCINV-02-2006). México: Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN.

Hernández-Díaz, A. G., Santana-Quintero, L. V., Coello Coello, C., Caballero, R., & Molina, J. (2006, July). A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory. In M. K. et al. (Ed.), *2006 genetic and evolutionary computation conference (gecco'2006)* (Vol. 1, pp. 675–682). Seattle, Washington, USA: ACM Press. ISBN 1-59593-186-4.

J., G. (1970). Optimal synthesis problems solved by means oj nonmlinear programming and random methods. *Journal of Mechanisms*, *5*, 287 - 309.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. California, USA: Morgan Kaufmann Publishers.

Kita, H., Yabumoto, Y., Mori, N., & Nishikawa, Y. (1996, September). Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In H.-M. Voigt, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature—ppsn iv* (pp. 504–512). Berlin, Germany: Springer-Verlag.

Knowles, J. D., & Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, *8*(2), 149–172.

Kursawe, F. (1991, October). A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel & R. Männer (Eds.), *Parallel problem solving from nature. 1st workshop, ppsn i* (Vol. 496, pp. 193–197). Berlin, Germany: Springer-Verlag.

Laguna, M., & Martí, R. (2003). *Scatter search: Methodology and implementations in c*. Kluwer Academic Publishers.

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. (2002, Fall). Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation*, *10*(3), 263–282.

Mahfouf, M., Chen, M.-Y., & Linkens, D. A. (2004, September). Adaptive Weighted Particle Swarm Optimisation for Multi-objective Optimal Design of Alloy Steels. In *Parallel problem solving from nature - ppsn viii* (pp. 762–771). Birmingham, UK: Springer-Verlag. LNCS Vol. 3242.

Moore, J., & Chapman, R. (1999). *Application of Particle Swarm to Multiobjective Optimization.* (Department of Computer Science and Software Engineering, Auburn University. (Unpublished manuscript))

Mostaghim, S., & Teich, J. (2003, April). Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *2003 ieee sis proceedings* (pp. 26–33). Indianapolis, USA: IEEE Service Center.

Osyczka, A., & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, *10*, 94–99.

Parsopoulos, K., Tasoulis, D., & Vrahatis, M. (2004, February). Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization. In *Proceedings of the iasted international conference on artificial intelligence and applications (aia 2004)* (Vol. 2, pp. 823–828). Innsbruck, Austria: ACTA Press.

Ragsdell, K. M., & Phillips, D. T. (1975). Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry Series B*, *B*(98), 1021–1025.

Reyes Sierra, M., & Coello Coello, C. A. (2005, June). Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *2005 ieee swarm intelligence symposium (sis'05)* (pp. 116–123). Pasadena, California, USA: IEEE Press.

Reyes-Sierra, M., & Coello Coello, C. A. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, *2*(3), 287–308.

Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic algorithms and their applications: Proceedings of the first international conference on genetic algorithms* (pp. 93–100). Lawrence Erlbaum.

Smith, R. E., Dike, B. A., & Stegmann, S. A. (1995). Fitness Inheritance in Genetic Algorithms. In *Sac '95: Proceedings of the 1995 acm symposium on applied computing* (pp. 345–350). Nashville, Tennessee, USA: ACM Press.

Toscano Pulido, G. (2005). *On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization*. Unpublished doctoral dissertation, Computer Science Section, Department of Electrical Engineering, CINVESTAV-IPN, Mexico.

Veldhuizen, D. A. V., & Lamont, G. B. (1998). *Multiobjective Evolutionary Algorithm Research: A History and Analysis* (Tech. Rep. No. TR-98-03). Wright-Patterson AFB, Ohio: Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology.

Veldhuizen, D. A. V., & Lamont, G. B. (2000, July). On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 congress on evolutionary computation* (Vol. 1, pp. 204–211). Piscataway, New Jersey: IEEE Service Center.

Zitzler, E., Deb, K., & Thiele, L. (2000, Summer). Comparison of Multiobjective Evolutionary Algorithms:Empirical Results. *Evolutionary Computation*, *8*(2), 173–195.

Zitzler, E., & Thiele, L. (1999, November). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

Additional Reading

There are several good (general) references on particle swarm optimization. The following are suggested as additional readings:

1. James Kennedy and Russell C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, California, 2001, ISBN 1-55860-595-9.

2. Andries P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, Ltd, 2005, ISBN 978-0-470-09191-3.

There are also several references that focus on evolutionary multi-objective optimization and others that specifically deal with multi-objective particle swarm optimizers. See for example:

1. Carlos A. Coello Coello, David A. Van Veldhuizen and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, March 2002, ISBN 0-3064-6762-3.

2. Kalyanmoy Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001, ISBN 0-471-87339-X.

3. K.C. Tan, E.F. Khor and T.H. Lee, *Multiobjective Evolutionary Algorithms and Applications*, Springer-Verlag, London, 2005, ISBN 1-85233-836-9.

4. Margarita Reyes-Sierra and Carlos A. Coello Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art", *International Journal of Computational Intelligence Research*, Vol. 2, No. 3, pp. 287–308, 2006.

5. Carlos A. Coello Coello, Gregorio Toscano Pulido and Maximino Salazar Lechuga, "Handling Multiple Objectives With Particle Swarm Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 256–279, June 2004.

6. Gregorio Toscano-Pulido, Carlos A. Coello Coello and Luis Vicente Santana-Quintero, "EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency", in Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu and

Tadahiko Murata (editors), *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pp. 272–285, Springer. Lecture Notes in Computer Science Vol. 4403, Matshushima, Japan, March 2007.

Finally, there are also several papers dealing with real-world applications of multi-objective particle swarm optimizers that are worth reading. See for example:

1. M.K. Gill, Y.H. Kaheil, A. Khalil, M. Mckee and L. Bastidas, "Multiobjective particle swarm optimization for parameter estimation in hydrology", *Water Resources Research*, Vol. 42, No. 7, Art. No. W07417, July 22, 2006.

2. Alexandre M. Baltar and Darrell G. Fontane, "A generalized multiobjective particle swarm optimization solver for spreadsheet models: application to water quality", in *Hydrology Days 2006*, Fort Collins, Colorado, USA, March 2006.

3. M. Janga Reddy and D. Nagesh Kumar, "An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design", *Engineering Optimization*, Vol. 39, No. 1, pp. 49–68, January, 2007.