A Study of Techniques to Improve the Efficiency of a Multi-Objective Particle Swarm Optimizer

Margarita Reyes-Sierra and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Ingeniería Eléctrica, Sección Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07360, México
mreyes@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Summary. The use of particle swarm optimization (PSO) in multi-objective optimization has become widespread in the last few years. However, very few researchers have explored the use of techniques that allow to reduce the number of fitness evaluations of a PSO-based approach for multi-objective optimization. This chapter precisely explores the advantages and disadvantages of using fitness inheritance and approximation techniques to reduce the number of fitness evaluations into a PSO-based multi-objective algorithm previously proposed by the authors. Our study includes fifteen fitness inheritance techniques and four approximation techniques which are applied to a set of test functions taken from the specialized literature.

1 Introduction

Given the high computational cost of evaluating the fitness functions of many real-world applications, the total number of fitness function evaluations that an Evolutionary Algorithm (EA) may perform in such applications may become severely limited. In order to improve the performance of EAs, several enhancement techniques have been proposed in the past. In this chapter, we will focus on two of these enhancement techniques: fitness inheritance and approximation techniques. Fitness Inheritance is an enhancement technique [1] in which the fitness value of an offspring is obtained from the fitness values of its parents. On the other hand, approximation techniques [2] let us estimate the fitness of an individual using the previously calculated fitness of its neighbors (either including its parents or not). In fact, fitness inheritance is a particular case of fitness approximation. However, in general, the idea of using enhancement techniques, is that we do not need to evaluate every individual at each generation, such that the total computational cost can be reduced. In this chapter, we perform a study of different inheritance and approximation techniques applied to a real-coded PSO-based approach that has

been previously proposed by the authors to solve multi-objective problems [3]. Since the main focus of our research is on fitness inheritance techniques, we are proposing a higher number of fitness inheritance techniques (fifteen) than approximation techniques (four). In our study, we use five well-known multi-objective test functions in an attempt to determine the best from the enhancement techniques analyzed.

In the final part of our study, once we have identified which were the best enhancement techniques, we compare these techniques with respect to other PSO-based multi-objective optimizers which are representative of the state-of-the-art, adopting different test functions.

The remainder of this chapter is organized as follows. An introduction to Fitness Inheritance and Fitness Approximation is given in Sections 2 and 3, respectively. Section 4 introduces the multi-objective PSO-based algorithm in which the proposed techniques are incorporated. The enhancement techniques proposed in this paper are presented in Section 5. In Sections 6 and 7 we present the obtained results and their discussion, respectively. A comparison against other PSO-based algorithms is presented in Section 8. Finally, our conclusions and some of the possible paths for future research are described in Section 9.

2 Fitness Inheritance

The use of fitness inheritance to improve the performance of Genetic Algorithms (GAs) was originally proposed by Smith et al. [1]. The authors proposed two possible ways of inheriting fitness: the first consists of taking the average fitnesses of the two parents and the other consists of taking a weighted (proportional) average of the fitnesses of the two parents. The second approach is related to how similar the offspring is with respect to its parents (this is done using a similarity measure). They applied inheritance to a very simple problem (the OneMax problem) [1] and found that the weighted fitness average resulted in a better performance and indicated that fitness inheritance was a viable alternative to reduce the computational cost of a genetic algorithm.

Zheng et al. [4] performed the first application of fitness inheritance on a GA for the problem of codebook design in data compression techniques. They concluded that the use of fitness inheritance didn't degrade the performance of the GA.

Sastry et al. [5] provided some theoretical foundations for fitness inheritance. They investigated convergence times, population sizing and the optimal proportion of inheritance for the OneMax problem. Chen et al. [6] investigated fitness inheritance as a way to speed up multi-objective GAs and EAs. They extended the analytical model proposed by Sastry et al. to multi-objective problems. Convergence and population-sizing models are derived and compared with respect to experimental results. The authors concluded that the

number of function evaluations can be reduced with the use of fitness inheritance.

Salami et al. [7] proposed a "Fast Evolutionary Algorithm" in which a fitness and associated reliability value are assigned to each new individual that is only evaluated using the true fitness function if the reliability value is below a certain threshold. Also, they incorporated random evaluation and error compensation strategies. The authors obtained an average reduction of 40% in the number of evaluations while obtaining similar solutions. In the same work, they presented an application of fitness inheritance to image compression obtaining reductions between 35% and 42% of the number of evaluations.

Ducheyne et al. [8] tested the performance of average and weighted average fitness inheritance on a well-known test suite of multi-objective optimization problems [9], using a binary GA. They concluded that the fitness inheritance efficiency enhancement techniques can be used in order to reduce the number of fitness evaluations provided that the Pareto front is convex and continuous. They also concluded that if the Pareto surface is not convex or if it is discontinuous, the fitness inheritance strategies fail to reach the true Pareto front.

Pelikan et al. [10] used fitness inheritance to estimate fitness for a proportion of solutions in the Bayesian Optimization Algorithm (BOA). They concluded that fitness inheritance is a promising concept in BOA, because population-sizing requirements for building appropriate models of promising solutions lead to good fitness estimates even if only a small proportion of candidate solutions is evaluated using the true fitness function.

Bui et al. [11] performed a comparison of the performance of anti-noise methods, particularly probabilistic and re-sampling methods, using the NSGA-II [12]. They applied the concept of fitness inheritance to both types of methods in order to reduce computational time. The authors obtained a substantial amount of savings in the number of computations, reaching a peak of 30% of savings without deteriorating the performance.

In a previous work [13], we proposed the first attempt to incorporate the concept of fitness inheritance to a real-coded Multi-Objective PSO (MOPSO) previously proposed by us [3]. In [13], we tested the performance of weighted average fitness inheritance on a well-known test suite of multi-objective optimization problems [9]. Based on the obtained results, we concluded that fitness inheritance reduces the computational cost without decreasing the quality of the results in a significant way. Also, the fitness inheritance technique used was able to generate non-convex and discontinuous Pareto fronts. These conclusions were somewhat surprising given the conclusions (pointing in the exact opposite direction) previously obtained by Ducheyne et al. in [8].

3 Fitness Approximation

A promising possibility when an evaluation is very time-consuming or expensive is not to evaluate every individual, but just estimate the quality of some of the individuals based on an approximate model of the fitness landscape.

Approximation techniques estimate individual fitness on the basis of previously observed objective function values of neighboring individuals. There are many possible approximation models. In the simplest case, the fitness of a new individual is derived from its parents' fitnesses (fitness inheritance). However, some other more complicated methods have been used. A survey of approximation methods in evolutionary computation can be found in [2]. Here, we briefly mention a few examples of different approximation techniques commonly used.

Ratle [14] presented a new approach based on a classical real-encoded genetic algorithm for accelerating convergence of evolutionary optimization methods. With this aim, a reduction in the number of fitness function calls is obtained by means of an approximate model of the fitness landscape using kriging interpolation. The author builds a statistical model from a small number of data points obtained during one or more generations of the evolutionary method using the true fitness landscape. The model is updated every time a convergence criteria is reached.

Jin et al. [15] investigated the convergence property of an evolution strategy with neural network based fitness evaluations. In this work, the authors introduce the concept of *controlled evolution*, in which, the evolution proceeds using not only the approximate fitness function, but also the true fitness function. They also introduce two possibilities to combine the true with the approximate fitness function: the *controlled individuals* approach and the *controlled generation* approach. The authors define *controlled* as *true evaluated*. Both approaches are studied and some interesting conclusions/recommendations about the correct use of such techniques are provided.

Sano et al. [16] proposed a genetic algorithm for optimization of continuous noisy fitness functions. In this approach, the authors utilize the history of the search to reduce the number of fitness evaluations. The fitness of a novel individual is estimated using the fitness values of the other individuals as well as the sampled fitness values for it. So, as to increase the number of individuals adopted for evaluation, they use not only the current generation but also the whole history of the search. To utilize the history of the search, a stochastic model of the fitness function is introduced, and the maximum likelihood technique is used for estimation of the fitness function. The authors concluded that the proposed method outperforms a conventional GA in noisy environments.

Branke et al. [17] suggest the use of local regression for estimation, taking the fitness of neighboring individuals into account. Since in local regression is very important to determine which individuals belong to the neighborhood of a given individual, the authors studied two different approaches for setting the

value of the size of the local neighborhood (*relative neighborhood* and *adaptive neighborhood*). The authors concluded that local regression provides better estimations than previously proposed approaches. In more recent work, Branke et al. [18] compared two estimation methods: interpolation and regression. They concluded that regression seems to be slightly preferable, particularly if only a very small fraction of the individuals in the population is evaluated. Their experiments also show that using fitness estimation, it is possible to either reach a better fitness level in a given time, or to reach a desired fitness level much faster (using roughly a half of the original number of fitness function evaluations).

Ong et al. [19] proposed a local surrogate modeling algorithm for parallel evolutionary optimization of computationally expensive problems. The proposed algorithm combines hybrid evolutionary optimization techniques, radial basis functions, and trust-region frameworks. The main idea of the proposed approach is using an EA combined with a feasible sequential quadratic programming solver. Each individual within an EA generation is used as an initial solution for local search, based on Lamarckian learning. The authors employ a trust-region framework to manage the interaction between the original objective and constraint functions and the computationally cheap surrogate models (which consist of radial basis networks constructed by using data points in the neighborhood of the initial solution), during local search. Extensive numerical studies are presented for some benchmark test functions and an aerodynamic wing design problem. The authors show that the proposed framework provides good designs on a limited computational budget. In more recent work, Ong et al. [20] present a study on the effects of uncertainty in the surrogate on Surrogate-Assisted Evolutionary Algorithms (SAEA). In particular, the authors focus on both the "curse of uncertainty" (impairments due to errors in the approximation) and "blessing of uncertainty" (benefits of approximation errors). The authors tested several algorithms: the Surrogated-Assisted Memetic Algorithm (SAMA) proposed in [19], a standard genetic algorithm, a memetic algorithm (considered as the standard hybridization of a genetic algorithm and the feasible sequential quadratic programming solver used in [19]) and the SAMA-Perfect algorithm (which is the SAMA algorithm but using the exact fitness function as surrogate model), on three multimodal benchmark problems (Ackley, Griewank and Rastrigin). The authors concluded that approximation errors lead to convergence at false global optima, but prove to be beneficial in some cases, accelerating the evolutionary search.

Gaspar-Cunha et al. [21] developed an algorithm using artificial neural networks to reduce the number of exact function evaluations for multi-objective optimization problems. The proposed method can save, in some cases, more than 50% of true function evaluations.

Regis and Shoemakes [22] developed an approach for the optimization of continuous costly functions that uses a space-filling experimental design and local function approximation to reduce the number of function evaluations in an evolutionary algorithm. The proposed approach estimates the objective

function value of an offspring by means of a function approximation model over the k nearest previously evaluated points. The estimated values are used to identify the most promising offspring per function evaluation. A Symmetric Latin Hypercube Design (SLHD) is used to determine initial points for function evaluation, and for the construction of the function approximation models. The authors compared the performance of an Evolution Strategy (ES) with local quadratic approximation, an ES with local cubic radial basis function interpolation, an ES whose initial parent population is obtained from a SLHD, and a conventional ES (in all cases, the authors use a (μ, λ)-ES with uncorrelated mutations). The algorithms are tested on a groundwater bioremediation problem and on some benchmark test functions for global optimization (including Dixon-Szegö, Rastrigin and Ackley). The obtained results (which include analysis of variance to provide stronger and solid claims regarding the relative performance of the algorithms) suggest that the approach that uses SLHDs together with local function approximations has potential for success in enhancing EAs for computationally expensive real-world problems. Also, the cubic radial basis function approach appears to be more promising than the quadratic approximation approach on difficult higher-dimensional problems.

Dim et al. [23] presented a Trusted Evolutionary Algorithm (TEA) for solving optimization problems with computationally expensive fitness functions. The TEA is designed to maintain good trustworthiness of the surrogate models in predicting fitness improvements or controlling approximation errors throughout the evolutionary search. In this case, the authors are more interested in predicting search improvement as opposed to the quality of the approximation, which is regarded as a secondary objective. TEA begins its search using the canonical EA, with only exact function evaluations. During the canonical EA search, the exact fitness values obtained are archived in a central database together with the design vectors (to be used later for constructing surrogate models). After some initial search generations (specified by the user), the trust region approach takes place beginning from the best solution provided by the canonical EA. The trust region approach uses a surrogate model (radial basis neural networks) and contracts or expands the trust radius depending on the ability of the approximation model in predicting fitness improvements, until the termination conditions are reached. The authors performed an empirical study on two highly multi-modal benchmark functions commonly used in the global optimization literature (Ackley and Griewank). Numerical comparisons to the canonical EA and the original trust region line search framework are also reported. From the obtained results, the authors concluded that TEA converges to near-optimum solutions more efficiently than the canonical evolutionary algorithm.

Voutchkov and Keane [24] discussed the idea of using surrogate models for multi-objective optimization. That is, instead of using the expensive computational methods during the optimization, they used a much cheaper but accurate replica. The authors aim to overview the usage of several methods,

using the NSGA-II algorithm [12] as their search engine. Several different surrogate models are used, including radial basis functions, regression models and kriging. The authors tested the surrogate models on four different test functions taken from the specialized multi-objective optimization literature. From the obtained results, the authors concluded that the performance of all methods depends on the features of the objective functions being optimized. Also, the authors discussed the weaknesses of these models on deceptive problems. In general, they concluded that the kriging method appears to perform well in most situations, however, it is much more computationally expensive than the rest.

Knowles [25] proposed an algorithm called ParEGO, which is a hybrid approach with on-line landscape approximation for expensive multi-objective optimization problems. ParEGO is an extension of the Efficient Global Optimization (EGO) algorithm, which is a frequently cited algorithm from the global optimization literature, designed for expensive functions of low dimension. The EGO algorithm makes use of kriging to model the search landscape from solutions visited during the search. Specifically, it exploits a version of the Design and Analysis of Computer Experiments (DACE) model, which is based on Gaussian processes. ParEGO extends the EGO algorithm for solving multi-objective problems by converting the k different cost values (objectives) of a solution into a single cost via a parameterized scalarizing weight vector (using the augmented Tchebycheff function). By choosing a different weight vector at each iteration of the search, an approximation to the whole Pareto front is built up gradually. The author tested the ParEGO algorithm on a test suite of nine difficult, but low-dimensional, multi-objective functions of limited ruggedness, over just 100 and 250 function evaluations. Also, the obtained results are compared against those obtained by the NSGA-II algorithm [12] (performing 100 and 260 function evaluations) and a random search approach (over 10000 function evaluations). From the obtained results, the author concluded that both ParEGO and NSGA-II outperform the random search, even over a very small number of function evaluations, and that ParEGO generally outperforms NSGA-II on the test functions adopted, at both 100 and 250 function evaluations (especially when the worst case performance is measured). Overall, the author concluded that ParEGO exhibits a promising performance for multi-objective optimization problems where evaluations are expensive or otherwise restricted in number.

In this chapter, we adopt very simple approximation techniques, based only on the objective values of the closest neighbors. Such techniques will be explained in Section 5.

4 Multi-Objective Particle Swarm Optimization

In this chapter, we incorporate several fitness inheritance and approximation techniques into a MOPSO that was previously proposed by us in [3] and updated in [13].

The PSO algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock [26]. In PSO, individuals, referred to as particles, are "flown" through a hyperdimensional search space. Changes to the position of the particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals.

A swarm consists of a set of particles, where each particle represents a potential solution. The position of each particle is changed according to its own experience and that of its neighbors. Let $\mathbf{x}_i(t)$ denote the position of particle i, at time step t. The position of particle i is then changed by adding a velocity $\mathbf{v}_i(t)$ to the current position, i.e.:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t) \tag{1}$$

The velocity vector drives the optimization process and reflects the socially exchanged information. In the global best version (used here) of PSO, each particle uses its history of experiences in terms of its own best solution thus far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*). Thus, the velocity vector changes in the following way:

$$\mathbf{v}_i(t) = W\mathbf{v}_i(t-1) + C_1 r_1(\mathbf{x}_{pbest_i} - \mathbf{x}_i(t-1)) + C_2 r_2(\mathbf{x}_{gbest_i} - \mathbf{x}_i(t-1)) \tag{2}$$

where W is the inertia weight, C_1 and C_2 are the learning factors (usually defined as constants), and $r_1, r_2 \in [0,1]$ are random values. In this work, we use $W = random(0.1, 0.5)$ and $C_1, C_2 = random(1.5, 2.0)$.

The MOPSO proposed in [3, 13] is based on Pareto dominance, since it considers every nondominated solution as a new leader.[1] Additionally, the approach uses a crowding factor [12] as a second discrimination criterion which is also adopted to filter out the list of available leaders. For each particle, we select the leader in the following way: 97% of the time a leader is randomly selected, if and only if that leader dominates the current particle, and, the remaining 3% of the time, we select a leader by means of a binary tournament based on the crowding value of the available set of leaders. If the size of the set of leaders is greater than the maximum allowable size, only the best leaders are retained based on their crowding value. We also proposed the use of different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. We proposed a scheme by which the swarm is subdivided in three parts (of equal size): the first sub-part has no mutation at all, the second sub-part

[1] A *leader* is used as the *gbest* solution in Equation 2.

uses uniform mutation and the third sub-part uses non-uniform mutation. The available set of leaders is the same for each of these sub-parts. Finally, the proposed approach also incorporates the ϵ-dominance concept [27] to fix the size of the set of final solutions produced by the algorithm. Figure 1 shows the pseudo-code of our proposed approach.

```
Begin
    Initialize swarm. Initialize leaders.
    Send leaders to ε-archive
    crowding(leaders), g = 0
    While g < gmax
        For each particle
            Select leader. Flight. Mutation.
⇒           If(p_i) Inherit Else Evaluation.
            Update pbest.
        EndFor
        Update leaders, Send leaders to ε-archive
        crowding(leaders), g++
    EndWhile
    Report results in ε-archive
End
```

Fig. 1. Pseudocode of our algorithm.

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

5 Proposed Techniques

5.1 Fitness Inheritance

Since PSO has no recombination operator, we adopted as "parents" of a particle the previous position of the particle, its *pbest* and its leader.

Linear Combination Based on Distances (LCBD)

We propose to calculate the new position in the objective space of a particle by means of a linear combination of the positions of the particles that were considered to calculate the new position in the search space. We consider the

position of the leader as the most important. Thus, the leader will be always considered.

Given a particle x_{old}, its personal best x_{pbest}, its assigned leader x_{ld} and the new particle x_{new}, we proceed to calculate the distance from x_{new} to its "parents" (as defined before): $d_1 = d(x_{new}, x_{old}), d_2 = d(x_{new}, x_{pbest}), d_3 = d(x_{new}, x_{ld})$, where d is an Euclidean distance. We propose variants of the same idea, based on the individuals that can be considered:

FI1 Previous position and leader: $r = \frac{d_1}{d_1+d_2}$,

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

FI2 *pbest* and leader. $r = \frac{d_2}{d_2+d_3}$,

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

FI3 Previous position, *pbest* and leader.
$r_1 = \frac{d_1}{d_1+d_2+d_3}$, $r_2 = \frac{d_2}{d_1+d_2+d_3}$, $r_3 = \frac{d_3}{d_1+d_2+d_3}$, $r_1 = 1/r_1, r_2 = 1/r_2, r_3 = 1/r_3$

$$f_i(x_{new}) = r_1 f_i(x_{old}) + r_2 f_i(x_{pbest}) + r_3 f_i(x_{ld}),$$

$i = 1, ..., n$. Where f_i is the value of the objective function i and n is the number of objective functions. See Figure 2 for an illustration of these techniques.

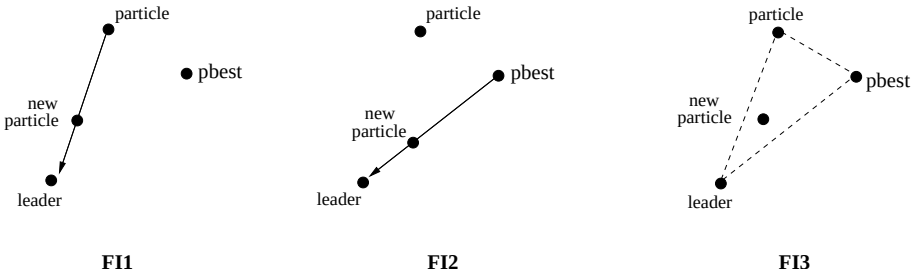

Fig. 2. Illustration of techniques FI1, FI2 and FI3.

The technique FI1 is the one proposed in [13]. As in [13], in all the inheritance techniques, if the leader selected does not dominate the current particle, we will proceed to calculate the inherited position and to assign the objective values of the closest leader to that position. This procedure is used to avoid the generation of invalid particles in the case of non-convex Pareto fronts. See Figure 3.

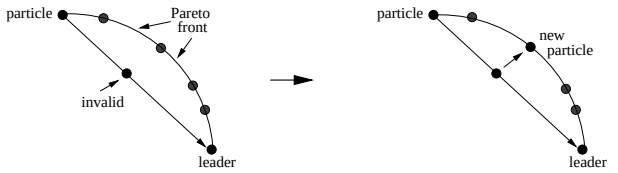

Fig. 3. Case in which "invalid" particles can be obtained and the method used to repair them.

Flight Formula on Objective Space (FFOS)

As we mentioned before, in PSO, the position of each particle in the search space is updated using the formula:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$$

$$\mathbf{v}_i(t) = W\mathbf{v}_i(t-1) + C_1 r_1(\mathbf{x}_{pbest_i} - \mathbf{x}_i(t-1)) + C_2 r_2(\mathbf{x}_{gbest_i} - \mathbf{x}_i(t-1))$$

In this case, we propose an analogous formula to update the position of each particle in objective function space:

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

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

where $\mathbf{f_i}$, $\mathbf{f_{pbest_i}}$ and $\mathbf{f_{gbest_i}}$ are the values of the objective function i for the current particle, its *pbest* and *gbest*, respectively. We adopt the same values of W, C_1, r_1, C_2 and r_2 previously used for the flight in the decision variable space. We will consider the following variants based on the vectors considered:

FI4 Considering the whole formula:

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

FI5 Ignoring the previous direction:

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

FI6 Ignoring the direction to the *pbest*:

$$\mathbf{vf_i}(t) = W\mathbf{vf_i}(t-1) + C_2 r_2(\mathbf{f_{gbest_i}} - \mathbf{f_i}(t-1))$$

Combination Using Flight Factors

Non-linear Combination (NLC)

In this case, we propose to calculate the new objective position of a particle using the elements of the flight formula:

$$\mathbf{f_i}(t) = W\mathbf{f_i}(t-1) + C_1r_1\mathbf{f_{pbest_i}} + C_2r_2\mathbf{f_{gbest_i}}$$

As in the previous cases, the variants considered are:

FI7 Considering the whole formula:

$$\mathbf{f_i}(t) = W\mathbf{f_i}(t-1) + C_1r_1\mathbf{f_{pbest_i}} + C_2r_2\mathbf{f_{gbest_i}}$$

FI8 Ignoring the previous position:

$$\mathbf{f_i}(t) = C_1r_1\mathbf{f_{pbest_i}} + C_2r_2\mathbf{f_{gbest_i}}$$

FI9 Ignoring the position of the *pbest*:

$$\mathbf{f_i}(t) = W\mathbf{f_i}(t-1) + C_2r_2\mathbf{f_{gbest_i}}$$

On the other hand, since $W \in (0.1, 0.5)$ and C_1r_1, $C_2r_2 \in (0.0, 2.0)$, we propose to modify the previous formula in the following way:

$$\mathbf{f_i}(t) = \frac{W}{0.5}\mathbf{f_i}(t-1) + \frac{C_1r_1}{2.0}\mathbf{f_{pbest_i}} + \frac{C_2r_2}{2.0}\mathbf{f_{gbest_i}}$$

As a result, we obtain the following variants:

FI10 Considering the whole formula:

$$\mathbf{f_i}(t) = \frac{W}{0.5}\mathbf{f_i}(t-1) + \frac{C_1r_1}{2.0}\mathbf{f_{pbest_i}} + \frac{C_2r_2}{2.0}\mathbf{f_{gbest_i}}$$

FI11 Ignoring the previous position:

$$\mathbf{f_i}(t) = \frac{C_1r_1}{2.0}\mathbf{f_{pbest_i}} + \frac{C_2r_2}{2.0}\mathbf{f_{gbest_i}}$$

FI12 Ignoring the position of the *pbest*:

$$\mathbf{f_i}(t) = \frac{W}{0.5}\mathbf{f_i}(t-1) + \frac{C_2r_2}{2.0}\mathbf{f_{gbest_i}}$$

Linear Combination (LC)

We propose to use the previous formula but in such a way that the result is a linear combination of the elements considered:

$$\mathbf{f_i}(t) = \frac{W}{r}\mathbf{f_i}(t-1) + \frac{C_1r_1}{r}\mathbf{f_{pbest_i}} + \frac{C_2r_2}{r}\mathbf{f_{gbest_i}}$$

where $r = W + C_1r_1 + C_2r_2$. The corresponding variants are the following (note the changes in r):

FI13 Considering the whole formula, $r = W + C_1 r_1 + C_2 r_2$:

$$\mathbf{f_i}(t) = \frac{W}{r}\mathbf{f_i}(t-1) + \frac{C_1 r_1}{r}\mathbf{f_{pbest_i}} + \frac{C_2 r_2}{r}\mathbf{f_{gbest_i}}$$

FI14 Ignoring the previous position, $r = C_1 r_1 + C_2 r_2$:

$$\mathbf{f_i}(t) = \frac{C_1 r_1}{r}\mathbf{f_{pbest_i}} + \frac{C_2 r_2}{r}\mathbf{f_{gbest_i}}$$

FI15 Ignoring the position of the *pbest*, $r = W + C_2 r_2$:

$$\mathbf{f_i}(t) = \frac{W}{r}\mathbf{f_i}(t-1) + \frac{C_2 r_2}{r}\mathbf{f_{gbest_i}}$$

5.2 Fitness Approximation (FA)

As we could see in the previous section, fitness inheritance techniques assign the fitness value (objective values, in our case) of an individual using the fitness values of its parents. However, in the case of fitness approximation techniques, it is possible to use any set of neighboring particles to estimate the fitness of a particle, based on an approximate model of the fitness landscape. We propose four simple approximation techniques. In each case, the particle will take the objective values of the particle indicated:

FA1 The closest particle. Since we have the available set of leaders stored in an external list (archive), in this case we will search for the closest leader, either member of the swarm itself or member of the available set of leaders.
FA2 The closest leader. In this case, we will search for the closest particle member of the available set of leaders.
FA3 The closest particle member of the swarm. In this case, we will not consider the available set of leaders.
FA4 The average of the 10 closest particles. In this case, we will consider both the particles members of the swarm and the particles members of the available set of leaders.

We use the Euclidean distance in the decision variable space. In technique FA4, there are cases in which an invalid particle may be created. In this way, if among the 10 closest particles there are two or more leaders, or there is just one leader but this leader does not dominate the current particle, we will proceed as it was explained before. See Figure 3.

6 Comparison of Results

In order to compare the proposed techniques, we performed a study using five well-known test functions taken from the specialized literature on evolutionary multi-objective optimization:

- Test Function ZDT1 [9]:

$$\begin{aligned} \text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ & f_1(\mathbf{x}) = x_1 \\ & f_2(\mathbf{x}) = g(\mathbf{x})h(f_1, g) \\ & g(\mathbf{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1), h(f1, g) = 1 - \sqrt{f_1/g} \end{aligned} \tag{3}$$

  where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT2 [9]:

$$\begin{aligned} \text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ & f_1(\mathbf{x}) = x_1 \\ & f_2(\mathbf{x}) = g(\mathbf{x})h(f_1, g) \\ & g(\mathbf{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1), h(f1, g) = 1 - (f_1/g)^2 \end{aligned} \tag{4}$$

  where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT3 [9]:

$$\begin{aligned} \text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ & f_1(\mathbf{x}) = x_1 \\ & f_2(\mathbf{x}) = g(\mathbf{x})h(f_1, g) \\ g(\mathbf{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1) \ , \ & h(f1, g) = 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1) \end{aligned} \tag{5}$$

  where $m = 30$, and $x_i \in$[0,1].

- Test Function ZDT4 [9]:

$$\begin{aligned} \text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ & f_1(\mathbf{x}) = x_1 \\ & f_2(\mathbf{x}) = g(\mathbf{x})h(f_1, g) \\ g(\mathbf{x}) = 1 + 10(m-1) + & \sum_{i=2}^{m}(x_i^2 - 10cos(4\pi x_i)), h(f1, g) = 1 - \sqrt{f_1/g} \end{aligned} \tag{6}$$

  where $m = 10$, $x_1 \in$[0,1] and $x_i \in$[-5,5], $i = 2, ..., m$.

- Test Function DTLZ2 [28]:

$$\begin{aligned} \text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})) \\ f_1(\mathbf{x}) &= (1+g(\mathbf{x}))cos(x_1\pi/2)cos(x_2\pi/2) \\ f_2(\mathbf{x}) &= (1+g(\mathbf{x}))cos(x_1\pi/2)sin(x_2\pi/2) \\ f_3(\mathbf{x}) &= (1+g(\mathbf{x}))sin(x_1\pi/2) \\ g(\mathbf{x}) &= \sum_{i=3}^{m}(x_i-0.5)^2 \end{aligned} \tag{7}$$

  where $m = 12$ and $x_i \in$[0,1].

Functions ZDT1 and ZDT4 have convex Pareto fronts, ZDT2 and DTLZ2 have non-convex Pareto fronts and ZDT3 has a non-convex and discontinuous Pareto front.

We performed experiments with different values of *inheritance (approximation) proportion* p_i. We experimented with: p_i= 0.1, 0.2, 0.3, 0.4. Note that this proportion of individuals indicates also the percentage by which the number of evaluations is reduced (e.g., $p_i = 0.1$ means that 10% less evaluations are performed). We performed 20 runs for each function and each technique. The parameters adopted for our MOPSO were: 100 particles, 200 generations and 100 particles in the external archive.

Next, we show the results corresponding to the following measure:

Success Counting (SCC): We define this measure based on the idea of the measure called *Error Ratio* proposed by Van Veldhuizen [29] which indicates the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set. In this case, we count the number of vectors (in the current set of nondominated vectors available) that are members of the Pareto optimal set:

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

where n is the number of vectors in the current set of nondominated vectors available; $s_i = 1$ if vector i is a member of the Pareto optimal set, and $s_i = 0$ otherwise. It should then be clear that $SCC = n$ indicates an ideal behavior, since it would mean that all the vectors generated by our algorithm belong to the true Pareto optimal set of the problem. Note that SCC avoids the bias introduced by the Error Ratio measure, which normalizes the number of solutions found (which belong to the true Pareto front) and, therefore, provides only a percentage of solutions that reached the true Pareto front. This percentage does not provide any idea regarding the actual number of nondominated solutions that each algorithm produced.

Tables 1, 2, 3, 4, 5 and 6 present a summary of the results obtained. In each case, we present the average of the SCC measure over the 20 runs, and the percentage of decrement or increment on the quality of the results. Also, we present the average of the percentages for each value of inheritance proportion, for each technique.

Table 1. Obtained results for different values of inheritance proportion, for techniques FI1, FI2 and FI3.

FI1	Inheritance proportion p_i							
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	77	(+8.5%)	64	(-9.9%)	62	(-12.7%)	61	(-14.1%)
ZDT2	89	83	(-6.7%)	86	(-3.4%)	79	(-11.2%)	77	(-13.5%)
ZDT3	68	73	(+7.4%)	65	(-4.4%)	64	(-5.9%)	59	(-13.2%)
ZDT4	80	81	(+1.3%)	81	(+1.3%)	60	(-25.0%)	68	(-15.0%)
DTLZ2	18	15	(-16.7%)	16	(-11.1%)	12	(-33.3%)	12	(-33.3%)
Average			**-1.2%**		**-5.5 %**		**-17.6 %**		**-17.8 %**
FI2	Inheritance proportion p_i							
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	68	(-4.2%)	68	(-4.2%)	59	(-16.9%)
ZDT2	89	81	(-9.0%)	82	(-7.9%)	78	(-12.4%)	77	(-13.5%)
ZDT3	68	64	(-5.9%)	67	(-1.5%)	58	(-14.7%)	63	(-7.4%)
ZDT4	80	77	(-3.8%)	83	(+3.8%)	67	(-16.3%)	69	(-13.8%)
DTLZ2	18	15	(-16.7%)	18	(0.0%)	15	(-16.7%)	14	(-22.2%)
Average			**-6.2%**		**-2.0 %**		**-12.9 %**		**-14.8 %**
FI3	Inheritance proportion p_i							
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	73	(+2.8%)	69	(-2.8%)	69	(-2.8%)	50	(-29.6%)
ZDT2	89	87	(-2.2%)	82	(-7.9%)	71	(-20.2%)	76	(-14.6%)
ZDT3	68	67	(-1.5%)	63	(-7.4%)	64	(-5.9%)	60	(-11.8%)
ZDT4	80	81	(+1.3%)	79	(-1.3%)	59	(-26.3%)	68	(-15.0%)
DTLZ2	18	17	(-5.6%)	18	(0.0%)	14	(-22.2%)	9	(-50.0%)
Average			**-1.0%**		**-3.9 %**		**-15.5 %**		**-24.2%**

7 Discussion of Results

Since comparing 19 different techniques is very difficult, we decided to represent each technique with a vector. The vector used is that containing the average of the change in the quality of results for each inheritance proportion value. For example, to represent technique FI1, we construct the following vector (see Table 1):

Inheritance proportion p_i	0.1	0.2	0.3	0.4
Average vector	-1.2	-5.5	-17.6	-17.8

Table 2. Obtained results for different values of inheritance proportion, for techniques FI4, FI5 and FI6.

FI4	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	62	(-12.7%)	62	(-12.7%)	59	(-16.9%)	49	(-31.0%)
ZDT2	89	85	(-4.5%)	84	(-5.6%)	78	(-12.4%)	79	(-11.2%)
ZDT3	68	73	(+7.4%)	69	(+1.5%)	60	(-11.8%)	58	(-14.7%)
ZDT4	80	88	(+10.0%)	88	(+10.0%)	85	(+6.3%)	82	(+2.5%)
DTLZ2	18	18	(0.0%)	11	(-38.9%)	11	(-38.9%)	12	(-33.3%)
Average			**0.0%**		**-9.1 %**		**-14.7 %**		**-17.5%**
FI5	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	69	(-2.8%)	61	(-14.1%)	56	(-21.1%)
ZDT2	89	89	(0.0%)	79	(-11.2%)	84	(-5.6%)	77	(-13.5%)
ZDT3	68	72	(+5.9%)	70	(+2.9%)	55	(-19.1%)	58	(-14.7%)
ZDT4	80	87	(+8.8%)	85	(+6.3%)	85	(+6.3%)	82	(+2.5%)
DTLZ2	18	18	(0.0%)	12	(-33.3%)	13	(-27.8%)	12	(-33.3%)
Average			**+3.8%**		**-7.6 %**		**-12.1 %**		**-16.1%**
FI6	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	70	(-1.4%)	61	(-14.1%)	62	(-12.7%)	47	(-33.8%)
ZDT2	89	83	(-6.7%)	82	(-7.9%)	76	(-14.6%)	70	(-21.3%)
ZDT3	68	72	(+5.9%)	72	(+5.9%)	59	(-13.2%)	61	(-10.3%)
ZDT4	80	83	(+3.8%)	84	(+5.0%)	80	(0.0%)	79	(-1.3%)
DTLZ2	18	16	(-11.1%)	14	(-22.2%)	14	(-22.2%)	11	(-38.9%)
Average			**-1.9%**		**-6.7 %**		**-12.5 %**		**-21.1%**

In this way, in Table 7 we present the vectors of all techniques. Since every entry in each vector is a change in the quality of the obtained results given a value of inheritance proportion, the bigger the values of the vector, the better the corresponding technique is. Thus, we are interested on the vector or vectors that represent the solution to the problem of maximizing all the entries (i.e., each entry is considered as an objective).

The nondominated vectors among all the 19 techniques are the vectors corresponding to techniques FI2, FI3, FI5, FI9, FI11, FI14, FA1, FA3 and FA4. That is, these nine techniques are the best overall performers. For this reason, all these techniques are marked with a level of 1 in Table 7. As we can see, among the best techniques, 6 are inheritance techniques and 3 are approximation techniques. In fact, it is very interesting to note that four of the six best inheritance techniques ignore the previous position of the particle, in order to update the position in objective function space. On the other hand, the only approximation technique that doesn't figure as one of the best is the one that only considers the set of leaders to assign the objective function values of a particle.

Table 3. Obtained results for different values of inheritance proportion, for techniques FI7, FI8 and FI9.

FI7	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	64	(-9.9%)	58	(-18.3%)	57	(-19.7%)	47	(-33.8%)
ZDT2	89	83	(-6.7%)	74	(-16.9%)	68	(-23.6%)	66	(-25.8%)
ZDT3	68	66	(-2.9%)	69	(+1.5%)	64	(-5.9%)	57	(-16.2%)
ZDT4	80	80	(0.0%)	74	(-7.5%)	57	(-28.8%)	44	(-45.0%)
DTLZ2	18	13	(-27.8%)	14	(-22.2%)	13	(-27.8%)	9	(-50.0%)
Average			**-9.5%**		**-12.7 %**		**-21.2 %**		**-34.2%**
FI8	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	62	(-12.7%)	53	(-25.4%)	47	(-33.8%)
ZDT2	89	85	(-4.5%)	84	(-5.6%)	66	(-25.8%)	65	(-27.0%)
ZDT3	68	71	(+4.4%)	67	(-1.5%)	61	(-10.3%)	52	(-23.5%)
ZDT4	80	80	(0.0%)	72	(-10.0%)	63	(-21.3%)	52	(-35.0%)
DTLZ2	18	16	(-11.1%)	14	(-22.2%)	13	(-27.8%)	12	(-33.3%)
Average			**-2.8%**		**-10.4 %**		**-22.1 %**		**-30.5%**
FI9	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	67	(-5.6%)	58	(-18.3%)	54	(-23.9%)	44	(-38.0%)
ZDT2	89	90	(+1.1%)	85	(-4.5%)	69	(-22.5%)	68	(-23.6%)
ZDT3	68	68	(0.0%)	67	(-1.5%)	61	(-10.3%)	51	(-25.0%)
ZDT4	80	83	(+3.8%)	76	(-5.0%)	72	(-10.0%)	58	(-27.5%)
DTLZ2	18	19	(+5.6%)	18	(0.0%)	19	(+5.6%)	18	(0.0%)
Average			**+1.0%**		**-5.9 %**		**-12.2 %**		**-22.8%**

8 Comparison with other PSO approaches

In the previous section, we found nine techniques to be the best from the set proposed. In this section, some of those nine techniques will be compared against two other PSO-based multi-objective approaches representative of the state-of-the-art: the Sigma-MOPSO [30] and the Cluster-MOPSO [31].

For our comparison, we chose only five from the nine best techniques to be compared. With this aim, we calculated the norm (distance to the origin) of the vector of each technique and we selected the five vectors with the lowest values. In this way, we selected the techniques from the *knee* of the Pareto front, that is, the compromise solutions from the central portion of the front. In Table 7, we show the norm values of the nine best techniques and their corresponding rank according to this value. As we can see, the five techniques with the lowest value of the norm are: FI2, FI5, FI11, FA1 and FA3.

For this comparative analysis, we will use the two following test functions:

- Test Function DTLZ4 [28]:

$$\text{Minimize} \quad (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})) \tag{8}$$

Table 4. Obtained results for different values of inheritance proportion, for techniques FI10, FI11 and FI12.

FI10	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	71	(0.0%)	58	(-18.3%)	59	(-16.9%)	48	(-32.4%)
ZDT2	89	78	(-12.4%)	78	(-12.4%)	69	(-22.5%)	58	(-34.8%)
ZDT3	68	70	(+2.9%)	63	(-7.4%)	61	(-10.3%)	47	(-30.9%)
ZDT4	80	78	(-2.5%)	81	(+1.3%)	58	(-27.5%)	52	(-35.0%)
DTLZ2	18	17	(-5.6%)	13	(-27.8%)	11	(-38.9%)	11	(-38.9%)
Average			**-3.5%**		**-12.9 %**		**-23.2 %**		**-34.4%**
FI11	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	62	(-12.7%)	63	(-11.3%)	55	(-22.5%)	37	(-48.0%)
ZDT2	89	84	(-5.6%)	87	(-2.2%)	81	(-9.0%)	76	(-14.6%)
ZDT3	68	69	(+1.5%)	60	(-11.8%)	57	(-16.2%)	44	(-35.3%)
ZDT4	80	82	(+2.5%)	81	(+1.3%)	73	(-8.8%)	73	(-8.8%)
DTLZ2	18	17	(-5.6%)	21	(+16.7%)	23	(+27.8%)	23	(+27.8%)
Average			**-4.0%**		**-1.5 %**		**-5.7 %**		**-15.8%**
FI12	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	66	(-7.0%)	56	(-21.1%)	55	(-22.5%)	48	(-32.4%)
ZDT2	89	87	(-2.2%)	85	(-4.5%)	74	(-16.9%)	80	(-10.1%)
ZDT3	68	66	(-2.9%)	64	(-5.9%)	55	(-19.1%)	53	(-22.1%)
ZDT4	80	80	(0.0%)	75	(-6.3%)	71	(-11.3%)	61	(-23.8%)
DTLZ2	18	18	(0.0%)	18	(0.0%)	16	(-11.1%)	16	(-11.1%)
Average			**-2.4%**		**-7.6 %**		**-16.2 %**		**-19.9%**

$$
\begin{aligned}
f_1(\mathbf{x}) &= (1+g(\mathbf{x}))cos(x_1^{\alpha}\pi/2)cos(x_2^{\alpha}\pi/2) \\
f_2(\mathbf{x}) &= (1+g(\mathbf{x}))cos(x_1^{\alpha}\pi/2)sin(x_2^{\alpha}\pi/2) \\
f_3(\mathbf{x}) &= (1+g(\mathbf{x}))sin(x_1^{\alpha}\pi/2) \\
g(\mathbf{x}) &= \sum_{i=3}^{m}(x_i-0.5)^2
\end{aligned}
$$

where α=100, $m = 12$ and $x_i \in$[0,1].

- Test Function DTLZ6 [28]:

$$
\begin{aligned}
\text{Minimize} \quad & (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})) \qquad (9) \\
& f_1(\mathbf{x}) = x_1 \\
& f_2(\mathbf{x}) = x_2 \\
& f_3(\mathbf{x}) = (1+g(\mathbf{x}))h(f_1, f_2, g) \\
g(\mathbf{x}) = 1 + 9/(m-2)\sum_{i=3}^{m} x_i \ , \ & h(f_1, f_2, g) = 3 - \sum_{i=1}^{2}[\frac{f_i}{1+g}(1+sin(3\pi f_i))]
\end{aligned}
$$

Table 5. Obtained results for different values of inheritance proportion, for techniques FI13, FI14 and FI15.

FI13	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	68	(-4.2%)	69	(-2.8%)	63	(-11.3%)	58	(-18.3%)
ZDT2	89	84	(-5.6%)	81	(-9.0%)	79	(-11.2%)	80	(-10.1%)
ZDT3	68	70	(+2.9%)	68	(0.0%)	63	(-7.4%)	54	(-20.6%)
ZDT4	80	79	(-1.3%)	81	(+1.3%)	64	(-20.0%)	59	(-26.3%)
DTLZ2	18	14	(-22.2%)	16	(-11.1%)	14	(-22.2%)	12	(-33.3%)
Average			**-6.1%**		**-4.3 %**		**-14.4 %**		**-21.7%**
FI14	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	75	(+5.6%)	66	(-7.0%)	58	(-18.3%)	59	(-16.9%)
ZDT2	89	88	(-1.1%)	79	(-11.2%)	83	(-6.7%)	72	(-19.1%)
ZDT3	68	74	(+8.8%)	69	(+1.5%)	63	(-7.4%)	60	(-11.8%)
ZDT4	80	81	(+1.3%)	79	(-1.3%)	73	(-8.8%)	67	(-16.3%)
DTLZ2	18	16	(-11.1%)	15	(-16.7%)	13	(-27.8%)	13	(-27.8%)
Average			**+0.7%**		**-6.9 %**		**-13.8 %**		**-18.4%**
FI15	Inheritance proportion p_i								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	63	(-11.3%)	69	(-2.8%)	56	(-21.1%)
ZDT2	89	86	(-3.4%)	81	(-9.0%)	72	(-19.1%)	73	(-18.0%)
ZDT3	68	72	(+5.9%)	70	(+2.9%)	64	(-5.9%)	58	(-14.7%)
ZDT4	80	81	(+1.3%)	78	(-2.5%)	63	(-21.3%)	70	(-12.5%)
DTLZ2	18	13	(-27.8%)	15	(-16.7%)	16	(-11.1%)	10	(-44.4%)
Average			**-5.4%**		**-7.3 %**		**-12.0 %**		**-22.1%**

where $m = 22$ and $x_i \in [0,1]$.

As in previous experiments, we used different values of p_i. We performed 20 runs for each function and each approach. The approaches without fitness inheritance or approximation performed 20000 objective function evaluations. The parameters adopted for our MOPSO were the same as before. Cluster-MOPSO used 40 particles, 4 swarms, 5 iterations per swarm and a total number of iterations of 100. In the case of Sigma-MOPSO, 200 particles were used through 100 iterations (these values were suggested by the author of the method). The PSO approaches will be identified with the following labels: sMOPSO refers to [30], cMOPSO refers to [31], and oMOPSO is our MOPSO. All the algorithms were set such that they provided Pareto fronts with 100 points. In this case, we also show the obtained results with respect to the following measure:

Inverted Generational Distance (IGD): The concept of generational distance was introduced by Van Veldhuizen & Lamont [32, 33] as a way of

Table 6. Obtained results for different values of approximation proportion, for techniques FA1, FA2, FA3 and FA4.

FA1	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	74	(+4.2%)	64	(-9.9%)	63	(-11.3%)	55	(-22.5%)
ZDT2	89	88	(-1.1%)	85	(-4.5%)	81	(-9.0%)	76	(-14.6%)
ZDT3	68	73	(+7.4%)	61	(-10.3%)	60	(-11.8%)	55	(-19.1%)
ZDT4	80	85	(+6.3%)	89	(+11.3%)	79	(-1.3%)	80	(0.0%)
DTLZ2	18	18	(0.0%)	13	(-27.8%)	14	(-22.2%)	12	(-33.3%)
Average			**+3.4%**		**-8.2 %**		**-11.1 %**		**-17.9%**
FA2	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	75	(+5.6%)	57	(-19.7%)	54	(-23.9%)	46	(-35.2%)
ZDT2	89	83	(-6.7%)	72	(-19.1%)	63	(-29.2%)	76	(-14.6%)
ZDT3	68	63	(-7.4%)	58	(-14.7%)	58	(-14.7%)	56	(-17.6%)
ZDT4	80	86	(+7.5%)	87	(+8.8%)	81	(+1.3%)	83	(+3.8%)
DTLZ2	18	15	(-16.7%)	13	(-27.8%)	11	(-38.9%)	10	(-44.4%)
Average			**-3.5%**		**-14.5 %**		**-21.1 %**		**-21.6%**
FA3	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	71	(0.0%)	67	(-5.6%)	63	(-11.3%)	50	(-29.6%)
ZDT2	89	88	(-1.1%)	87	(-2.2%)	85	(-4.5%)	76	(-14.6%)
ZDT3	68	65	(-4.4%)	65	(-4.4%)	55	(-19.1%)	57	(-16.2%)
ZDT4	80	89	(+11.3%)	91	(+13.8%)	86	(+7.5%)	87	(+8.8%)
DTLZ2	18	16	(-11.1%)	15	(-16.7%)	16	(-11.1%)	13	(-27.8%)
Average			**-1.1%**		**-3.0 %**		**-7.7 %**		**-15.9%**
FA4	Approximation proportion p_a								
function	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
ZDT1	71	69	(-2.8%)	59	(-16.9%)	60	(-15.5%)	52	(-26.8%)
ZDT2	89	87	(-2.2%)	80	(-10.1%)	76	(-14.6%)	71	(-20.2%)
ZDT3	68	67	(-1.5%)	71	(+4.4%)	56	(-17.6%)	56	(-17.6%)
ZDT4	80	86	(+7.5%)	85	(+6.3%)	79	(-1.3%)	80	(0.0%)
DTLZ2	18	20	(+11.1%)	16	(-11.1%)	14	(-22.2%)	12	(-33.3%)
Average			**+2.4%**		**-5.5 %**		**-14.2 %**		**-19.6%**

estimating how far are the elements in the Pareto front produced by our algorithm from those in the true Pareto front of the problem. This measure is defined as:

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

where n is the number of nondominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the true Pareto front. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem. Therefore, any other value will in-

Table 7. Vectors of change in quality for each technique, for each value of inheritance or approximation proportion.

Group		0.1	0.2	0.3	0.4	level	norm	rank
LCBD	FI1	-1.2	-5.5	-17.6	-17.8			
	FI2	-6.2	-2.0	-12.9	-14.8	**1**	20.69	**3**
	FI3	-1.0	-3.9	-15.5	-24.2	**1**	29.62	9
FFOS	FI4	0.0	-9.1	-14.7	-17.5			
	FI5	3.8	-7.6	-12.1	-16.1	**1**	21.86	**4**
	FI6	-1.9	-6.7	-12.5	-21.1			
NLC	FI7	-9.5	-12.7	-21.2	-34.2			
	FI8	-2.8	-10.4	-22.1	-30.5			
	FI9	1.0	-5.9	-12.2	-22.8	**1**	26.54	8
	FI10	-3.5	-12.9	-23.2	-34.4			
	FI11	-4.0	-1.5	-5.7	-15.8	**1**	17.33	**1**
	FI12	-2.4	-7.6	-16.2	-19.9			
LC	FI13	-6.1	-4.3	-14.4	-21.7			
	FI14	0.7	-6.9	-13.8	-18.4	**1**	24.02	6
	FI15	-5.4	-7.3	-12.0	-22.1			
FA	FA1	3.4	-8.2	-11.1	-17.9	**1**	22.86	**5**
	FA2	-3.5	-14.5	-21.1	-21.6			
	FA3	-1.1	-3.0	-7.7	-15.9	**1**	17.95	**2**
	FA4	2.4	-5.5	-14.2	-19.6	**1**	24.94	7

dicate how "far" we are from the global Pareto front of our problem. In our case, we implemented an "inverted" generational distance measure (IGD) in which we use as a reference the true Pareto front, and we compare each of its elements with respect to the front produced by an algorithm. In this way, we are calculating how far are the elements of the true Pareto front, from those in the Pareto front produced by our algorithm. Computing this "inverted" generational distance value reduces the bias that can arise when an algorithm didn't fully cover the true Pareto front.

Tables 8 and 9 present a summary of the results obtained. In each case, we present the average and standard deviation of the Success Counting (SCC) and Inverted Generational Distance (IGD) measures over the 20 runs.

As we can see in Table 8, in function DTLZ4 our approach (oMOPSO) is outperformed by one of the other PSO-based approaches (sMOPSO). On the other hand, in Table 9 we can see that in function DTLZ6 our approach is clearly the best.

Table 8 shows that, in function DTLZ4, all the techniques have a very good performance with respect to the IGD measure, with technique FA1 being (marginally) the best (considering the standard deviation values). On the other hand, although with respect to the average values of the SCC measure, technique FI11 is the best, this technique has the highest standard deviations. Thus, we conclude that the best technique is FA1 also in this case. In fact,

Table 8. Obtained results for the test function DTLZ4, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with techniques FI2, FI5, FI11, FA1 and FA3 incorporated (p_i=0.1,0.2,0.3,0.4).

FI2		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	39	1.4	11	10	9	10	12
	std. dev.	14.8	3.3	4.4	4.7	3.0	3.8	8.1
IGD	mean	0.0064	0.0223	0.0106	0.0121	0.0129	0.0111	0.0107
	std. dev.	0.0007	0.0074	0.0034	0.0052	0.0037	0.0043	0.0050
FI5		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	39	1.4	11	10	10	9	11
	std. dev.	14.8	3.3	4.4	6.8	4.9	4.7	6.9
IGD	mean	0.0064	0.0223	0.0106	0.0100	0.0119	0.0109	0.0108
	std. dev.	0.0007	0.0074	0.0034	0.0045	0.0050	0.0050	0.0046
FI11		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	39	1.4	11	11	13	14	12
	std. dev.	14.8	3.3	4.4	4.5	11.5	14.8	16.2
IGD	mean	0.0064	0.0223	0.0106	0.0107	0.0105	0.0111	0.0119
	std. dev.	0.0007	0.0074	0.0034	0.0047	0.0046	0.0059	0.0065
FA1		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	39	1.4	11	**13**	**10**	**12**	**8**
	std. dev.	14.8	3.3	4.4	**6.9**	**3.4**	**6.5**	**4.2**
IGD	mean	0.0064	0.0223	0.0106	**0.0093**	**0.0112**	**0.0102**	**0.0122**
	std. dev.	0.0007	0.0074	0.0034	**0.0045**	**0.0039**	**0.0042**	**0.0044**
FA3		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	39	1.4	11	11	8	9	7
	std. dev.	14.8	3.3	4.4	3.2	3.1	3.7	3.8
IGD	mean	0.0064	0.0223	0.0106	0.0120	0.0138	0.0113	0.0129
	std. dev.	0.0007	0.0074	0.0034	0.0033	0.0040	0.0040	0.0042

both approximation techniques have the lowest standard deviations, in both performance measures.

From Table 9 we can conclude that, as in function DTLZ4, in function DTLZ6 all the techniques have a very good performance with respect to the IGD measure. However, in this case technique FI11 is the best. Also, with respect to the SCC measure technique FI5 is the best, considering all cases.

In general, fitness inheritance techniques seem to have a better performance in function DTLZ6 and fitness approximation techniques seem to have a better performance in function DTLZ4. These results agree with the results obtained before. As we can see in Table 10, the results obtained in the previous study show that technique FA3 was the best in function ZDT4. On the other hand, in function DTLZ2 technique FA1 is one of the two technique with the best results (the other is FI11). In fact, functions ZDT4 and DTLZ2 have the lowest number of variables. Function ZDT4 has 10 variables and function DTLZ2 has 12 variables, while functions ZDT1, ZDT2 and ZDT3

Table 9. Obtained results for the test function DTLZ6, for sMOPSO, cMOPSO, oMOPSO, and oMOPSO with techniques FI2, FI5, FI11, FA1 and FA3 incorporated (p_i=0.1,0.2,0.3,0.4).

FI2		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	62	60	53	50	47
	std. dev.	0.0	0.0	13.0	21.4	20.9	22.0	17.3
IGD	mean	0.0673	0.0373	0.0091	0.0082	0.0092	0.0101	0.0111
	std. dev.	0.0000	0.0172	0.0058	0.0060	0.0062	0.0065	0.0062
FI5		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	62	**61**	**60**	**61**	**43**
	std. dev.	0.0	0.0	13.0	**17.3**	**21.7**	**17.2**	**20.7**
IGD	mean	0.0673	0.0373	0.0091	0.0074	0.0089	0.0087	0.0101
	std. dev.	0.0000	0.0172	0.0058	0.0060	0.0060	0.0058	0.0058
FI11		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	62	54	62	54	50
	std. dev.	0.0	0.0	13.0	23.4	20.9	26.8	20.6
IGD	mean	0.0673	0.0373	0.0091	**0.0090**	**0.0064**	**0.0057**	**0.0058**
	std. dev.	0.0000	0.0172	0.0058	**0.0058**	**0.0052**	**0.0053**	**0.0052**
FA1		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	62	54	56	49	53
	std. dev.	0.0	0.0	13.0	10.7	20.0	21.6	21.5
IGD	mean	0.0673	0.0373	0.0091	0.0123	0.0089	0.0108	0.0082
	std. dev.	0.0000	0.0172	0.0058	0.0045	0.0062	0.0060	0.0068
FA3		sMOPSO	cMOPSO	oMOPSO	0.1	0.2	0.3	0.4
SCC	mean	1	0	62	62	51	54	53
	std. dev.	0.0	0.0	13.0	15.0	27.0	20.0	20
IGD	mean	0.0673	0.0373	0.0091	0.0109	0.0099	0.0100	0.0116
	std. dev.	0.0000	0.0172	0.0058	0.0056	0.0059	0.0061	0.0056

Table 10. Obtained results for test functions ZDT4 and DTLZ2, for oMOPSO with techniques FI2, FI5, FI11, FA1 and FA3 incorporated (p_i=0.1,0.2,0.3,0.4).

ZDT4	Inheritance proportion p_i								
technique	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
FI2	80	77	(-3.8%)	83	(+3.8%)	67	(-16.3%)	69	(-13.8%)
FI5	80	87	(+8.8%)	85	(+6.3%)	85	(+6.3%)	82	(+2.5%)
FI11	80	82	(+2.5%)	81	(+1.3%)	73	(-8.8%)	73	(-8.8%)
FA1	80	85	(+6.3%)	89	(+11.3%)	79	(-1.3%)	80	(0.0%)
FA3	80	89	**(+11.3%)**	91	**(+13.8%)**	86	**(+7.5%)**	87	**(+8.8%)**

DTLZ2	Inheritance proportion p_i								
technique	0.0	0.1 (-10%)		0.2 (-20%)		0.3 (-30%)		0.4 (-40%)	
FI2	18	15	(-16.7%)	18	(0.0%)	15	(-16.7%)	14	(-22.2%)
FI5	18	18	(0.0%)	12	(-33.3%)	13	(-27.8%)	12	(-33.3%)
FI11	18	17	**(-5.6%)**	21	**(+16.7%)**	23	**(+27.8%)**	23	**(+27.8%)**
FA1	18	18	**(0.0%)**	13	**(-27.8%)**	14	**(-22.2%)**	12	**(-33.3%)**
FA3	18	16	(-11.1%)	15	(-16.7%)	16	(-11.1%)	13	(-27.8%)

have 30 variables each, and DTLZ6 has 22 variables. In this way, we can conclude that, in general, fitness approximation techniques have better results when the test function has a low dimensional decision space and that fitness inheritance techniques have better results when the test function has a high dimensional decision space. This conclusion seems to agree with the results obtained in some of our previous work [13].

Finally, we should observe the results of technique FI11. In function DTLZ2, this technique obtained almost the best results. Also, in function DTLZ4, technique FI11 obtained the best average results. However, in the case of function DTLZ4, we could see that technique FI11 had the highest standard deviation values. In this way, technique FI11 is the only inheritance technique that had a very good performance in the test functions with a low number of variables, specifically from the test suite DTLZ. This denotes the importance of considering test functions from different test suites and with different characteristics. Technique FI11 almost always improved the results of the approach without inheritance even when a 40% of the number of evaluation was saved, in functions DTLZ2 and DTLZ4. Certainly, technique FI11 deserves further study, and this is already one of the priorities of our future work.

9 Conclusions

We proposed several fitness inheritance and approximation techniques and incorporated them into a Multi-Objective Particle Swarm Optimizer previously proposed by the authors. We studied the proposed techniques using several standard test functions taken from the multi-objective optimization literature.

From the nineteen techniques proposed, nine were found to be the best. Six of those nine techniques were inheritance techniques and the other three were approximation techniques. From the six best inheritance techniques, four techniques don't consider the previous position of a particle in order to compute the new objective position. On the other hand, the only approximation technique that didn't appear as one of the best was the one that only considers the set of leaders to assign the objective values of a particle. The results obtained indicate that the members of the swarm must be always considered in order to estimate the fitness value of a particle. Also, the importance of the *pbest* particle constitutes a topic that deserves further analysis, since it doesn't provide any useful information when the fitness value of a particle is being inherited.

Five of the nine best techniques found were selected to be tested on other functions and compared with respect to other PSO-based multi-objective algorithms. The obtained results show that the five enhancement techniques have a good performance and are very promising. In general, fitness inheritance techniques seem to be more appropriate for high-dimensional decision

space problems and fitness approximation techniques seem more appropriate for low-dimensional decision space problems. Only one of the inheritance techniques had a very good performance when applied to functions with low number of variables, from a specified test suite. Such inheritance technique almost always improved the results of the approach without inheritance and it certainly deserves further study.

As part of our future work, we plan to improve the enhancement techniques that were found to be the best in this study, in order to minimize the decrement in quality of results while obtaining major savings in the number of evaluations performed.

Acknowledgments

The first author acknowledges CONACyT for granting her a scholarship to pursue graduate studies at the computer science section of CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT project number 45683.

References

1. Robert E. Smith, B. A. Dike, and S. A. Stegmann. Fitness Inheritance in Genetic Algorithms. In *SAC '95*, pages 345–350. ACM Press, 1995.
2. Yaochu Jin and Bernhard Sendhoff. Fitness Approximation in Evolutionary Computation: A Survey. In *GECCO 2002*, pages 1105–1112. Morgan Kaufmann, 2002.
3. Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-based Multi-objective Optimization using Crowding, Mutation and ϵ-Dominance. In *EMO 2005*, pages 505–519. Springer-Verlag, LNCS 3410, 2005.
4. Xiaowei Zheng, Bryant A. Julstrom, and Weidong Cheng. Design of Vector Quantization Codebooks Using a Genetic Algorithm. In *IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 525–529. IEEE Press, 1997.
5. Kumara Sastry, David E. Goldberg, and Martin Pelikan. Don't Evaluate, Inherit. In *GECCO 2001*, pages 551–558. Morgan Kaufmann, 7-11 2001.
6. Jian-Jung Chen, David E. Goldberg, Shinn-Ying Ho, and Kumara Sastry. Fitness Inheritance in Multi-Objective Optimization. In *GECCO'2002*, pages 319–326. Morgan Kaufmann Publishers, 2002.
7. Mehrdad Salami and Tim Hendtlass. A Fast Evaluation Estrategy for Evolutionary Algorithms. *Applied Soft Computing*, 2(3):156–173, 2003.
8. Els I. Ducheyne, Bernard De Baets, and Robert De Wulf. Is Fitness Inheritance Useful for Real-World Applications? In *EMO 2003*, pages 31–42. Springer. LNCS 2632, 2003.
9. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

10. Martin Pelikan and Kumara Sastry. Fitness Inheritance in the Bayesian Optimization Algorithm. Technical Report 2004009, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 2004.
11. Lam T. Bui, Hussein A. Abbass, and Daryl Essam. Fitness Inheritance for Noisy Evolutionary Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, pages 25–29. ACM, 2005.
12. 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.
13. Margarita Reyes Sierra and Carlos A. Coello Coello. Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *IEEE Swarm Intelligence Symposium*, pages 116–123, Pasadena, California, USA, 2005. IEEE Service Center.
14. Alain Ratle. Accelerating the Convergence of Evolutionary Algorithms by Fitness Landscape Approximation. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN V), LNCS 3242*, pages 87–96. Springer-Verlag, 1998.
15. Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On Evolutionary Optimization with Approximate Fitness Function. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 786–793. Morgan Kaufmann Publishers, 2000.
16. Yasuhito Sano and Hajime Kita. Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 571–580. Springer-Verlag, 2000.
17. Jürgen Branke anad Christian Schmidth and Hartmut Schmeck. Efficient Fitness Estimation in Noisy Environments. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 243–250. Morgan Kaufmann Publishers, 2001.
18. J. Branke and C. Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9(1):13–20, January 2005.
19. Yew S. Ong, Prasanth B. Nair, and Andrew J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, April 2003.
20. Yew-Soon Ong, Zongzhao Zhu, and Dudy Lim. Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, Vancouver, Canada, July 2006. IEEE Service Center.
21. A. Gaspar-Cunha, Armando S. Vieira, and Carlos M. Fonseca. Multi-Objective Optimization: Hybridization of an Evolutionary Algorithm with Artificial Neural Networks for Fast Convergence. In *4th EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*, page http://webhost.ua.ac.be/eume/welcome.htm?workshops/hybrid/index.php&1, Nottingham, UK, November 2004. European Association of OR Societies.
22. Rommel G. Regis and Christine A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8(5):490–505, October 2004.

23. Dudy Lim, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Trusted evolutionary algorithm. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, Vancouver, Canada, July 2006. IEEE Service Center.
24. Ivan Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006. The Institute for People-centered Computation (IP-CC).
25. Joshua Knowles. ParEGO: A Hybrid Algorithm With On-Line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, February 2006.
26. James Kennedy and Russell C. Eberhart. Particle Swarm Optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, New Jersey, 1995. IEEE Service Center.
27. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation*, 10(3):263–282, 2002.
28. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *CEC 2002*, volume 1, pages 825–830, USA, 2002. IEEE Service Center.
29. 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.
30. Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *IEEE Swarm Intelligence Symposium Proceedings*, pages 26–33, USA, 2003. IEEE Service Center.
31. Gregorio Toscano Pulido and Carlos A. Coello Coello. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In *GECCO 2004. Part I*, pages 225–237, USA, 2004. Springer-Verlag, LNCS 3102.
32. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.
33. David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.