Natural Computing **1:** 235–306, 2002.

Recent approaches to global optimization problems through Particle Swarm Optimization

K.E. PARSOPOULOS[1] and M.N. VRAHATIS[2]
Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece
([1]E-mail: kostasp@math.upatras.gr; [2]E-mail: vrahatis@math.upatras.gr)

Abstract. This paper presents an overview of our most recent results concerning the Particle Swarm Optimization (PSO) method. Techniques for the alleviation of local minima, and for detecting multiple minimizers are described. Moreover, results on the ability of the PSO in tackling Multiobjective, Minimax, Integer Programming and ℓ_1 errors-in-variables problems, as well as problems in noisy and continuously changing environments, are reported. Finally, a Composite PSO, in which the heuristic parameters of PSO are controlled by a Differential Evolution algorithm during the optimization, is described, and results for many well-known and widely used test functions are given.

Key words: Differential Evolution, Evolutionary Computation, Global Optimization, Integer Programming, Matlab Code Implementation, Minimax Problems, Multiobjective Optimization, Noisy Problems, Particle Swarm Optimization, Swarm Intelligence

Abbreviations: ANN – Artificial Neural Network; BWA – Bang-Bang Weighted Aggregation; CWA – Conventional Weighted Aggregation; DE – Differential Evolution; DLS – Dynamic Light Scattering; DWA – Dynamic Weighted Aggregation; EA – Evolutionary Algorithms; EC – Evolutionary Computation; ES – Evolution Strategies; GA – Genetic Algorithms; GO – Global Optimization; MO – Multiobjective Optimization; PSO – Particle Swarm Optimization; SLS – Static Light Scattering; SPSO – "Stretched" PSO; VEGA – Vector Evaluated Genetic Algorithms; VEPSO – Vector Evaluated PSO

ACM Computing Classification (1998): G.1.6, F.1.2, G.3–4, I.2.8

1. Introduction

In numerous optimization problems encountered in different areas of scientific inquiry, the search for a solution is identified with the discovery of the global minimizer of a real valued objective function $f : S \to \mathbb{R}$, i.e., finding a point $x^* \in S$ such that

$$f(x^*) \leqslant f(x), \quad \forall x \in S, \tag{1}$$

where $S \subset \mathbb{R}^D$ is a nonempty compact set.

Global Optimization (GO) methods can be classified into two main categories: *deterministic* and *probabilistic* methods. Most of the deterministic methods involve the application of heuristics, such as modifying the trajectory (trajectory methods) or adding penalties (penalty-based methods), to escape from local minima. On the other hand, probabilistic methods rely on probabilistic judgements to determine whether or not search should depart from the neighborhood of a local minimum (Forgó, 1988; Hansen, 1992; Horst and Pardalos, 1995; Horst and Tuy, 1996; Pintér, 1996; Rao, 1996; Schwefel, 1995; Törn and Žilinskas, 1989).

In contrast with different adaptive stochastic search algorithms, Evolutionary Computation (EC) techniques (Bäck et al., 1997) exploit a set of potential solutions, named *population*, and detect the optimal problem solution through cooperation and competition among the individuals of the population. These techniques often find optima in complicated optimization problems faster than traditional optimization methods. The most commonly met population-based EC techniques, such as Evolution Strategies (ES) (Bäck, 1996; Beyer, 2001; Beyer and Schwefel, 2002; Rechenberg, 1994; Rudolph, 1997; Schwefel 1975; Schwefel, 1981; Schwefel, 1995; Schwefel and Rudolph, 1995), Genetic Algorithms (GA) (Goldberg, 1989; Michalewicz, 1994), Genetic Programming (Banzhaf et al., 1998; Koza, 1992), Evolutionary Programming (Fogel, 1996) and Artificial Life methods, are inspired from the evolution of nature.

The Particle Swarm Optimization (PSO) method is a member of the wide category of Swarm Intelligence methods (Kennedy and Eberhart, 2001), for solving GO problems. It was originally proposed by J. Kennedy as a simulation of social behavior, and it was initially introduced as an optimization method in 1995 (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995). PSO is related with Artificial Life, and specifically to swarming theories, and also with EC, especially ES and GA.

PSO can be easily implemented and it is computationally inexpensive, since its memory and CPU speed requirements are low (Eberhart et al., 1996). Moreover, it does not require gradient information of the objective function under consideration, but only its values, and it uses only primitive mathematical operators. PSO has been proved to be an efficient method for many GO problems and in some cases it does not suffer the difficulties encountered by other EC techniques (Eberhart and Kennedy, 1995).

In this paper, some recent approaches for solving GO problems through PSO are presented. After an analytic description of the concepts behind PSO and its historical background in Section 2, a recently proposed technique, called Function "Stretching", is described in Section 3. Incorporating Function "Stretching" in PSO, the method becomes capable of alleviating local

minima of the objective function, thus increasing significantly its success rates. In Section 4, PSO is tested in noisy and continuously changing environments, where the function values are imprecise and the global minimizer moves within the search space. In this context, an application of PSO for solving Light Scattering problems, is presented. The PSO's ability to cope efficiently with Multiobjective Optimization problems is investigated in Section 5. In Section 6, PSO is used to solve ℓ_1 norm errors-in-variables problems, determining the best model for fitting sets of data. Two other interesting classes of optimization problems are the Minimax and Integer Programming problems. Sections 7 and 8 respectively, are devoted to the investigation of the performance of the PSO method with respect to such problems. Finally, a technique for locating multiple global minima, as well as a composite PSO with dynamically controlled parameters by a Differential Evolution algorithm (Storn and Price, 1997), are presented in Sections 9 and 10 respectively, and conclusions are derived in Section 11.

2. Particle Swarm Optimization

2.1 *Historical background*

The implicit rules adhered to by the members of bird flocks and fish schools, that enable them to move synchronized, without colliding, resulting in an amazing choreography, was studied and simulated by several scientists (Heppner and Grenander, 1990; Reynolds, 1987). In simulations, the movement of the flock was an outcome of the individuals' (birds, fishes etc.) efforts to maintain an optimum distance from their neighboring individuals (Eberhart et al., 1996).

The social behavior of animals, and in some cases of humans, is governed by similar rules (Wilson, 1975). However, human social behavior is more complex than a flock's movement. Besides physical motion, humans adjust their beliefs, moving, thus, in a belief space. Although two persons cannot occupy the same space of their physical environment, they can have the same beliefs, occupying the same position in the belief space, without collision. This abstractness in human social behavior is intriguing and has constituted the motivation for developing simulations of it. There is a general belief, and numerous examples coming from nature enforce the view, that social sharing of information among the individuals of a population, may provide an evolutionary advantage. This was the core idea behind the development of PSO (Eberhart et al., 1996).

2.2 *The Particle Swarm Optimization algorithm*

PSO's precursor was a simulator of social behavior, that was used to visualize the movement of a birds' flock. Several versions of the simulation model were developed, incorporating concepts such as nearest-neighbor velocity matching and acceleration by distance (Eberhart et al., 1996; Kennedy and Eberhart, 1995). When it was realized that the simulation could be used as an optimizer, several parameters were omitted, through a trial and error process, resulting in the first simple version of PSO (Eberhart et al., 1996).

PSO is similar to EC techniques in that, a population of potential solutions to the problem under consideration, is used to probe the search space. However, in PSO, each individual of the population has an *adaptable velocity* (position change), according to which it moves in the search space. Moreover, each individual has a *memory*, remembering the best position of the search space it has ever visited (Eberhart et al., 1996). Thus, its movement is an aggregated acceleration towards its best previously visited position and towards the best individual of a topological neighborhood. Since the "acceleration" term was mainly used for particle systems in Particle Physics (Reeves, 1983), the pioneers of this technique decided to use the term *particle* for each individual, and the name *swarm* for the population, thus, coming up with the name *Particle Swarm* for their algorithm (Kennedy and Eberhart, 1995).

Two variants of the PSO algorithm were developed. One with a global neighborhood, and one with a local neighborhood. According to the global variant, each particle moves towards its best previous position and towards the best particle in the whole swarm. On the other hand, according to the local variant, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood (Eberhart et al., 1996). In the following paragraphs, the global variant is exposed (the local variant can be easily derived through minor changes).

Suppose that the search space is D-dimensional, then the i-th particle of the swarm can be represented by a D-dimensional vector, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^\top$. The *velocity* (position change) of this particle, can be represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^\top$. The best previously visited position of the i-th particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^\top$. Defining g as the index of the best particle in the swarm (i.e., the g-th particle is the best), and let the superscripts denote the iteration number, then the swarm is manipulated according to the following two equations (Eberhart et al., 1996):

$$v_{id}^{n+1} = v_{id}^{n} + cr_1^n(p_{id}^n - x_{id}^n) + cr_2^n(p_{gd}^n - x_{id}^n), \tag{2}$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1}, \tag{3}$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$, and N is the size of the swarm; c is a positive constant, called *acceleration constant*; r_1, r_2 are random numbers, uniformly distributed in [0, 1]; and $n = 1, 2, \ldots$, determines the iteration number.

Equations (2) and (3) define the initial version of the PSO algorithm. Since there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value V_{max} on it. If the velocity exceeded this threshold, it was set equal to V_{max}. This parameter proved to be crucial, because large values could result in particles moving past good solutions, while small values could result in insufficient exploration of the search space. This lack of a control mechanism for the velocity resulted in low efficiency for PSO, compared to EC techniques (Angeline, 1998). Specifically, PSO located the area of the optimum faster than EC techniques, but once in the region of the optimum, it could not adjust its velocity stepsize to continue the search at a finer grain.

The aforementioned problem was addressed by incorporating a weight parameter for the previous velocity of the particle. Thus, in the latest versions of the PSO, Equations (2) and (3) are changed to the following ones (Eberhart and Shi, 1998; Shi and Eberhart, 1998a, Shi and Eberhart, 1998b):

$$v_{id}^{n+1} = \chi \left(w v_{id}^{n} + c_1 r_1^n (p_{id}^n - x_{id}^n) + c_2 r_2^n (p_{gd}^n - x_{id}^n) \right), \tag{4}$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1}, \tag{5}$$

where w is called *inertia weight*; c_1, c_2 are two positive constants, called *cognitive* and *social* parameter respectively; and χ is a *constriction factor*, which is used, alternatively to w to limit velocity. The role of these parameters is discussed in the next section.

In the local variant of PSO, each particle moves towards the best particle of its neighborhood. For example, if the size of the neighborhood is 2, then the i-th particle moves towards the best particle among the $(i - 1)$-th, the $(i + 1)$-th and itself.

The PSO method appears to adhere to the five basic principles of swarm intelligence, as defined by (Eberhart et al., 1996; Millonas, 1994):

(a) *Proximity*, i.e., the swarm must be able to perform simple space and time computations;
(b) *Quality*, i.e., the swarm should be a able to respond to quality factors in the environment;
(c) *Diverse response*, i.e., the swarm should not commit its activities along excessively narrow channels;
(d) *Stability*, i.e., the swarm should not change its behavior every time the environment alters; and finally

(e) *Adaptability*, i.e., the swarm must be able to change its behavior, when the computational cost is not prohibitive.

Indeed, the swarm in PSO performs space calculations for several time steps. It responds to the quality factors implied by each particle's best position and the best particle in the swarm, allocating the responses in a way that ensures diversity. Moreover, the swarm alters its behavior (state) only when the best particle in the swarm (or in the neighborhood, in the local variant of PSO) changes, thus, it is both adaptive and stable (Eberhart et al., 1996).

2.3 *The parameters of PSO*

The role of the *inertia weight* w, in Equation (4), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter w regulates the trade-off between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e., fine-tuning the current search area. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. Initially, the inertia weight was constant. However, experimental results indicated that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions (Shi and Eberhart, 1998a; Shi and Eberhart, 1998b). Thus, an initial value around 1.2 and a gradual decline towards 0 can be considered as a good choice for w.

The parameters c_1 and c_2, in Equation (4), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima. An extended study of the acceleration parameter in the first version of PSO, is given in (Kennedy, 1998). As default values, $c_1 = c_2 = 2$ were proposed, but experimental results indicate that $c_1 = c_2 = 0.5$ might provide even better results. Recent work reports that it might be even better to choose a larger cognitive parameter, c_1, than a social parameter, c_2, but with $c_1 + c_2 \leqslant 4$ (Carlisle and Dozier, 2001).

The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0, 1]. The constriction factor χ controls on the magnitude of the velocities, in a way similar to the V_{max} parameter, resulting in a variant of PSO, different from the one with the inertia weight.

2.4 *Differences between PSO and EC techniques*

In EC techniques, three main operators are involved. The *recombination*, the *mutation* and the *selection* operator.

PSO does not have a direct recombination operator. However, the stochastic acceleration of a particle towards its previous best position, as well as towards the best particle of the swarm (or towards the best in its neighborhood in the local version), resembles the recombination procedure in EC (Eberhart and Shi, 1998; Rechenberg, 1994; Schwefel, 1975; Schwefel, 1995). In PSO the information exchange takes place only among the particle's own experience and the experience of the best particle in the swarm, instead of being carried from fitness dependent selected "parents" to descendants as in GA's.

Moreover, PSO's directional position updating operation resembles mutation of GA, with a kind of memory built in. This mutation-like procedure is multidirectional both in PSO and GA, and it includes control of the mutation's severity, utilizing factors such as the V_{max} and χ.

PSO is actually the only evolutionary algorithm that does not use the "survival of the fittest" concept. It does not utilize a direct selection function. Thus, particles with lower fitness can survive during the optimization and potentially visit any point of the search space (Eberhart and Shi, 1998).

3. The Function "Stretching" technique

3.1 *Motivation*

Perhaps the most common problem encountered by many GO methods, either deterministic or evolutionary, when coping with the GO problem as defined in Equation (1), is the problem of local minima (Forgó, 1988; Hansen, 1992; Horst and Pardalos, 1995; Horst and Tuy, 1996; Pintér, 1996; Rao, 1996; Schwefel, 1995; Törn and Žilinskas, 1989). Especially in multimodal functions, the existence of many local minima makes it quite difficult for most techniques to detect the global minimum. PSO, despite being an efficient method, also suffers from this problem.

Assume a point $\bar{x}$ such that a neighborhood B of $\bar{x}$ with

$$f(\bar{x}) \leqslant f(x), \quad \forall x \in B, \tag{6}$$

exists. This point is a local minimizer of the objective function $f(x)$ and in many cases this sub-optimal solution is acceptable. However, there are applications where the optimal solution, i.e., the global minimizer is not only desirable but also indispensable. Therefore, the development of robust and

efficient techniques for alleviating the local minima problem is a subject of considerable ongoing research. To this end, the *Function "Stretching"* technique is applied, and it is shown through simulation experiments that Function "Stretching" provides a way to escape from the local minima when PSO's convergence stagnates.

3.2 *Equipping PSO with Function "Stretching"*

The idea behind Function "Stretching", is to perform a two-stage transformation of the original objective function $f(x)$. This can be applied immediately after a local minimum $\bar{x}$ of the function $f(x)$ has been detected. This transformation has been proposed by Vrahatis in 1996, and it is defined as follows:

$$G(x) = f(x) + \gamma_1 \|x - \bar{x}\| \Big(\text{sign}(f(x) - f(\bar{x})) + 1\Big), \tag{7}$$

$$H(x) = G(x) + \gamma_2 \frac{\text{sign}\Big(f(x) - f(\bar{x})\Big) + 1}{\tanh\Big(\mu(G(x) - G(\bar{x}))\Big)}, \tag{8}$$

where γ_1, γ_2, and μ are arbitrary chosen positive constants, and sign$(\cdot)$ defines the well known triple valued sign function

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0, \\ -1, & \text{if } x < 0. \end{cases}$$

The first transformation stage, defined in Equation (7), elevates the function $f(x)$ and eliminates all the local minima that are located above $f(\bar{x})$. The second stage, Equation (8), stretches the neighborhood of $\bar{x}$ upwards, since it assigns higher function values to those points. Both stages do not alter the local minima located below $\bar{x}$; thus, the location of the global minimum is left unaltered. Note that the sign$(\cdot)$ function, used in the above transformation, can be approximated by the well known logistic function

$$\text{sign}(x) \approx \text{logsig}(x) = \frac{2}{1 + \exp(-\lambda x)} - 1 \simeq \tanh\left(\frac{\lambda x}{2}\right),$$

for large values of λ. This sigmoid function is continuously differentiable and is widely used as a transfer function in artificial neural networks.

At this point, it is useful to illustrate the impact of the two transformations suggested by the Function "Stretching" technique, on a well known optimization test function (Parsopoulos et al., 2001a; Parsopoulos et al., 2001b,

Parsopoulos et al., 2001c). The problem under consideration is a notorious two dimensional test function, called the *Levy No. 5* (Levy et al., 1981):

$$f(x) = \sum_{i=1}^{5}\Big[i\,\cos\Big((i-1)x_1+i\Big)\Big]\sum_{j=1}^{5}\Big[j\,\cos\Big((j+1)x_2+j\Big)\Big] + (x_1+1.42513)^2+(x_2+0.80032)^2, \quad (9)$$

where $-10 \leqslant x_i \leqslant 10$, and $i = 1, 2$. There are about 760 local minima and one global minimum with function value $f(x^*) = -176.1375$, at the point $x^* = (-1.3068, -1.4248)^\top$. The large number of local minimizers makes it difficult for any method to locate the global minimizer. In Figure 1, the original plot of the *Levy No. 5* function, within the cube $[-2, 2]^2$, is shown.

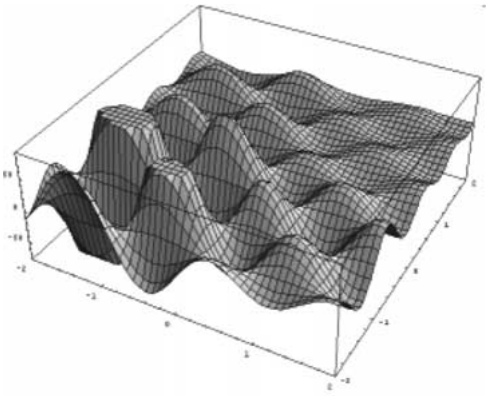

Figure 1. Plot of the original *Levy No. 5* function.

After applying the transformation of Equation (7) (first stage of Function "Stretching") to the *Levy No. 5* function, the new form of the function is shown on the upper side of Figure 2. As one can observe, local minima with higher functional values than the "stretched" local minimum (which looks as if a pin is positioned over it and the rest of the function is stretched around it) disappeared, while lower minima as well as the global one have not been affected. The final shape of the landscape is shown on the lower side of Figure 2; being the result of applying the second transformation stage to the *Levy No. 5*. It is clearly shown how the whole neighborhood of the local minimum has been elevated; thus, the former local minimum has been turned into a local maximum of the function.

Table 1, below, provides an algorithm model for a modified PSO method, named "Stretched" PSO (SPSO). SPSO initially applies the PSO method, for minimizing the objective function $f(x)$. When PSO stumbles upon a local

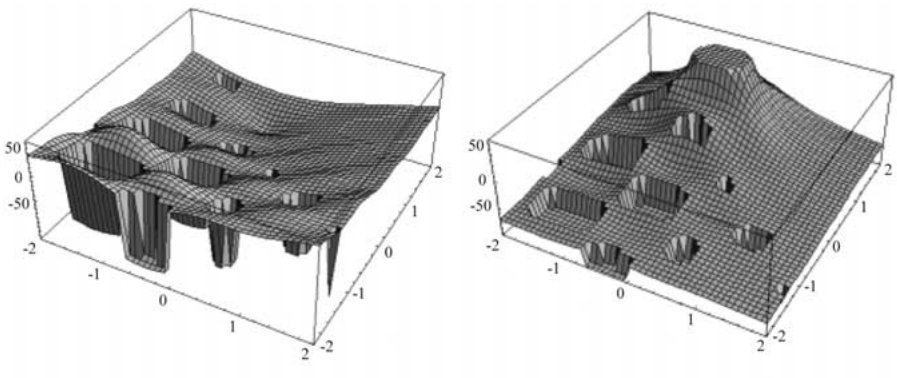

Figure 2. Plot of the *Levy No. 5* function, after the first stage (left) and the second stage (right) of the Function "Stretching" transformations.

minimum, the Function "Stretching" technique is applied to the original function and SPSO is re-initialized with the PSO for the minimization of the stretched function.

Table 1. Algorithm model of the SPSO.

"Stretched" Particle Swarm Optimizer	
Step 1.	**Set** randomly the population and the velocities.
Step 2.	**While** (Stopping Criterion not met) **Do**
Step 3.	**Update** inertia and perform a PSO iteration for the objective function $f(x)$.
Step 4.	**If** a local minimum $\bar{x}$ has been found, **Then**
Step 5.	**Set** $f(x) \leftarrow H(x)$, where $H(x)$ is defined in Equation (8).
Step 5.	**End While**
Step 6.	**Report** the results.

Details on the performance of the SPSO in some well known and widely used test problems, as well as suggestions for selecting parameter values for the Function "Stretching" technique are presented in the next section.

3.3 *Experimental results*

In this subsection, results from testing the classical PSO method and the "Stretched" PSO on well-known test functions, are reported. Difficult optimization problems were studied, such as the minimization of several of the *Levy* family test functions (Levy et al., 1981), the 2-dimensional and the

Table 2. Analysis of the results for the minimization of several test problems. Mean and the standard deviation of the number of required function evaluations, as well as the corresponding success rates.

Test Problem	"Stretching" applied			PSO			SPSO		
	Mean	St.D.	Succ.	Mean	St.D.	Succ.	Mean	St.D.	Succ.
Levy No. 3	15246.6	6027.3	15%	5530.5	6748.0	85%	6988.0	7405.5	100%
Levy No. 5	3854.2	1630.1	7%	1049.4	235.1	93%	1245.8	854.2	100%
Levy No. 8	0	0	0%	509.6	253.2	100%	509.6	253.2	100%
Freud.-Roth	3615.0	156.1	40%	1543.3	268.1	60%	2372.0	1092.1	100%
Goldst.-Price	17420.0	3236.56	5%	1080.0	225.6	95%	1897.0	3660.3	100%
Corana 2D	0	0	0%	1409.6	392.8	100%	1409.6	392.8	100%
Corana 4D	13704.6	7433.5	26%	2563.2	677.5	74%	5460.0	6183.8	100%
XOR	29328.6	15504.2	23%	1459.7	1143.1	77%	7869.6	13905.4	100%

4-dimensional *Corana* functions (Corana et al., 1987), the *Freudenstein-Roth* and *Goldstein-Price* functions (More et al., 1981), and a classical Artificial Neural Networks (ANN) pattern classification problem, i.e., the classification of the eXclusive–OR (XOR) patterns.

In all the simulations reported, the values of γ_1, γ_2 and μ were fixed: $\gamma_1 = 5000$, $\gamma_2 = 0.5$ and $\mu = 10^{-10}$. Default values for the parameters c_1 and c_2 were used: $c_1 = c_2 = 0.5$. Although the choice of the parameter values seems not to be critical for the success of the methods, it appears that faster convergence can be obtained by proper fine-tuning. The balance between the global and local exploration abilities of the SPSO is mainly controlled by the inertia weight, since the positions of the particles are updated according to the classical PSO strategy. A time decreasing inertia weight value, i.e., start from 1 and gradually decrease towards 0.4, proved to be superior than a constant value.

For each problem 100 runs have been performed using the SPSO and the average performance is exhibited in terms of the mean value and the standard deviation of the required number of function evaluations, and the percentage of SPSO success. Results for all of the aforementioned problems are reported in Table 2, while the size of the population used for each problem, as well as the initial hypercube in which the initial population was randomly initialized, are presented in Table 3.

Table 3. Dimension, swarm's size, and initial hypercube for each test problem.

Test Problem	Dim.	Popul. Size	Initial Hypercube
Levy No. 3	2	20	$[-10, 10]^2$
Levy No. 5	2	20	$[-2, 2]^2$
Levy No. 8	3	20	$[-5, 5]^3$
Freud.-Roth	2	20	$[-5, 5]^2$
Goldst.-Price	2	20	$[-2, 2]^2$
Corana 2D	2	20	$[-5, 5]^2$
Corana 4D	4	80	$[-1, 1]^4$
XOR	9	80	$[-1, 1]^9$

TEST PROBLEM 3.1, *Levy No. 3* (Levy et al., 1981):

$$f(x) = \sum_{i=1}^{5} \Big[i \cos\Big((i-1)x_1 + i\Big)\Big] \sum_{j=1}^{5} \Big[j \cos\Big((j+1)x_2 + j\Big)\Big], \quad (10)$$

where $-10 \leqslant x_i \leqslant 10$, $i = 1, 2$. There are about 760 local minima for this function and 18 global minima with function value $f(x^*) = -176.542$. As it can be seen from Table 2, in 85 out of 100 cases the plain PSO found the global minimum, while in 15 cases it was entrapped in a local minimum, and Function "Stretching" was successfully applied. Thus the success rate of PSO increased by 15%.

TEST PROBLEMS 3.2–3.3, *Levy No. 5* and *Levy No. 8* (Levy et al., 1981): The *Levy No. 5* function is given by Equation (9), while the *Levy No. 8* function is defined by the equation:

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2, \quad (11)$$

where $y_i = 1 + \frac{x_i - 1}{4}$, $i = 1, \ldots, n$; $x_i \in [-10, 10]$ for $i = 1, 2, 3$; and has one global minimum at the point $x^* = (1, 1, 1)^\top$ with function value $f(x^*) = 0$, and, approximately, 125 local minima.

An increase of the success rate, from 93% to 100%, was observed for the *Levy No. 5* function, where "Stretching" was applied in 7 cases, while for the *Levy No. 8*, plain PSO detected the global minimum in all 100 runs.

TEST PROBLEM 3.4, *Corana* function (Storn and Price, 1997):

$$f(x) = \sum_{j=1}^{4} \begin{cases} 0.15 \left(z_j - 0.05 \operatorname{sign}(z_j) \right)^2 d_j & , \text{ if } |x_j - z_j| < 0.05, \\ d_j \, x_j^2 & , \text{ otherwise,} \end{cases} \tag{12}$$

where $x_j \in [-1000, 1000]$, $(d_1, d_2, d_3, d_4) = (1, 1000, 10, 100)$, and

$$z_j = \left\lfloor \left| \frac{x_j}{0.2} \right| + 0.49999 \right\rfloor \operatorname{sgn}(x_j) \, 0.2.$$

It is indeed a very difficult minimization problem for most methods. As illustrated in Table 2, plain PSO had only 74% success rate, but using SPSO the success rate increased to 100%.

Similar results are reported for the following test problems, where the PSO success rate was increased by 40% and 5% respectively.

TEST PROBLEM 3.5, *Freudenstein-Roth* function:
The *Freudenstein-Roth* function is defined as

$$\begin{aligned} f(x) \; = \; & [-13 + x_1 + ((5 - x_2)x_2 - 2)x_2]^2 + \\ & + [-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2]^2, \end{aligned} \tag{13}$$

and has a global minimizer $x^* = (5, 4)^\top$ with function value $f(x^*) = 0$.

TEST PROBLEM 3.6, *Goldstein-Price* function:
The *Goldstein-Price* function is given by the formula

$$\begin{aligned} f(x) = & \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \\ & \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \right], \end{aligned} \tag{14}$$

and has one global minimum with function value $f(x^*) = 3$ at the point $x^* = (0, -1)^\top$.

Other optimization methods, such as Steepest Descent with adaptive step-size and backtracking (SD), Fletcher – Reeves (FR), Polak – Ribiere (PR), Davidon – Fletcher – Powell (DFP) and Broyden – Fletcher – Goldfarb – Shanno (BFGS) (Polak, 1997), were tested in the *Freudenstein-Roth* and the *Goldstein-Price* functions. None of the aforementioned methods outperformed the SPSO, as can be seen from the results exhibited in Table 4.

In a different experiment, an ANN was trained using the SPSO to learn the XOR Boolean classification problem. The XOR function maps two binary

Table 4. Comparative results for the Freudenstein-Roth and the Goldstein-Price test functions. Mean and the standard deviation of the number of the required function evaluations, as well as the corresponding success rates.

Algorithm	Freudenstein-Roth			Goldstein-Price		
	Mean	St.D.	Succ.	Mean	St.D.	Succ.
SPSO	2372	1092.1	100%	1897	3660.3	100%
SD	41651	6153.4	77.3%	700	363.9	84.3%
FR	245	599.8	44.3%	1475	3097.1	71.4%
PR	120	13.2	43.8%	238	41.3	79.9%
DFP	91	9.7	43.7%	162	28.6	59.3%
BFGS	91	9.8	43.8%	161	27.0	59.9%

inputs to a single binary output and the ANN that is trained to solve the problem, has two linear input nodes, two hidden nodes, and one output node, all with logistic activation functions. This task corresponds to the minimization of the following objective function (Vrahatis et al., 2000):

TEST PROBLEM 3.7, *XOR* function (Vrahatis et al., 2000):

$$
\begin{aligned}
f(x) \;=\; & \left[1+\exp\left(-\frac{x_7}{1+\exp(-x_1-x_2-x_5)}-\frac{x_8}{1+\exp(-x_3-x_4-x_6)}-x_9\right)\right]^{-2} \\
& +\left[1+\exp\left(-\frac{x_7}{1+\exp(-x_5)}-\frac{x_8}{1+\exp(-x_6)}-x_9\right)\right]^{-2} \\
& +\left[1-\left\{1+\exp\left(-\frac{x_7}{1+\exp(-x_1-x_5)}-\frac{x_8}{1+\exp(-x_3-x_6)}-x_9\right)\right\}^{-1}\right]^{2} \\
& +\left[1-\left\{1+\exp\left(-\frac{x_7}{1+\exp(-x_2-x_5)}-\frac{x_8}{1+\exp(-x_4-x_6)}-x_9\right)\right\}^{-1}\right]^{2}.
\end{aligned}
$$

In the context of ANNs, the parameters $x_1, x_2, \ldots, x_9$, are called *weights*, and they are usually initialized in the interval [–1, 1]. It is well known from the relative literature that successful training in this case, i.e., reaching a global minimizer, strongly depends on the initial weight values and that the above-mentioned function presents a multitude of local minima (Blum, 1989). It is obvious from the results reported in Table 2 that the Function "Stretching" technique increases significantly the success rates of the PSO, indeed the success rate was increased from 77% to 100%.

3.4 *Synopsis*

Locating global minimizers is a very challenging task for all minimization methods. An effective technique, named Function "Stretching", was applied

for the alleviation of the local minima problem for the PSO. This technique applies a two-stage transformation to the shape of the fitness function that eliminates undesirable local minima, but preserves the global ones.

Experiments on many difficult optimization test problems indicate that the PSO method, when equipped with the proposed technique (SPSO), is capable of escaping from the areas of local minima and locate the global minimizer effectively. The Function "Stretching" technique provides stable convergence and thus a better probability of success for the PSO. The heavier computational cost paid for SPSO, is balanced by the increased success rates, which imply that the final result is the global minimum of the objective function with high probability.

4. Particle Swarm Optimization for noisy and continuously changing environments

4.1 *Motivation and problem formulation*

Most of the GO methods, require precise function and gradient values. In many applications though, precise values are either impossible or time consuming to obtain. For example, when the function and gradient values depend on the results of numerical simulations, then it may be difficult or impossible to get very precise values. In other cases, it may be necessary to integrate numerically a system of differential equations in order to obtain a function value, so that the precision of the computed value is limited (Drossos et al., 1996; Kalantonis et al., 2001). Furthermore, in many problems the accurate values of the function to be minimized are computationally expensive (Vrahatis et al., 2001). Such problems are common in real life applications, such as the optimization of parameters in chemical experiments, or finite element calculations. Having such applications in mind, robust methods which progress with the fewest possible number of function evaluations and limited precision, are required.

The problem of optimization of noisy or imprecise (not exactly known) functions occurs in various applications, as, for instance, in the task of experimental optimization. Also, the problem of locating local maxima and minima of a function from approximate measurement results is vital for many physical applications. In spectral analysis, chemical species are identified by locating local maxima of the spectra. In Radioastronomy, sources of celestial radio emission and their subcomponents are identified by locating local maxima of the measured brightness of the radio sky. Elementary particles are identified by locating local maxima of the experimental curves.

The theory of local optimization provides a large variety of efficient methods for the computation of an optimizer of a smooth function $f(x)$. For example, Newton-type and quasi-Newton methods exhibit superlinear convergence in the vicinity of a nondegenerate optimizer. However, these methods require the Hessian or the gradient, respectively, in contrast to other optimization procedures, like the Nonlinear Simplex method (Nelder and Mead, 1965), the direction set method of Powell (Fletcher, 1987), or some other recently proposed methods (Boutsinas and Vrahatis, 2001; Elster and Neumaier, 1995; Elster and Neumaier, 1997; Vrahatis et al., 1996).

In some applications, however, the function to be minimized is only known within some (often unknown and low) precision. This might be due to the fact that evaluation of the function implies measuring some physical or chemical quantity or performing a finite element calculation in order to solve partial differential equations. The function values obtained are corrupted by noise, namely stochastic measurement errors or discretization errors. This means that, although the underlying function is smooth, the function values available show a discontinuous behavior. Moreover, no gradient information is available. For small variances in a neighborhood of a point the corresponding function values reflect the local behavior of noise rather than that of the function. Thus, a finite-difference procedure to estimate the gradient fails (Elster and Neumaier, 1995).

The traditional method for optimizing noisy functions is the simplex or polytope method by Nelder and Mead (Nelder and Mead, 1965; Nocedal, 1991; Powell, 1988). This method surpasses other well-known optimization methods when dealing with large noise. Yet, this is not valid in the noiseless case. Torczon (Torczon, 1989) has shown experimentally that the algorithm of Nelder and Mead fails even on the sphere unless the dimensionality of the search space is very small. The ability of this method to cope with noise is due to the fact that it proceeds solely by comparing the relative size of function values, as is the case with the proposed method. The Simplex method does not use a local model of the function f and works without the assumption of continuity. Although this method has poor convergence properties (for a convergence proof of a modified version see (Torczon, 1991)), it proved to be useful in many sequential applications. Unfortunately, the method can be deficient when the current simplex is very "flat". This can be avoided by applying suitable variants (see for example (Torczon, 1991)). More sophisticated methods in this direction are discussed by Powell (Powell, 1992).

Recently, Arnold in his Ph.D. thesis (Arnold, 2001) extensively tested numerous optimization methods under noise, including:

(1) the direct pattern search algorithm of Hooke and Jeeves (Hooke and Jeeves, 1961),

(2) the simplex metdod of Nelder and Mead (Nelder and Mead, 1965),
(3) the multi-directional search algorithm of Torczon (Torczon, 1989),
(4) the implicit filtering algorithm of Gilmore and Kelley (Gilmore and Kelley, 1995; Kelley, 1999) that is based on explicitly approximating the local gradient of the objective functions by means of finite differencing,
(5) the simultaneous perturbation stochastic approximation algorithm due to Spall (Spall, 1992; Spall, 1998a; Spall, 1998b),
(6) the evolutionary gradient search algorithm of Salomon (Salomon, 1998),
(7) the evolution strategy with cumulative mutation strength adaptation mechanism by Hansen and Ostermeier (Hansen, 1998; Hansen and Ostermeier, 2001).

To study the behavior of the above methods Arnold considered a simple n-dimensional sphere model (see also (Beyer, 2000; Beyer, 2001)) and extensively tested these methods in the presence of additive Gaussian noise. For almost all of the methods he considered there exists a noise strength beyond which convergence becomes unreliable. In most cases, this effect leads to stagnation of the search process. He also made the following observations:

(1) the efficiency of the direct pattern search algorithm of Hooke and Jeeves is quite good in the absence of noise, but rather rapidly declines if noise is present even in small-dimensional search spaces,
(2) the simplex method of Nelder and Mead fails even on the sphere unless the dimension of the search space is very small, and moreover the presence of noise worsens the tendency of the method to stagnate at non-optimal points,
(3) the multi-directional search algorithm of Torczon never stagnates but rather diverges if the noise level is too high,
(4) the implicit filtering method in the absence of noise converges much faster than any of the other methods as the exact gradient direction is obtained. This method exhibits poor performance in the presence of high noise levels,
(5) the attempt to use the simultaneous perturbation stochastic approximation to the gradient approximation, instead of computing the central difference gradient in the implicit filtering scheme, has failed even for zero noise strength,
(6) the evolutionary gradient search algorithm is one of the most efficient strategies excluding implicit filtering in the absence of noise. In the presence of noise the efficiency of this method declines,
(7) the evolution strategy with cumulative mutation strength adaptation mechanism by Hansen and Ostermeier clearly is the most robust strategy with respect to the effects of noise.

In this context, to study the impact of imprecise information (regarding the values of the objective function), we simulate imprecisions through the following approach. Information about $f(x)$ is obtained in the form of $f^{\eta}(x)$, where $f^{\eta}(x)$ is an approximation to the true function value $f(x)$, contaminated by a small amount of noise η. Specifically, the function values are obtained, for the additive noise case, as (Elster and Neumaier, 1997, p. 40):

$$f^{\eta}(x) = f(x) + \eta, \tag{15}$$

and for the multiplivative noise case, as:

$$f^{\eta}(x) = f(x)(1 + \eta), \qquad 1 + \eta > 0, \tag{16}$$

where η is a Gaussian noise term with zero mean and standard deviation σ

$$\eta \sim \mathcal{N}\left(0, \sigma^2\right), \tag{17}$$

i.e., relative stochastic errors are used for the test problems. Assuming a normal noise distribution may be regarded as an approximation of reality based on the maximum entropy principle (Beyer, 2000; Jaynes, 1979), the probability density function of the noise reads

$$p(\eta) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\frac{1}{2}\left(\frac{\eta}{\sigma}\right)^2\right]. \tag{18}$$

To obtain η, we apply the method of Box and Muller (Box and Muller, 1958), using various *noise strengths* (standard deviations) σ. In our approach, the assumption of the normal distribution is not mandatory. However, the measurement errors in nature and technology are very often modeled using this distribution. Moreover, most of the mean values of stochastic processes have this distribution (central limit theorem of statistics) (Beyer, 2001).

Furthermore, the global minimum of the objective function, potentially moves within the search space. The movement can be simulated by applying a rotation of the whole space and/or adding an offset to the global minimum.

4.2 *Experimental results*

In initial experiments, noise was added to the actual function values, according to Equations (16) and (17). Additive as well as multiplicative noise was considered.

Six well known optimization test problems were investigated (Parsopoulos and Vrahatis, 2001b; Parsopoulos and Vrahatis, 2001c), in order to check the performance of the PSO technique: *DeJong* in two and ten dimensions, *Rastrigin* in two dimensions, the *Hyper Ellipsoid* in six and nine dimensions (all defined in (Storn and Price, 1997)), and the *Levy No. 5* (defined in Equation (9)).

At each function evaluation, noise was added to the actual function values, for different values of the noise's strength, from 0.01 to 0.1. For all test problems, 50 runs of the PSO algorithm were performed. In all experiments the swarm's size was set equal to $[-5, 5]^D$, where D is the corresponding dimension of the problem, and $c_1 = c_2 = 0.5$. The maximum number of iterations was 5000 and the inertia weight was gradually decreased from 1.2 towards 0.1. The desired accuracy was 10^{-3} and the swarm's size was set equal to $10D$ for all problems (e.g., for a 10-dimensional problem it was equal to 100).

The mean distance of the obtained solution from the actual global minimum, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise, are reported in Tables 5–10.

In other experiments, an offset was added to the original global minimizer's position, at each iteration, as performed by Angeline (Angeline, 1997). The progress of the mean function value of the swarm, for 100 runs, for the *Rosenbrock*, the *Levy No. 5*, and the *Beale* test function, defined below, are exhibited in Figures 3, 4, and 5.

TEST PROBLEM 4.1, *Rosenbrock* function (More et al., 1981):
The *Rosenbrock* function is defined as

$$f(x_1, x_2) = (10^4 x_1 x_2 - 1)^2 + (1 - x_1)^2. \tag{19}$$

TEST PROBLEMS 4.2–4.3, *Levy No. 5* and *Beale* function (More et al., 1981):
The *Levy No. 5* function is defined by Equation (9), while the *Beale* function is defined as

$$f(x_1, x_2) = [y_1 - x_1(1 - x_2)]^2 + [y_2 - x_1(1 - x_2^2)]^2 + [y_3 - x_1(1 - x_2^3)]^2. \tag{20}$$

Different line styles in the figures correspond to different values of the offset. For all runs, a value of the noise's strength equal to 0.01 was used. The vertical axis of each plot is logarithmically scaled to facilitate visual comparison. In Figures 3 and 5, the logarithm $\log_{10}$ of the swarm's mean function value for 100 runs is exhibited, while in Figure 4 the logarithm $\log_{10}$ of the swarm's mean absolute error is exhibited, due to the negative values of the *Levy No. 5* function.

TEST PROBLEM 4.4, *Particle Identification by Light Scattering* (Hodgson, 2000):
A very interesting application where the function value is imprecise, is the *Particle Identification by Light Scattering problem*. Astronomy, Meteorology,

Table 5. Results for the 2-dimensional *DeJong* function. The standard deviation of noise, the mean distance of the obtained minimizer from the actual global minimizer, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.087383	1312.0	0.023535	1473.95
0.02	0.10389	1164.35	0.024269	1394.25
0.05	0.236045	941.95	0.018750	1394.55
0.07	0.256558	726.75	0.021531	1391.25
0.1	0.209896	724.8	0.017322	1400.05

Table 6. Results for the *Levy No. 5* function. The standard deviation of noise, the mean distance of the obtained minimizer from the actual global minimizer, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.002107	1510.6	0.146252	1875.7
0.02	0.116313	1854.9	0.152991	1715.2
0.05	0.005402	1512.8	0.059617	1502.1
0.07	0.007007	1488.2	0.171982	1218.4
0.1	0.006019	1512.4	0.814959	1483.0

Table 7. Results for the 2-dimensional *Rastrigin* function. The standard deviation of noise, the mean distance of the obtained minimizer from the mean actual global minimizer, and the number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.004984	1496.5	0.007625	1487.8
0.02	0.007584	1550.7	0.010616	1519.2
0.05	0.045541	1499.8	0.247842	1485.2
0.07	0.164456	1507.5	0.247611	1472.1
0.1	0.292265	1491.5	0.253639	1477.4

Table 8. Results for the 6-dimensional *Hyper Ellipsoid* function. The standard from the actual deviation of noise, the mean distance of the obtained minimizer global minimizer, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.049631	1518.8	0.011949	1522.0
0.02	0.060305	1507.4	0.012013	1517.6
0.05	0.112297	1505.6	0.011953	1522.4
0.07	0.124931	1507.2	0.011034	1518.6
0.1	0.143174	1500.6	0.017555	1518.6

Table 9. Results for the 9-dimensional *Hyper Ellipsoid* function. The standard deviation of noise, the mean distance of the obtained minimizer from the actual global minimizer, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.050984	1536.4	0.009048	1538.2
0.02	0.072194	1536.6	0.012693	1547.6
0.05	0.093558	1531.4	0.008291	1541.0
0.07	0.134740	1525.2	0.010766	1548.8
0.1	0.142287	1528.0	0.011485	1546.8

Table 10. Results for the 10-dimensional *DeJong* function. The standard deviation of noise, the mean distance of the obtained minimizer from the actual global minimizer, and the mean number of iterations performed by PSO, for both the additive and the multiplicative noise case, are reported.

Noise's St.D.	Additive Noise		Multiplicative Noise	
	Mean Dist.	Mean Iter.	Mean Dist.	Mean Iter.
0.01	0.143042	1525.0	0.029195	1536.5
0.02	0.193130	1530.9	0.029030	1538.6
0.05	0.293527	1524.9	0.028555	1535.9
0.07	0.376479	1522.8	0.028876	1539.0
0.1	0.444429	1517.8	0.032689	1540.6

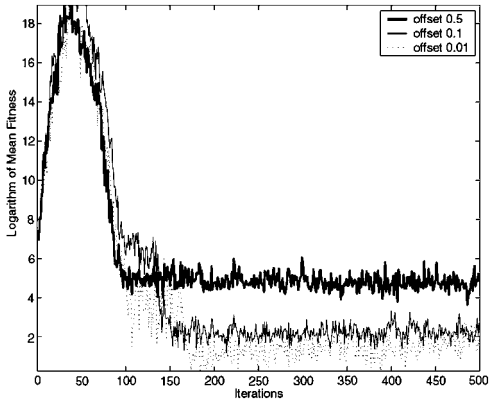

Figure 3. Mean fitness value for the Rosenbrock function.

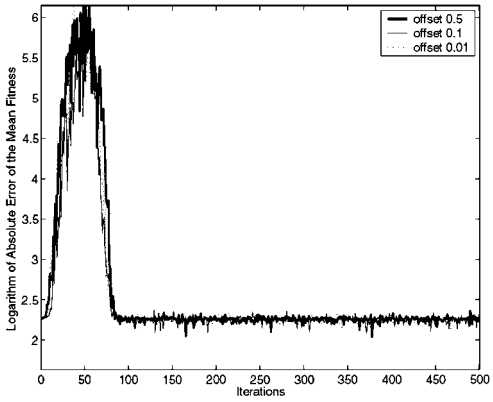

Figure 4. Mean fitness value for the Levy No. 5 function.

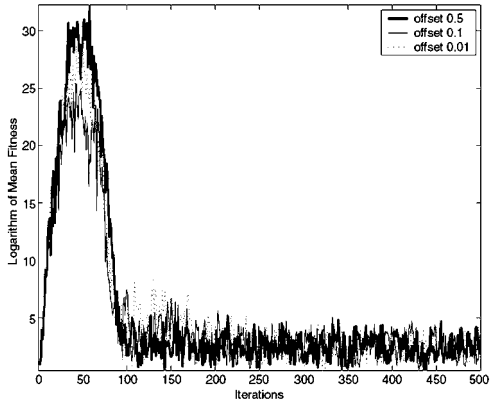

Figure 5. Mean fitness value for the Beale function.

Medicine and Bioengineering are just some of the fields where the laser light scattering measurements have become an important tool in the determination of the size and optical characteristics of small particles. There are actually two techniques for the determination of particle size using light scattering: Dynamic Light Scattering (DLS) and Static Light Scattering (SLS). In DLS a laser is focused into a small volume of solution containing the colloidal particles and the scattered light is collected over a small solid angle. The phase and polarization of the light scattered by any given molecule is determined by its size, shape, and composition. Random Brownian motion causes the total intensity at the detector to fluctuate at time. Autocorrelation functions are generated from these fluctuations and then inverted to obtain the distribution of particle sizes (Hodgson, 2000; Parsopoulos and Vrahatis, 2001c). On the other hand, in SLS the total intensity of the scattered light is measured as a function of angle and this information is used to determine the particle size distributions. For spheres the angular dependence of the scattered light is described by the Mie scattering function (Bohren and Huffman, 1983), which depends on the refractive indices of the particle, the surrounding medium, and the size of the particles present (Hodgson, 2000).

A fundamental problem in inverse light scattering is the determination of the refractive index and the radius of homogeneous spherical particles suspended in a known medium. In order to solve that problem, plane polarized light of known wavelength is scattered from these particles and the intensity I is measured at a series of angles. The standard Lorenz-Mie theory (Bohren and Huffman, 1983) is used to describe the process and the one-dimensional scattering pattern observed is used to characterize both single particles and particle distributions. Thus, having experimental measures $I_s(\theta_1), I_s(\theta_2), \ldots, I_s(\theta_m)$, of the scattered light intensity for several angles, we wish to determine the corresponding values of the index of refraction n and the particle radius r. The value of n can be either real or complex. The intensity values vary widely relatively to the angle and thus it is better to work with the logarithm of the intensities. The objective function that is used is:

$$E_1 = \sum_{j=1}^{m} [i_s(\theta_j) - z(\theta_j, r, n)]^2, \tag{21}$$

where $i_s = \log(I_s)$ and the $z(\cdot)$ function is the logarithm of the theoretically computed intensity. If $n \in \mathbb{C}$, then the problem of minimizing E_1 is three-dimensional, otherwise, if $n \in \mathbb{R}$, it is two-dimensional. Several techniques have been used to solve the aforementioned problem. For real values of n, random search and multilevel single-linkage clustering have been applied. However, the function E_1 has numerous local minima and thus the initial

guess of the solution is of crucial importance. Since evolutionary techniques do not depend on the initial guess, they appear as a proper alternative to cope with this problem (Hodgson, 2000).

The problem of particle identification by light scattering, defined as the minimization of the function defined in Equation (21), was investigated using PSO. The simulated intensity functions, $I(\theta)$, were generated using the BHMIE routine of Bohren and Huffman (Bohren and Huffman, 1983). To these values, random amount of multiplicative noise with several variance values was added. In the first test problem, $n^* = 1.40$ and $r^* = 1.55\mu m$ were used (these values constituted the global minimum) and the values of n and r were bounded by $1 \leqslant n, r \leqslant 2$. The angular range was between 0 and 180 degrees, with data points taken at 9 degrees increments. The wavelength was set equal to $0.5145\mu m$, and the refractive index of the surrounding medium (water) was set equal to 1.336. The swarm's size was 30, the inertia weight w was gradually decreased from 1.2 towards 0.4, and $c_1 = c_2 = 0.5$. The maximum number of PSO iterations was 150, and the desired accuracy for all 30 runs was 10^{-5}. The results are reported in Table 11. The results presented in the tables are the success rate, the mean value of the Frobenius norm of the difference between the obtained and the actual global minimizer, the mean number of iterations and the mean number of function evaluations required.

Table 11. Results for the light scattering problem, for $n^* = 1.40$, $r^* = 1.55$.

Noise's Variance	Succ. Rate	Mean Dist.	Mean Iter.	Mean F. Ev.
0 (noiseless)	90%	0.0793	99.9	3027.0
0.1	95%	0.0369	101.6	3078.0
0.3	95%	0.0216	119.5	3615.0
0.5	65%	0.1627	130.3	3939.0

As can be seen, the addition of noise with small variance value increased the success rate of PSO by contributing to the alleviation of local minima, while the addition of noise with variance value around 0.5 decreased the overall performance of the algorithm. Similar are the results for $n^* = 1.65$ and $r^* = 4.0$, which is an area of the search space with many more local minima than the one in the previous experiment. The parameters were bounded by $1 \leqslant n \leqslant 2$, and $3 \leqslant r \leqslant 5$, and the swarm size was increased to 50, while the accuracy was relaxed to 10^{-3}. The results are reported in Table 12.

For additional results, regarding the ability of PSO to cope with noisy problems, for well-known and widely-used test functions, see (Parsopoulos and Vrahatis, 2001b; Parsopoulos and Vrahatis, 2001c).

Table 12. Results for the light scattering problem, for $n^* = 1.65$, $r^* = 4.0$.

Noise's Variance	Succ. Rate	Mean Dist.	Mean Iter.	Mean F. Ev.
0 (noiseless)	90%	0.1237	111.33	3370.0
0.1	95%	0.1601	70	2130.0
0.3	83%	0.4886	136.2	4530.0
0.5	94%	0.0008	89	2700.0

4.3 *Synopsis*

The experimental results indicate that in the presence of noise, additive or multiplicative, the PSO method is very stable and efficient. In fact, in several cases, the presence of noise seems to contribute to the avoidance of local minima and the detection of the global minimum of the objective function by the PSO. Even in the cases where the strength of the noise is large, PSO is able to move closer to the position of the global minimizer. Thus, PSO has the ability to cope with noisy environments effectively and in a stable manner.

5. Particle Swarm for Multiobjective Optimization

5.1 *Motivation and problem formulation*

Multiobjective Optimization (MO) problems are very common, especially in engineering applications, due to the multicriteria nature of most real-world problems. Design of complex hardware/software systems (Zitzler, 1999), atomic structure determination of proteins (Bush et al., 1995), potential function parameter optimization (Skinner and Broughton, 1995), x-ray diffraction pattern recognition (Paszkowicz, 1996), curve fitting (Ahonen et al., 1997) and production scheduling (Swinehart et al., 1996) are such applications, where two or more, sometimes competing and/or incommensurable, objective functions have to be minimized simultaneously. In contrast to the single-objective optimization case where the optimal solution is clearly defined, in MO there is a whole set of trade-offs which are known as *Pareto Optimal* solutions. These points are optimal solutions for the MO problem when all objectives are considered simultaneously.

Although the traditional gradient-based optimization techniques can be used to detect Pareto Optimal solutions, this approach is not popular due to two main drawbacks:

(a) the objectives have to be aggregated in one single objective function and,
(b) only one solution can be detected per optimization run.

The inherent difficulty to foreknow which aggregation of the objectives is the most proper in combination with the heavy computational cost of gradient-based techniques, implies the necessity for more efficient and rigorous methods. Evolutionary Algorithms (EA) seem to be especially suited to MO because they search for multiple Pareto Optimal solutions in a single run, they perform better global search of the search space and they have been proved to be robust and powerful search techniques (Zitzler, 1999).

Although there are many results of the PSO performance in single-objective optimization tasks, there are no well-known studies of the performance of PSO in MO problems. A study of the PSO's performance in MO tasks, is presented through experiments on well-known test problems (Parsopoulos and Vrahatis, 2002).

Let X be an n-dimensional search space, and $f_i(x)$, $i = 1, \ldots, k$, be k objective functions defined over X. Furthermore, let

$$g_i(x) \leqslant 0, \qquad i = 1, \ldots, m, \tag{22}$$

be m inequality constraints. Then, the MO problem can be defined as finding a vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)^\top \in X$ that satisfies the constraints, and optimizes (without loss of generality we consider only the minimization case) the vector function

$$\mathbf{f}(x) = \Big(f_1(x), f_2(x), \ldots, f_k(x)\Big)^\top. \tag{23}$$

The objective functions $f_i(x)$, may be conflicting with each other, thus, most of the time it is impossible to obtain for all objectives the global minimum at the same point. The goal of MO is to provide a set of solutions, to the aforementioned problem, that are Pareto Optimal.

Let $u = (u_1, \ldots, u_k)^\top$ and $v = (v_1, \ldots, v_k)^\top$, be two vectors. Then, u *dominates* v if and only if $u_i \leqslant v_i$, $i = 1, \ldots, k$, and $u_i < v_i$ for at least one component. This property is known as *Pareto Dominance* and it is used to define the Pareto Optimal points. Thus, a solution x of the MO problem is said to be Pareto Optimal if and only if there does not exist another solution y such that $\mathbf{f}(y)$ dominates $\mathbf{f}(x)$. The set of all Pareto Optimal solutions of an MO problem is called *Pareto Optimal Set* and it is denoted as $\mathcal{P}^*$. The set

$$\mathcal{PF}^* = \Big\{\big(f_1(x), \ldots, f_k(x)\big) \mid x \in \mathcal{P}^*\Big\}, \tag{24}$$

is called *Pareto Front*. A Pareto Front $\mathcal{PF}^*$ is called *convex* if and only if

$$\forall\, u, v \in \mathcal{PF}^*,\ \forall \lambda \in (0, 1),\ \exists w \in \mathcal{PF}^* :\ \lambda\|u\| + (1 - \lambda)\|v\| \geqslant \|w\|,$$

and it is called *concave* if and only if

$$\forall u, v \in \mathcal{PF}^*,\ \forall \lambda \in (0,1),\ \exists w \in \mathcal{PF}^* :\ \lambda\|u\| + (1-\lambda)\|v\| \leqslant \|w\|.$$

A Pareto Front can be convex, concave or partially convex and/or concave and/or discontinuous. The last three cases are the most difficult for most MO techniques.

5.2 *The Weighted Aggregation approach*

The *Weighted Aggregation* is the most common approach to MO problems. According to this approach, all the objectives, $f_i(x)$, are summed to a weighted combination

$$F = \sum_{i=1}^{k} w_i f_i(x), \tag{25}$$

where $w_i, i = 1, \ldots, k$, are non–negative weights. It is usually assumed that $\sum_{i=1}^{k} w_i = 1$. These weights can be either fixed or adapt dynamically during the optimization.

If the weights are fixed, then we are in the case of the *Conventional Weighted Aggregation* (CWA). Using this approach only a single Pareto Optimal point can be obtained per optimization run and a priori knowledge of the search space is required in order to determine the most proper weights (Jin et al., 2001). Thus, the search has to be repeated several times to obtain a desired number of Pareto Optimal points but this is not allowed in most optimization problems due to time limitations and heavy computational cost.

Other Weighted Aggregation approaches have been proposed to alleviate the problems that appear when using the CWA. Thus, for a two-objective MO problem we can modify the weights during the optimization according to the following equations

$$w_1(t) = \operatorname{sign}\Big(\sin(2\pi t/R)\Big), \tag{26}$$

$$w_2(t) = 1 - w_1, \tag{27}$$

where t is the iteration's index, and R is the weights' change frequency. This is the well-known *Bang-Bang Weighted Aggregation* (BWA) approach according to which the weights are changed abruptly due to the usage of the sign$(\cdot)$ function. Alternatively, the weights can be changed gradually according to the equations

$$w_1(t) = |\sin(2\pi t/R)|, \tag{28}$$

$$w_2(t) = 1 - w_1. \tag{29}$$

This approach is called *Dynamic Weighted Aggregation* (DWA) and the slow change of weights that takes place will force the optimizer to keep moving on the Pareto Front if it is convex, thus performing better than BWA. If the Pareto Front is concave then the performance of DWA and BWA is almost identical (Jin et al., 2001).

The three distinct approaches mentioned above have been used for experiments applying the PSO technique as optimizer, with $R = 100$ for the BWA and $R = 200$ for the DWA case respectively. The benchmark problems that were considered are taken from (Knowles and Corne, 2000; Zitzler et al., 2000):

TEST PROBLEM 5.1, (convex, uniform Pareto Front):

$$\begin{aligned} f_1 &= \frac{1}{n}\sum_{i=1}^{n} x_i^2, \\ f_2 &= \frac{1}{n}\sum_{i=1}^{n} (x_i - 2)^2. \end{aligned}$$

TEST PROBLEM 5.2, (convex, non–uniform Pareto Front):

$$\begin{aligned} f_1 &= x_1, \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ f_2 &= g\left(1 - \sqrt{f_1/g}\right). \end{aligned}$$

TEST PROBLEM 5.3, (concave Pareto Front):

$$\begin{aligned} f_1 &= x_1, \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ f_2 &= g\left(1 - (f_1/g)^2\right). \end{aligned}$$

TEST PROBLEM 5.4, (neither purely convex nor purely concave Pareto Front):

$$\begin{aligned} f_1 &= x_1, \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ f_2 &= g\left(1 - \sqrt[4]{f_1/g} - (f_1/g)^4\right). \end{aligned}$$

TEST PROBLEM 5.5, (Pareto Front that consists of separated convex parts):

$$\begin{aligned} f_1 &= x_1, \\ g &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ f_2 &= g\left(1 - \sqrt{f_1/g} - (f_1/g)\ \sin(10\pi f_1)\right). \end{aligned}$$

Each test problem is denoted by the corresponding function F (e.g., the Test Problem 5.1 is denoted by F_1). Although simple and containing only two objectives, these test functions comprise difficult MO problems (especially F_3, F_4 and F_5) due to the shape of the Pareto Front (purely or partially concave, discontinuous etc.). In order to have comparable results in detecting the Pareto Front of the benchmark problems with the results provided in (Jin et al., 2001) for the EA case, we used the pseudocode provided in (Jin et al., 2001) to build and maintain the archive of Pareto solutions and we performed all simulations using the same parameter values. Thus, 150 iterations of PSO were performed for each problem, with $x \in [0, 1]^2$. The PSO parameters were fixed $c_1 = c_2 = 0.5$, and the inertia w was gradually decreased from 1 toward 0.4. The size of the swarm depended on the problem but it never exceeded 40.

The first experiments were performed using the CWA approach with a small swarm of 10 particles. The desired number of Pareto Optimal points was 20 for the functions F_1, F_2, F_3, and 40 for the function F_4. Thus we had to run the algorithm 20 times for the first three functions and 40 times for the fourth. The obtained Pareto Fronts are exhibited in Figure 6. The computational cost was low and convergence rates were high (less than 2 minutes were needed for each function).

The results for the experiments done using the BWA and DWA approaches are exhibited in Figures 7–10.

It is clear that PSO succeeds in capturing the shape of the Pareto Front in each case. The results are better when using the DWA approach except for the case of the concave Pareto Front of the function F_3, at which the BWA approach performed better. The swarm's size was equal to 20 for all simulations, except for the case of function F_5 which was equal to 40.

The MO problems can be alternatively solved using population-based non-Pareto approaches instead of Weights Aggregating approaches. One such approach is the VEGA (Vector Evaluated Genetic Algorithm) developed by Schaffer (Schaffer, 1985). In the next section, a modification of the PSO algorithm using ideas from the VEGA is used to solve MO problems.

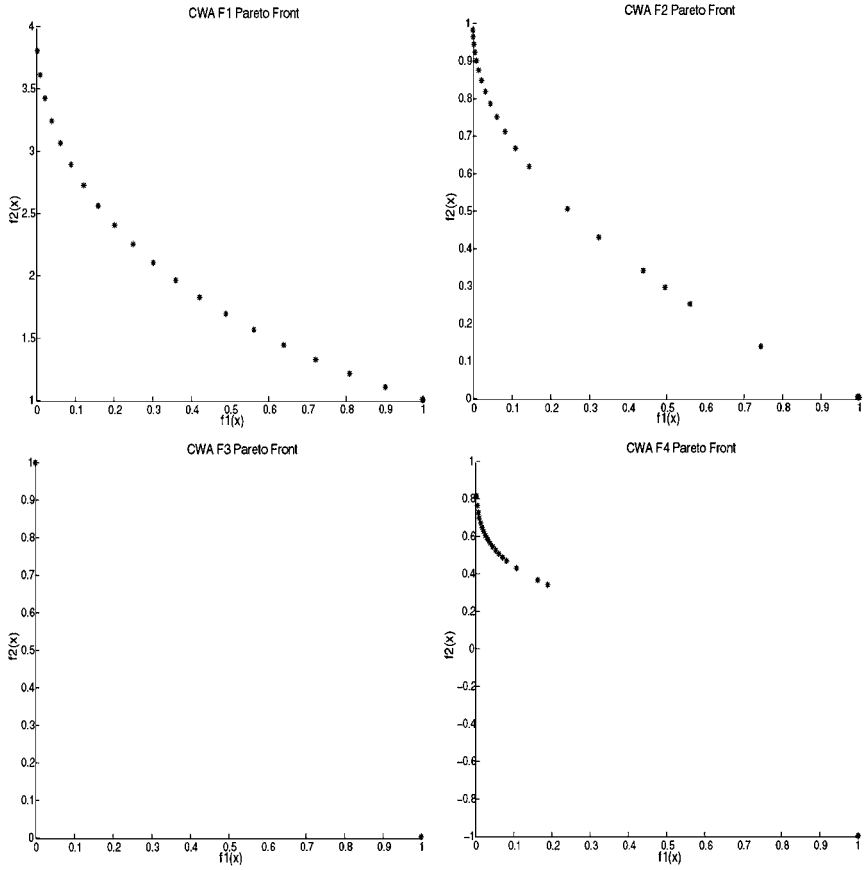

Figure 6. CWA approach results.

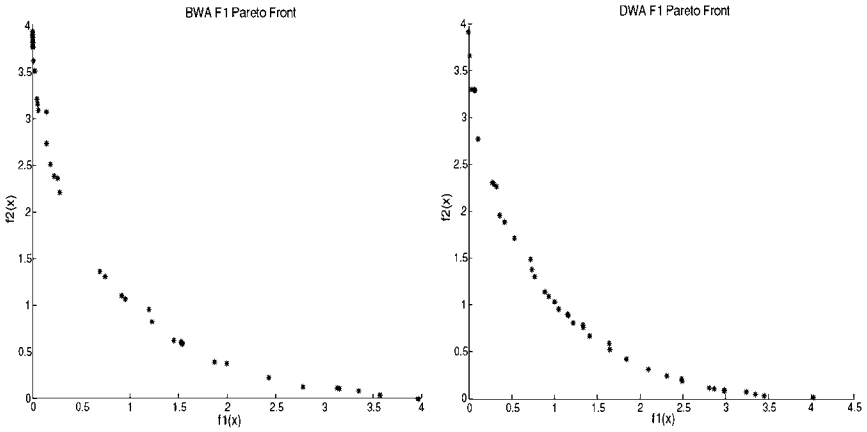

Figure 7. BWA and DWA approaches results for the function F_1.

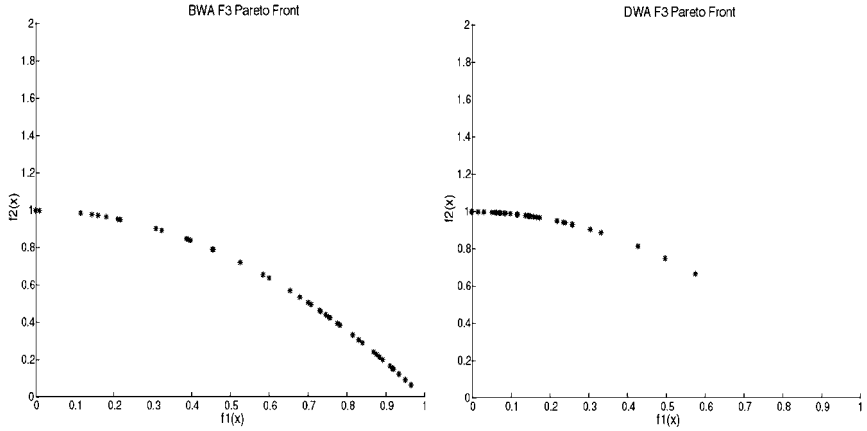

Figure 8. BWA and DWA approaches results for the function F_3.

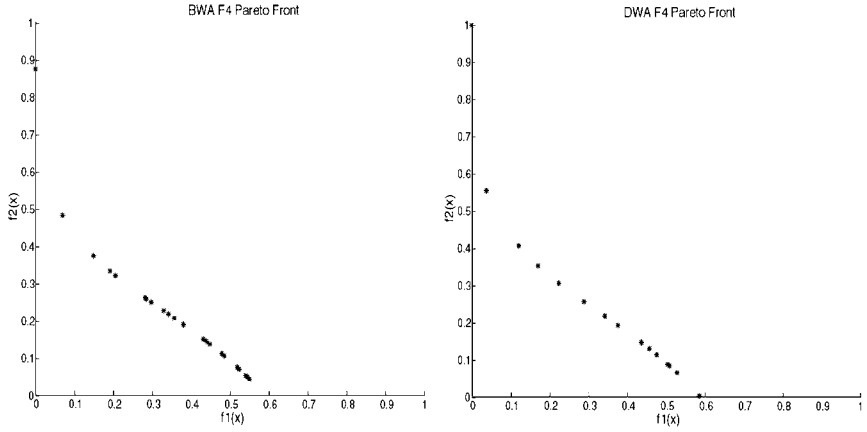

Figure 9. BWA and DWA approaches results for the function F_4.

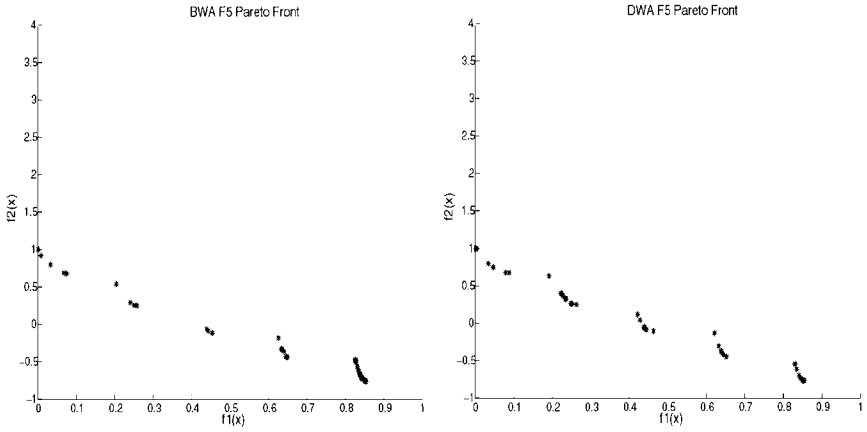

Figure 10. BWA and DWA approaches results for the function F_5.

5.3 *A population-based non-pareto approach (VEPSO)*

According to the VEGA approach, fractions of the next generation, or sub-populations, are selected from the old generation according to each of the objectives, separately. After shuffling all these sub-populations together, crossover and mutation are applied as usual to generate the new population (Schaffer, 1985).

The main ideas of VEGA were adopted and modified to fit the PSO framework and develop the VEPSO algorithm (Parsopoulos and Vrahatis, 2002). Thus, we used two swarms to solve the five benchmark problems F_1–F_5. Each swarm was evaluated according to one of the objectives but the change of velocities was performed based on information coming from the other swarm. More specifically, the best particle of the second swarm was used for calculation of the new velocities of the first swarm's particles, using Equation (4), and vice versa. In a second version, we used the best positions of the second swarm, in addition to the best particle of the second swarm, for the evaluation of the velocities of the first swarm and vice versa. The obtained Pareto Fronts for all benchmark problems are exhibited in Figures 11–15. The left part of each figure is the Pareto Front obtained using only the best particle of the other swarm while the right part is obtained using both the best particle and the best previous positions of the other swarm. Except for the case of Function F_2, there is no significant difference between the two approaches. For each experiment, two swarms of size 20 were used and the algorithm ran for 200 iterations.

5.4 *Synopsis*

A study of the performance of the PSO technique in MO problems has been presented. The PSO method has been used and efficiently solved well-known and widely used test problems, including difficult cases for MO techniques, such as concavity and/or discontinuouity of the Pareto Front, although only two objectives were involved in each problem. We used low dimensional objectives in order to investigate the simplest cases first. Besides, it is a general feeling that two objectives are sufficient to reflect essential aspects of MO (Zitzler et al., 2000). Even using very small swarms resulted in very promising results. In addition to the Weighted Aggregation cases, a modified version of PSO that resembles the VEGA ideas has been developed and applied on the same problems with promising results also.

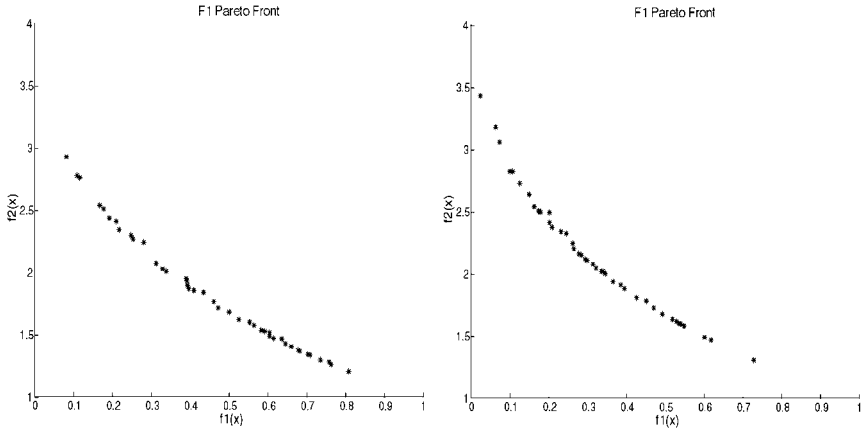

Figure 11. Results for the function F_1.

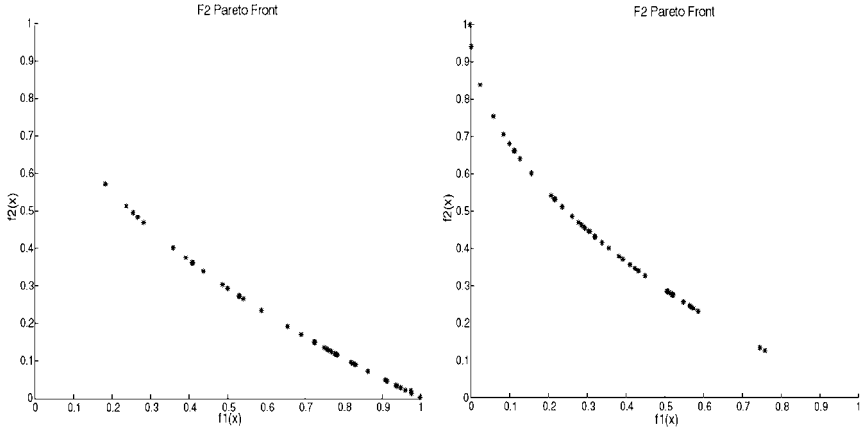

Figure 12. Results for the function F_2.

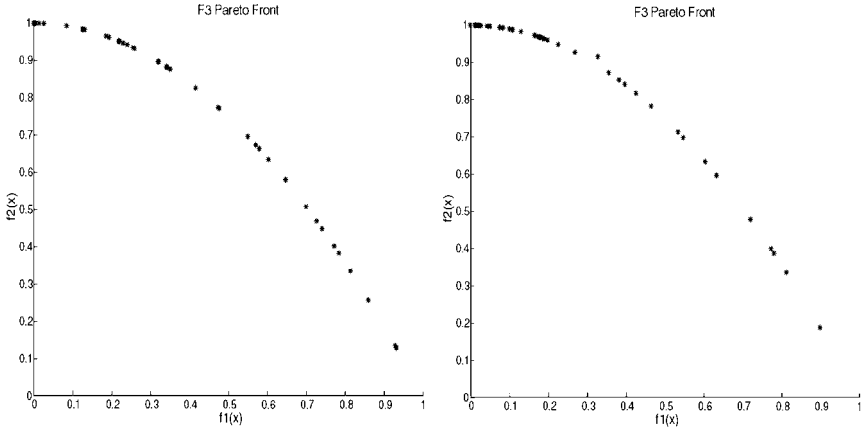

Figure 13. Results for the function F_3.

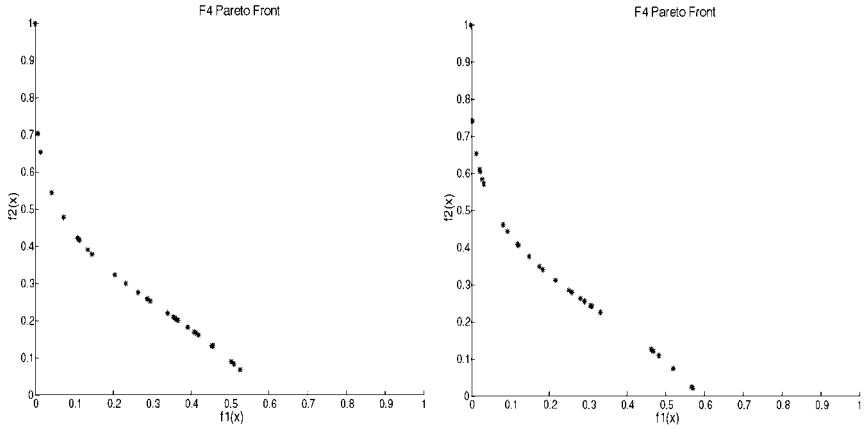

Figure 14. Results for the function F_4.

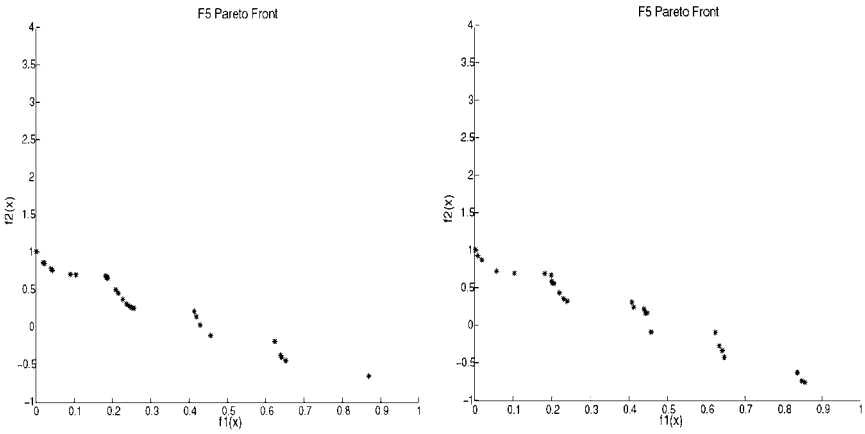

Figure 15. Results for the function F_5.

6. Solving ℓ_1 norm errors-in-variables problems using Particle Swarm Optimization

6.1 *Motivation and problem formulation*

Data fitting is a central problem in Approximation Theory and can be met in many engineering applications. A fundamental issue in such problems is the choice of the measure of quality of the fit which should be used. Of course, this choice depends on the underlying problem but, in general, the ℓ_p norms are used. Given an m-dimensional vector, $x \in \mathbb{R}^m$, the ℓ_p norm of

x is defined as

$$\|x\|_p = \Big(\sum_{i=1}^{m} |x_i|^p\Big)^{1/p}, \quad 1 \leqslant p < \infty,$$
$$\|x\|_\infty = \max_{1 \leqslant i \leqslant m} |x_i|.$$

The two most commonly met norms are the ℓ_1 and ℓ_2, which are defined as:

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|,$$
$$\|x\|_2 = \Big(\sum_{i=1}^{m} |x_i|^2\Big)^{1/2}.$$

The ℓ_1 norm plays an important role in data fitting, especially when there are large errors or "wild" points in the data. This is due to the assignment of smaller weights to these points than the weights assigned by other more popular measures, such as the ℓ_2 norm (least squares) (Watson, 1998). The case considered here is the one where all variables have errors and the explicit or implicit models, which are used for fitting the data, are in general nonlinear. In the case of implicit models, constraints appear and a common technique to solve them involves solving a sequence of linear subproblems, whose solutions are constrained to lie in a trust region (Watson, 1997; Watson, 1998; Watson and Yiu, 1991). Thus, Trust Region methods, such as the one presented in (Watson, 1997; Watson and Yiu, 1991), are used to solve the problem, exploiting the structure of the subproblems, which can be solved as equality bounded variable linear programming problems.

Suppose that we have a set of observations $y_i \in \mathbb{R}$, $i = 1, \ldots, n$, at points $x_i \in \mathbb{R}^t$, $i = 1, \ldots, n$, and a model for the underlying relationship

$$y = F(x, \alpha), \tag{30}$$

where $\alpha \in \mathbb{R}^p$ is a vector of free parameters. We are interested in finding the values of the free parameters that give the best fit. If it is assumed that both y_i and x_i contain significant errors, r_i and ε_i respectively, then the model can be written as

$$y_i + r_i = F(x_i + \varepsilon_i, \alpha), \quad i = 1, 2, \ldots, n. \tag{31}$$

A proper vector α has to be chosen, in order to minimize the errors r_i and ε_i for $i = 1, 2, \ldots, n$.

Although the least squares norm (ℓ_2) is widely used in data fitting modeling, and it is well studied, there are some drawbacks, becoming evident whenever "wild" points or large errors are introduced into the data set. This

is due to the assignment of excessive weights to these points, which results in non-satisfactory estimators. The alternative, which is commonly used in cases of this sort, is the ℓ_1 norm and to this end several algorithms have been developed (Watson, 1997; Watson and Yiu, 1991).

Although Equation (30) is in explicit form, it can always be written in the more general implicit form

$$f(x, \alpha) = 0, \tag{32}$$

where $x \in \mathbb{R}^k$, $k = t + 1$, represents all the variables. The corresponding model for Equation (32) is

$$f(x_i + \varepsilon_i, \alpha) = 0, \quad i = 1, 2, \ldots, n, \tag{33}$$

where ε_i represents now the vector of the errors for all the variables. Thus, the problem under consideration is

$$\min_{\alpha,\varepsilon} \ \|\varepsilon\|_1, \tag{34}$$

$$\text{subject to} \ \ f(x_i + \varepsilon_i, \alpha) = 0, \quad i = 1, \ldots, n, \tag{35}$$

which is a constrained problem, in contradiction to the case of the explicit model of Equation (31), where the corresponding problem is unconstrained

$$\min_{\alpha,\varepsilon} \{\|r\|_1 + \|\varepsilon\|_1\}, \tag{36}$$

with $r_i = F(x_i + \varepsilon_i, \alpha) - y_i$, $i = 1, \ldots, n$.

There are several ways to solve the constrained minimization problem. The Kuhn-Tucker theory provides necessary conditions for the solution of such problems, but a subgradient has to be used (Hirriart, 1978). Alternatively, this can be solved by using an ℓ_1 penalty function (Fletcher, 1987; Watson, 1997). In this case, the problem under consideration is

$$\min_{\alpha,\varepsilon} \{\|f\|_1 + \nu\|\varepsilon\|_1\}, \tag{37}$$

where $\nu > 0$.

In (Watson, 1997; Watson and Yiu, 1991), a Trust Region type algorithm which involves Simplex steps for solving the linear subproblems has been considered and it has been applied to various test problems. Although the solutions resulted from that algorithm are acceptable, there is another approach to tackle the problem. This approach is more straightforward, easy to implement as well as economic in function evaluations, and it uses the PSO technique to solve the problem of Equation (37) (Parsopoulos et al., 2001d).

6.2 *Experimental results*

The models that we consider in this section are (or are assumed to be) implicit and they are the same as those reported in (Watson, 1997; Watson and Yiu, 1991), in order to obtain comparable results. All the models are simple and some of them are actually explicit, but yet treated as implicit. The initial approximation for the vector ε is usually set equal to zero and this is the approach we followed too. Concerning parameter ν, the fixed value 0.1 was used, although it is more proper for many problems to gradually decrease its value to force convergence of the penalty function to zero. Parameter α had a different initial value for each model and the desired accuracy for all unknowns was 10^{-3}.

The values of the PSO parameters were $c_1 = c_2 = 2$ (default values) and w was gradually decreased from 1.2 toward 0.1. The maximum allowable number of PSO iterations was 500 and the size of the swarm was fixed and equal to 150 for all experiments.

In all the tables, the retrieved optimal values for α and for the ℓ_1 norm of ε, denoted as α^* and $\|\varepsilon^*\|_1$ respectively, are reported.

TEST PROBLEM 6.1, (Watson, 1997):
The first model considered was the *hyperbolic* model introduced by Britt and Luecke in (Britt and Luecke, 1973) with

$$f = x_1^2 x_2^2 - \alpha.$$

The data used is given in Table 13 and the initial value of α was taken equal to 100. The results for both the Trust Region and the PSO methods are given in Table 14. PSO found the solution after 130 iterations.

Table 13. Data set 1, used in experiments 6.1 and 6.3.

x_1	0.1	0.9	1.8	2.6	3.3	4.4	5.2	6.1	6.5	7.4
x_2	5.9	5.4	4.4	4.6	3.5	3.7	2.8	2.8	2.4	1.5

Table 14. Results of Experiment 6.1.

	Trust Region	PSO
α^*	133.402	139.688
$\|\varepsilon^*\|_1$	7.255	3.446

TEST PROBLEM 6.2, (Watson, 1997):
The second model considered has

$$f = \sum_{i=0}^{2} \alpha_i x_1^i - x_2,$$

and the data used is given in Table 15. Initially, $\alpha = (1, 1, 1)^\top$ was taken and PSO found the solution after 158 iterations. The results are exhibited in Table 16.

Table 15. Data set 2, used in experiments 6.2, 6.4 and 6.5.

x_1	0.05	0.11	0.15	0.31	0.46	0.52	0.70	0.74	0.82	0.98	1.17
x_2	0.956	0.89	0.832	0.717	0.571	0.539	0.378	0.370	0.306	0.242	0.104

Table 16. Results of Experiment 6.2.

	Trust Region	PSO
α^*	$(1.001, -1.038, 0.232)^\top$	$(0.997, -1.001, 0.201)^\top$
$\|\varepsilon^*\|_1$	0.112	0.001

TEST PROBLEM 6.3, (Watson, 1997):
This is another model with

$$f = \sum_{i=0}^{3} \alpha_i x_1^i - x_2,$$

and the data used is given in Table 13, after setting the first value of x_1 equal to 0.0. Initially, $\alpha = 0$ was taken, and PSO found the solution after 160 iterations. The results are exhibited in Table 17.

Table 17. Results of Experiment 6.3.

	Trust Region	PSO
α^*	$(5.9, -0.839, 0.142, -0.015)^\top$	$(5.91, -0.624, 0.057, -0.006)^\top$
$\|\varepsilon^*\|_1$	1.925	0.298

TEST PROBLEM 6.4, (Watson, 1997):
Here we had the nonlinear model

$$f = \alpha_1 + \frac{\alpha_2}{x_1 + \alpha_3} - x_2,$$

and the data used is given in Table 18. The initial approximation was $\alpha = (-4, 5, -4)^\top$ and PSO found the solution after 282 iterations. The results are exhibited in Table 18.

Table 18. Results of Experiment 6.4.

	Trust Region	PSO
α^*	$(-1.934, 7.761, -2.638)^\top$	$(0.503, -17.009, -482.805)^\top$
$\|\varepsilon^*\|_1$	0.109	0.165

TEST PROBLEM 6.5, (Watson, 1997; Watson and Yiu, 1991):
This is a growth model with

$$f = \alpha_1 + \alpha_2 e^{-x_1} - x_2,$$

and the data used is given in Table 15. The initial approximation was $\alpha = (0, 0)^\top$ and PSO found the solution after 490 iterations. The results are exhibited in Table 19.

Table 19. Results of Experiment 6.5.

	Trust Region	PSO
α^*	$(-0.225, 1.245)^\top$	$(-0.227, 1.247)^\top$
$\|\varepsilon^*\|_1$	0.169	0.005

TEST PROBLEM 6.6, (Watson and Yiu, 1991):
This is the *ordinary rational function* model with

$$f = \frac{\alpha_1}{x_1 - \alpha_2} - x_2,$$

and the data was generated according to the scheme

$$\begin{aligned} x_i &= 0.01 + 0.05\ (i - 1), \\ y_i + r_i &= 1 + x_i + (x_i + \varepsilon_i)^2,\ i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in $(-0.15, 0.15)$, and ε_i are uniformly distributed in $(-0.05, 0.05)$. The initial approximation was $\alpha = (1, 1)^\top$ and PSO found the solution after 211 iterations. The results are exhibited in Table 20.

Table 20. Results of Experiment 6.6.

	Trust Region	PSO
α^*	$(0.978, 1.006)^\top$	$(1.0017, 1.0002)^\top$
$\|\varepsilon^*\|_1$	1.458	1.011

TEST PROBLEM 6.7, (Watson and Yiu, 1991):
This is the *simple decay curve*, which describes the decay of a radioactive substance,

$$f = \alpha_1 - \alpha_2\ e^{-\alpha_3 x_1} - x_2.$$

The data was generated according to the scheme

$$\begin{aligned} x_i &= 0.02\,(i-1), \\ y_i + r_i &= 3 - e^{-(x_i+\varepsilon_i)},\ i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in $(-0.15, 0.15)$, and ε_i are uniformly distributed in $(-0.05, 0.05)$. The initial approximation was $\alpha = (3, 1, 1)^\top$ and PSO found the solution after 315 iterations. The results are exhibited in Table 21.

Table 21. Results of Experiment 6.7.

	Trust Region	PSO
α^*	$(2.528, 0.918, 4.396)^\top$	$(2.709, 1.007, 0.975)^\top$
$\|\varepsilon^*\|_1$	2.564	2.231

TEST PROBLEM 6.8, (Watson and Yiu, 1991):
This is the *logistic curve*, which describes the decay of a radioactive substance,

$$f = \alpha_1 - \log\left(1 + e^{-(\alpha_2+\alpha_3 x_1)}\right) - x_2.$$

The data was generated according to the scheme

$$\begin{aligned} x_i &= 0.02\,(i-1), \\ y_i + r_i &= 3 - \log\left(1 + e^{-(1+x_i+\varepsilon_i)}\right),\ i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in $(-0.15, 0.15)^\top$, and ε_i are uniformly distributed in $(-0.05, 0.05)^\top$. The initial approximation was again $\alpha = (3, 1, 1)^\top$ and PSO found the solution after 336 iterations. The results are exhibited in Table 22.

Table 22. Results of Experiment 6.8.

	Trust Region	PSO
α^*	$(2.812, 0.487, 16.033)^\top$	$(2.722, 0.851, 1.164)^\top$
$\|\varepsilon^*\|_1$	2.348	0.471

As exhibited in all tables, PSO outperformed the Trust Region algorithm in almost all of the cases. It was able to detect superior solutions even in vicinities of the search space that were far removed from the vicinity of solutions obtained by the Trust Region method. Although, the same initial vector

for the parameter α, as in (Watson, 1997), was used, in further experiments, we obtained the same good solutions starting from arbitrarily chosen initial points. The number of iterations performed by the PSO method to detect the optimum solutions was surprisingly small.

6.3 *Synopsis*

The ability of the PSO to tackle data fitting models was investigated on well-known implicit, as well as explicit models. The results obtained by our approach have been contrasted with the corresponding results presented in (Watson, 1997; Watson and Yiu, 1991), where a Trust Region method was used instead.

The results reported, are indeed very promising and clarify that PSO can solve efficiently such problems. In many cases, the problem becomes easier due to the ability of the method to detect good solutions starting from several initial points, in contrast to the Trust Region method whose behavior is heavily dependent on the starting point. Furthermore, the solutions are obtained after a relatively small number of iterations. An important role is played by the penalty parameter ν. Usually, the convergence rates of both techniques and the quality of results depend on the value of ν.

7. Particle Swarm Optimization for Minimax problems

7.1 *Motivation and problem formulation*

Minimax problems are encountered in numerous optimal control, engineering design, discrete optimization, Chebyshev approximation and game theory applications (Demyanov and Molozemov, 1974; Du and Pardalos, 1995; Polak, 1987; Waren et al., 1967; Zuhe et al., 1990). Specifically, in Chebyshev approximation, given a function $g : Y^{(0)} \subset \mathbb{R}^m \to \mathbb{R}$, the Chebyshev approximate p_z of g in p_n solves the following minimax problem (Zuhe et al., 1990):

$$\min_{z} \max_{y \in Y^{(0)}} (g(y) - p_z(y))^2.$$

In game theory, a game is a triple (Y, Z, k) where Y, Z denote the spaces of strategies for player I and II, respectively, and k is a real-valued pay-off function of $y \in Y$ and $z \in Z$. Under natural conditions, the optimal strategies for both players solve the saddle point problem (Zuhe et al., 1990):

$$\min_{z \in Z} \max_{y \in Y} k(y, z) = \max_{y \in Y} \min_{z \in Z} k(y, z).$$

Additionally, in many engineering design problems, one is interested in minimizing the largest eigenvalue of an $n \times n$ symmetric matrix-valued function $A(y)$ of a variable y in $\mathbb{R}^n$. Thus, if $\lambda_i(y)$, $i = 1, \ldots, n$, is the i-th eigenvalue of $A(y)$ and by setting $f(i, n) = \lambda_i(y)$, then the following minimax problem is obtained (Zuhe et al., 1990):

$$\min_{y \in Y^{(0)}} \max_{i=1,\ldots,n} f(i, y).$$

Another example is the minimization of error in the manufacturing of electronic parts, with a prespecified level of tolerance. Specifically, suppose that when a state z is specified, the process actually produces the state $y + z$ for some y in the tolerance set Z and let $\theta(y + z)$ measure the resulting distortion. Since y is not known in advance, the worst–case distortion should be minimized, leading to the minimax problem (Zuhe et al., 1990):

$$\min_{z \in Z} \max_{y \in Y} \theta(y + z).$$

Numerous other important applications involve solving minimax problems, justifying the ongoing interest for development of techniques that can cope efficiently with it.

In general, the minimax problem can be stated as

$$\min_{x} f(x), \tag{38}$$

where

$$f(x) = \max_{i=1,\ldots,m} f_i(x), \tag{39}$$

with $f_i(x) : S \subset \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$.

The nature of the objective function $f(x)$ may pose difficulties in the process of solving minimax problems. Specifically, at points where $f_j(x) = f(x)$ for two or more values of $j \in \{1, \ldots, m\}$, the first partial derivatives of $f(x)$ are discontinuous, even if all functions $f_i(x)$, $i = 1, \ldots, m$, have continuous first partial derivatives. This difficulty cannot be addressed directly by the well-known and widely used gradient–based methods, and several techniques have been proposed to cope with it (Charalambous and Conn, 1978; Murray and Overton, 1980; Osborne and Watson, 1969; Waren et al., 1967).

Smoothing methods have proved to be very useful in solving the minimax problem. Following this approach, a smoothing function, which is sometimes called the *Exponential Penalty Function* or *Aggregate Function*, is used to approximate the objective function $f(x)$ (Bertsekas, 1976a; Bertsekas, 1977; Kort and Bertsekas, 1972; Li, 1991). The smoothing function is minimized

using a gradient-based technique with line-search. Line-search is used to ensure the global convergence of the algorithm.

Recently, a new smoothing function has been proposed in (Xu, 2001), and a quadratic approximation of it is solved by using a gradient-based method with line-search.

In this section, the performance of the PSO with regard to minimax problems is reported (Laskari et al., 2002a), and for some cases its performance is compared with the performance of the Smoothing algorithm proposed in (Xu, 2001).

7.2 *The Smoothing technique*

Recently, an interesting Smoothing Technique has been proposed in (Xu, 2001) for solving minimax problems. It exploits the smoothing function

$$f(x, \mu) = \mu \ \ln \sum_{i=1}^{m} \exp\left(\frac{f_i(x)}{\mu}\right), \tag{40}$$

to approximate the objective function $f(x)$. This function is considered a good approximation of $f(x)$ in the sense that

$$f(x) \leqslant f(x, \mu) \leqslant f(x) + \mu \ln m,$$

for $\mu > 0$, as it is mentioned in (Xu, 2001).

The proposed method solves a quadratic approximation of $f(x, \mu)$ for decreasing values of μ. The global convergence of the algorithm is ensured using Armijo's line-search procedure (Polak, 1997; Vrahatis et al., 2000). The method bears similarity to the recently proposed smoothing methods for Complementarity Problems and for Mathematical Programming with Equilibrium Constraints (Burke and Xu, 2000; Facchinei et al., 1999).

A point x^* is a stationary point to the minimax problem defined in Equation (38), if there exists a vector $y^* = (y_1^*, \ldots, y_m^*)$, such that

$$\sum_{j=1}^{m} y_j^* \ \nabla f_j(x^*) = 0, \tag{41}$$

with

$$y_j^* \geqslant 0, \ \ j = 1, \ldots, m, \tag{42}$$

$$\textstyle\sum_{j=1}^{m} y_j^* = 1, \tag{43}$$

$$y_j^* = 0, \ \text{if } f_j(x^*) < \max\{f_1(x^*), \ldots, f_m(x^*)\}. \tag{44}$$

Related to the above, the following two theorems have been proved:

Theorem 1. (Demyanov and Molozemov, 1974; Xu, 2001) *If x^* is a local minimum to the problem defined in Equation (38), then it is a stationary point*

that satisfies Equations (41)–(44). Conversely, assume that $f(x)$ is convex, then if x^* is a stationary point, x^* is a global minimum to the minimax problem.

Theorem 2. (Peng and Lin, 1999; Xu, 2001) *Suppose that $f_i(x)$, $i = 1, \ldots, m$, are twice continuous differentiable, $f(x)$ and $f(x, \mu)$ are defined in Equations (39) and (40) respectively, then*
(i) $f(x, \mu)$ is increasing with respect to μ, and

$$f(x) \leqslant f(x, \mu) \leqslant f(x) + \mu \ln m.$$

(ii) $f(x, \mu)$ is twice continuous differentiable for all $\mu > 0$, and

$$\nabla_x f(x, \mu) = \sum_{i=1}^{m} \lambda_i(x, \mu) \nabla f_i(x), \tag{45}$$

$$\nabla_x^2 f(x, \mu) = \sum_{i=1}^{m} \left(\lambda_i(x, \mu) \nabla^2 f_i(x) + \frac{1}{\mu} \lambda_i(x, \mu) \nabla f_i(x) \nabla f_i(x)^\top \right) - \frac{1}{\mu} \left(\sum_{i=1}^{m} \lambda_i(x, \mu) \nabla f_i(x) \right) \left(\sum_{i=1}^{m} \lambda_i(x, \mu) \nabla f_i(x) \right)^\top, \tag{46}$$

where

$$\lambda_i(x, \mu) = \frac{\exp\left(\frac{f_i(x)}{\mu}\right)}{\sum_{j=1}^{m} \exp\left(\frac{f_j(x)}{\mu}\right)} \in (0, 1), \sum_{i=1}^{m} \lambda_i(x, \mu) = 1. \tag{47}$$

(iii) For any $x \in \mathbb{R}^n$ and $\mu > 0$, it holds that $0 \leqslant f'_\mu(x, \mu) \leqslant \ln m$.

An extended theoretical presentation of all the aforementioned aspects, as well as the convergence properties of the Smoothing Technique, can be found in (Xu, 2001).

7.3 *Experimental results*

The performance of the PSO was studied on several test problems defined in (Charalambous and Conn, 1978; Lukšan and Vlček, 2000; Schwefel, 1995; Xu, 2001). For all experiments, the maximum number of PSO iterations was set to 500, the desired accuracy was equal to 10^{-8}, the inertia weight w was gradually decreased from 1.2 towards 0.4, and $c_1 = c_2 = 0.5$. For each test problem, 25 experiments were performed, and the obtained results are the total number of PSO successes in finding the global minimum, as well as the mean number of required iterations and function evaluations.

First, the functions given in Examples 1 and 2 in (Xu, 2001), were considered (we denote them as $F1$ and $F2$). They are both 2-dimensional, involving 3 functions $f_i(x)$, and they are defined as follows:

TEST PROBLEM 7.1, (Xu, 2001):

$$\min_x F1(x),$$

where

$$\begin{aligned} F1(x) &= \max\{f_i(x)\},\ i = 1, 2, 3, \\ f_1(x) &= x_1^2 + x_2^4, \\ f_2(x) &= (2 - x_1)^2 + (2 - x_2)^2, \\ f_3(x) &= 2\exp(-x_1 + x_2), \end{aligned}$$

TEST PROBLEM 7.2, (Xu, 2001):

$$\min_x F2(x),$$

where

$$\begin{aligned} F2(x) &= \max\{f_i(x)\},\ i = 1, 2, 3, \\ f_1(x) &= x_1^4 + x_2^2, \\ f_2(x) &= (2 - x_1)^2 + (2 - x_2)^2, \\ f_3(x) &= 2\exp(-x_1 + x_2). \end{aligned}$$

The initial swarm was uniformly distributed in $[-5, 5]^2$ and its size was 20 in both cases. The results are exhibited in Table 23.

Table 23. Success rates, mean number of iterations and mean number of function evaluations, for the functions $F1$ and $F2$.

Function	Succ.	Mean Iter.	Mean F. Ev.
F1	25/25	331.2	6644.0
F2	25/25	297.36	5967.2

Consider the nonlinear programming problem:

$$\begin{aligned} &\min F(x), \\ &\text{subject to } g_i(x) \geqslant 0, i = 1, \ldots, m. \end{aligned}$$

Then, the following minimax problem can be constructed:

$$\min_x f(x),$$

where

$$\begin{aligned} f(x) &= \max_{1\leqslant i\leqslant m} f_i(x), \\ f_1(x) &= F(x), \\ f_i(x) &= F(x) - \alpha_i g_i(x), \\ \alpha_i &> 0, \end{aligned}$$

for $2 \leqslant i \leqslant m$.

It has been proved that for sufficiently large α_i the optimum of the minimax function coincides with that of the nonlinear programming problem (Bandler and Charalambous, 1974). Following this approach, the two minimax problems in Examples 3 and 4 in (Xu, 2001) were constructed (we denote them as $F3$ and $F4$ respectively) and they are defined as follows:

TEST PROBLEM 7.3, (Xu, 2001):

$$\begin{aligned} F3(x) &= x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4, \\ g_2(x) &= -x_1^2 - x_2^2 - x_3^3 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8, \\ g_3(x) &= -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10, \\ g_4(x) &= -x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 + 5, \end{aligned}$$

and

TEST PROBLEM 7.4, (Xu, 2001):

$$\begin{aligned} F4(x) &= (x_1 - 10)^2 + 5(x_2 - 12)^2 + 3(x_4 - 11)^2 + x_3^4 + 10x_5^6 + \\ &\quad +7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \\ g_2(x) &= -2x_1^2 - 3x_3^4 - x_3 - 4x_4^2 - 5x_5 + 127, \\ g_3(x) &= -7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 + 282, \\ g_4(x) &= -23x_1 - x_2^2 - 6x_6^2 + 8x_7 + 196, \\ g_5(x) &= -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7. \end{aligned}$$

These problems are four and seven dimensional with $m = 4$ and $m = 5$ respectively, and a PSO with a swarm of 20 particles was used to solve both. The initial swarm was again uniformly distributed in the range $[-5, 5]^D$, were D is the corresponding dimension. The results are exhibited in Table 24.

Table 24. Success rates, mean number of iterations and mean number of function evaluations, for the functions $F3$ and $F4$.

Function	Succ.	Mean Iter.	Mean F. Ev.
F3	25/25	365.56	7331.2
F4	25/25	335.08	6721.6

At this point it is important to note that the solutions found by PSO satisfy all the constraints of the nonlinear programming problem, while the solutions obtained using the Smoothing Technique do not satisfy the constraints $g_2(x)$ for $F3$, and $g_5(x)$ for $F4$, respectively.

Furthermore, two minimax problems defined in (Schwefel, 1995) were considered. The first is 2–dimensional (we will refer it as $F5$) and it is defined as:

TEST PROBLEM 7.5, (Schwefel, 1995):

$$\min \max\{|x_1 + 2x_2 - 7|, |2x_1 + x_2 - 5|\}.$$

The swarm's size for this problem was 20 and the initial swarm was uniformly distributed in $[-5, 5]^2$. The second problem was considered in dimension 10 (we will refer it as $F6$), and it is defined as:

TEST PROBLEM 7.6, (Schwefel, 1995):

$$\min \max\{|x_i|\}, \quad 1 \leqslant i \leqslant 10.$$

The swarm's size for this problem was 100 and the initial swarm was uniformly distributed in $[-5, 5]^{10}$. The results are exhibited in Table 25.

Table 25. Success rates, mean number of iterations and mean number of function evaluations, for the functions $F5$ and $F6$.

Function	Succ.	Mean Iter.	Mean F. Ev.
F5	25/25	285.44	5728.8
F6	25/25	320.8	32180.0

TEST PROBLEMS 7.6–7.13, (Lukšan and Vlček, 2000):
Furthermore, seven test problems were selected from (Lukšan and Vlček, 2000) to investigate further the performance of PSO. The name of each problem, its dimension, and the number of $f_i(x)$ functions involved are reported in Table 26.

Table 26. Dimension and number of functions $f_i(x)$ for the test problems.

Function	Dimension	# $f_i(x)$
CB2	2	3
WF	2	3
SPIRAL	2	2
EVD52	3	6
POLAK6	4	4
WONG1	7	5
WONG2	10	9

For all problems mentioned in Table 26, the swarm's size was 20 except for the two WONG problems where it was set equal to 70 and 100 respectively. The initial swarm was uniformly distributed in $[-5, 5]^D$, where D is the corresponding dimension, except for the WONG2 case were it was distributed in $[-3, 3]^{10}$. The results are exhibited in Table 27.

Table 27. Success rates, mean number of iterations and mean number of function evaluations, for the test problems.

Function	Succ.	Mean Iter.	Mean F. Ev.
CB2	25/25	319.48	6409.6
WF	25/25	306.04	6140.8
SPIRAL	25/25	257.52	5170.4
EVD52	25/25	319.12	6402.4
POLAK6	25/25	337.8	6777.6
WONG1	24/25	366.04	25692.8
WONG2	25/25	353.48	35448.0

7.4 *Synopsis*

Experimental results indicate that PSO is highly efficient in solving minimax problems. Comparison of the results obtained in some test problems with the results obtained using a smoothing technique, indicate that the quality of the solutions obtained through PSO in many cases is superior than that of smoothing.

Furthermore, the success rates were 100% in almost all experiments and the mean number of function evaluations as well as the mean number of PSO's iterations were very small, even in high dimensional test problems.

Thus, PSO can be considered as a good alternative for solving minimax problems, in cases where the gradient-based techniques fail.

8. Particle Swarm Optimization for Integer Programming

8.1 *Motivation and problem formulation*

In the context of numerous optimization problems, fractional values for the variables are either not permitted or practically meaningless. For instance, consider a variable representing the number of employees working on a project, or the number of machines in a production line; evidently such variables cannot take fractional values (Rao, 1996).

In effect, a remarkably wide variety of problems can be represented by discrete optimization models. An important area of application concerns the efficient management of a limited number of resources so as to increase productivity and/or profit. Such applications are encountered in Operational Research problems such as goods distribution, production scheduling, and machine sequencing. Capital budgeting, portfolio analysis, network and VLSI circuit design, as well as automated production systems are some other applications in which Integer Programming problems are met (Nemhauser et al., 1989).

Yet another recent and promising application is the training of neural networks with integer weights, where the activation function and weight values are confined in a narrow band of integers. Such neural networks are better suited for hardware implementations than the real weight ones (Plagianakos and Vrahatis, 1999; Plagianakos and Vrahatis, 2000).

The Integer Programming problem can be stated as

$$\min_x f(x), \qquad x \in S \subseteq \mathbb{Z}^n, \tag{48}$$

where $\mathbb{Z}$ is the set of integers, and S is a not necessarily bounded set, which is considered as the feasible region (maximization Integer Programming problems are very common in the literature, but we will consider only the minimization case, since a maximization problem can be easily turned to a minimization problem).

Optimization techniques applied on real search spaces can be applied on such problems and determine the optimum solution by rounding off the real optimum values to the nearest integer. However, in many cases, certain constraints of the problem are violated due to the rounding of the real optimum solutions. Moreover, the rounding might result in a value of the objective function that is far removed from the optimum (Nemhauser et al., 1989; Rao, 1996).

Early approaches in the direction EA for Integer Programming can be found in (Gall, 1966; Kelahan and Gaddy, 1978). In GA, the potential solutions are encoded in binary bit strings. Since the integer search space, of the problem defined in Equation (48), is potentially not bounded, the representation of a solution using a fixed length binary string is not feasible (Rudolph, 1994). As an alternative, ES can be used, by embedding the search space $\mathbb{Z}^n$ into $\mathbb{R}^n$ and truncating the real values to integers. However, this approach is not always efficient. The reason behind this is the existence of certain features of these algorithms, that are used for detecting real valued minima with arbitrary accuracy. These features are not needed in integer spaces since the smallest distance of two points in ℓ_1 norm is equal to 1. Thus, the search is performed only if the step sizes are greater than 1, and therefore EA for

Integer Programming should operate directly on the integer space (Rudolph, 1994).

In this section, results regarding the performance of the PSO method on the class of problems known as Integer Programming, are reported. The truncation of real values to integers seems not to affect significantly the performance of the method as the experimental results indicate.

8.2 *Experimental results*

Seven Integer Programming test problems were selected to investigate the performance of the PSO method. Each particle of the swarm was truncated to the closest integer, after the determination of its new position using Equation (5).

The test problems are defined immediately below:

TEST PROBLEM 8.1, (Rudolph, 1994):

$$F_1(x) = \|x\|_1 = |x_1| + \ldots + |x_D|,$$

with $x = (x_1, \ldots, x_D) \in [-100, 100]^D$, where D is the corresponding dimension. The solution is $x_i^* = 0$, $i = 1, \ldots, D$, with $F_1(x^*) = 0$.

TEST PROBLEM 8.2, (Rudolph, 1994):

$$F_2(x) = x^\top x = \begin{pmatrix} x_1 & \ldots & x_D \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix},$$

with $x = (x_1, \ldots, x_D)^\top \in [-100, 100]^D$, where D is the corresponding dimension. The solution is $x_i^* = 0$, $i = 1, \ldots, D$, with $F_2(x^*) = 0$.

TEST PROBLEM 8.3, (Glankwahmdee et al., 1979):

$$F_3(x) = -\begin{pmatrix} 15 & 27 & 36 & 18 & 12 \end{pmatrix} x + x^\top \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix} x,$$

with best known solutions $x^* = (0, 11, 22, 16, 6)^\top$ and $x^* = (0, 12, 23, 17, 6)^\top$, with $F_3(x^*) = -737$.

TEST PROBLEM 8.4, (Glankwahmdee et al., 1979):

$$F_4(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,$$

with solution $x^* = (3, 2)^\top$ and $F_4(x^*) = 0$.

TEST PROBLEM 8.5, (Glankwahmdee et al., 1979):

$$F_5(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2,$$

with solution $x^* = (1, 1)^\top$ and $F_5(x^*) = 0$.

TEST PROBLEM 8.6, (Glankwahmdee et al., 1979):

$$F_6(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

with solution $x^* = (1, 1)^\top$ and $F_6(x^*) = 0$.

TEST PROBLEM 8.7, (Glankwahmdee et al., 1979):

$$F_7(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4, \tag{49}$$

with solution $x^* = (0, 0, 0, 0)^\top$ and $F_7(x^*) = 0$.

The PSO's parameters used for all experiments were $c_1 = c_2 = 0.5$ and w was gradually decreased from 1.2 toward 0.4 during the three quarters of the maximum allowed number of iterations.

For the test functions F_1 and F_2, several experiments were performed for different dimensions. For all experiments the initial swarm was taken uniformly distributed inside $[-100, 100]^D$, where D is the corresponding dimension. The swarm's size as well as the maximum number of iterations, for each dimension, for the first two test functions, are exhibited in Table 28. For each dimension and for each function, 100 experiments were performed. The success rate, the mean number of PSO's iterations as well as the mean number of function evaluations for each dimension are reported in Table 29.

Table 28. Swarm's size and maximum number of iterations, for different dimension's values, for the functions F_1 and F_2.

D	Swarm's Size	Max. It.
5	20	1000
10	50	1000
15	100	1000
20	200	1000
25	250	1500
30	300	2000

For the test functions F_3–F_7, the swarm's size was fixed and 100 experiments were done for each of the three different intervals, within which the initial swarm was uniformly distributed. The swarm's size as well as the maximum number of iterations for each of these test functions, are exhibited in Table 30.

The success rate, the mean number of iterations and the mean number of function evaluations for each of the intervals $[-100, 100]^D$, $[-50, 50]^D$, $[-25, 25]^D$, where D is the corresponding dimension of the function, are exhibited in Tables 31–33.

Table 29. Success rates, mean number of iterations and mean number of function evaluations for different dimensions of the functions F_1 and F_2.

Function	D	Suc. Rate	Mean Iter.	Mean F. Eval.
F_1	5	100%	440.72	8834.3
F_2	5	100%	440.92	8838.4
F_1	10	100%	453.48	22724.0
F_2	10	100%	454.88	22794.0
F_1	15	100%	459.44	46044.0
F_2	15	100%	462.88	46388.0
F_1	20	100%	465.24	93248.0
F_2	20	100%	467.08	93616.0
F_1	25	100%	683.44	171110.0
F_2	25	100%	685.44	171610.0
F_1	30	80%	914.64	274692.0
F_2	30	84%	913.28	274284.0

Table 30. Dimension, swarm's size and maximum number of iterations, for the functions F_3–F_7.

Function	D	Swarm's Size	Max. It.
F_3	5	150	1000
F_4	2	20	1000
F_5	2	20	1000
F_6	2	20	1000
F_7	4	40	1000

Table 31. Success rates, mean number of iterations and mean number of function evaluations for the F_3–F_7 with initial interval $[-100, 100]^D$.

Function	Suc. Rate	Mean Iter.	Mean F. Eval.
F_3	80%	499.4	75060.0
F_4	100%	402.32	8066.4
F_5	100%	415.08	8321.6
F_6	100%	418.4	8388.0
F_7	92%	460.92	18476.8

Table 32. Success rates, mean number of iterations and mean number of function evaluations for the F_3–F_7 with initial interval $[-50, 50]^D$.

Function	Suc. Rate	Mean Iter.	Mean F. Eval.
F_3	92%	447.72	67308.0
F_4	100%	328.48	6589.6
F_5	100%	357.12	7162.4
F_6	100%	342.64	6872.8
F_7	100%	440.8	17672.0

Table 33. Success rates, mean number of iterations and mean number of function evaluations for the F_3–F_7 with initial interval $[-25, 25]^D$.

Function	Suc. Rate	Mean Iter.	Mean F. Eval.
F_3	96%	420.28	63192.0
F_4	100%	246.16	4943.2
F_5	100%	228.28	4585.6
F_6	100%	332.08	6661.6
F_7	100%	429.32	17212.8

In a second round of experiments, a PSO with gradually truncated particles was used. Specifically, the particles for the first 50 iterations were rounded to 6 decimal digits (d.d.), for another 100 iterations they were rounded to 4 d.d., for another 100 iterations they were rounded to 2 d.d., and for the rest iterations they were rounded to the nearest integer. The results obtained using this gradually truncated version of PSO, were almost similar to the results exhibited in the tables for the plain PSO. In general, our experience is that PSO is able to cope with Integer Programming problems efficiently.

8.3 *Synopsis*

Experimental results indicate that PSO is an efficient method and should be considered as a good alternative to handle Integer Programming problems. The behavior of the PSO seems to be stable even for high dimensional cases, exhibiting very high success rates even with modest swarm sizes. Moreover, the method does not seem to suffer from search stagnation. The aggregate movement of each particle towards its own best position and the best position ever detected by the swarm, added to its weighted previous position

change, ensures that particles maintain a position change during the process of optimization, which is of proper magnitude.

9. Finding multiple minima using Particle Swarm Optimization

9.1 *Motivation and problem formulation*

In numerous applications, the objective function exhibits multiple global minima, and all or a number of them have to be computed quickly and reliably.

Although, in general, PSO results in global solutions even in high-dimensional spaces, there are some problems that might arise, whenever the objective function has many global and few (or none) local minima.

In this section a strategy that detects all global minima (or some of them if their number is unknown or infinite) is described (Parsopoulos and Vrahatis, 2001a). It is shown, through simulation experiments, that, using a modification of the PSO technique, this strategy is efficient and effective.

Let $f : S \to \mathbb{R}$ be an objective function that has many global minima inside a hypercube S. If we use the plain PSO algorithm to compute just one global minimizer, i.e., a point $x^* \in S$, such that $f(x^*) \leqslant f(x)$, for all $x \in S$, there are two things that might happen: either the PSO will find one global minimum (but we don't foreknow which one) or the swarm will ramble over the search space failing to decide where to land. This rambling might happen due to the equally "good" information that each global minimizer possesses. Each particle moves toward a global minimizer and influences the swarm in order to move toward that direction, but it is also affected by the rest of the particles in order to move toward the other global minimizer that they target. The result of this interaction between particles is a cyclic movement over the search space and results in a disability to detect a minimum. A strategy to overcome these problems and find multiple global minimizers of $f(x)$ is described in the rest of this section (Parsopoulos and Vrahatis, 2001a).

In many applications, such as neural networks training, the goal is to find a global minimizer of a nonnegative function. The global minimum value is a priori known and is equal to zero, but there is a finite (or infinite in the case of neural networks) number of global minimizers. In order to avoid the problem mentioned in the previous paragraph, we can do the following: we determine a not-so-small threshold $\epsilon > 0$ (e.g., if the desired accuracy is 10^{-5}, a threshold around 0.01 or 0.001 will work) and whenever a particle has a functional value that is smaller than ϵ, we pull this particle away from the swarm and isolate it. Simultaneously, we apply *Deflation*:

$$f(x) \leftarrow \frac{f(x)}{\|x - x^*\|},$$

where x^* is the isolated point, or apply Function "Stretching" (see Section 3) to the original objective function $f(x)$ at that point, in order to repel the rest of the swarm from moving toward it. Additionally, a new particle (randomly generated) is added in the swarm, in the place of the isolated particle, in order to keep the swarm's size constant.

Thus, after isolating a particle, we check its functional value. If the functional value is far from the desired accuracy, we can generate a small population of particles around it and constrain this small swarm in the isolated neighborhood of $f(x)$ to perform a finer search, while the big swarm continues searching the rest of the search space for other minimizers. If we set the threshold to a slightly higher value, then the isolated particle is probably a local minimizer and during the local search, a global minimum will not be detected but we have already instructed PSO to avoid it by deflating or stretching it. If we know how many global minimizers of $f(x)$ exist in S then, after some cycles, we will find all of them. In case we do not know the number of global minimizers, we can ask for a specific number of them, or let the PSO run until it reaches the maximum number of iterations in a cycle. This will imply that no other minimizers can be detected by PSO. The whole algorithm can be parallelized and run the two procedures (for the big and the small swarm) simultaneously. A model for the algorithm is given in Table 34.

Table 34. Algorithm model of PSO for locating multiple global minima.

PSO for Locating Multiple Global Minima	
Step 0.	**Set** a threshold $\epsilon > 0$ and the number of desired minima N.
Step 1.	**Set** randomly the population, velocities and set the initial inertia $w \leftarrow w_0$ for PSO. Set the set of found global minima $L = \emptyset$.
Step 2.	**Set** maximum number of iterations MI and the counter $i \leftarrow 0$
Step 3.	**While** $\{\text{card}(L) \neq N \text{AND } i < MI\}$ **Do**
Step 4.	**Perform** a single PSO step and set $i \leftarrow i + 1$, $w \leftarrow w - c$.
Step 5.	**If** the best particle's value $f_{best} \leqslant \epsilon$, **Then** isolate x_{best}.
Step 6.	**Perform** constrained local search around x_{best} (using PSO with a constrained small swarm or another technique).
Step 7.	**If** a minimizer x^* is found by the local search, **Then** set $L \leftarrow L \cup \{x^*\}$.
Step 8.	**Perform** *Deflation* $\left(f(x) \leftarrow \frac{f(x)}{\|x-x^*\|}\right)$ or Function "Stetching" to x_{best} and add a new, randomly chosen, particle into the swarm.
Step 9.	**End While**
Step 10.	**Report** all points in L and other desired results.

In the next section, results regarding the proposed algorithm are reported.

9.2 *Experimental results*

Let f be the 2-dimensional function

$$f(x_1, x_2) = \cos(x_1)^2 + \sin(x_2)^2, \tag{50}$$

where $(x_1, x_2) \in \mathbb{R}^2$. This function has infinite number of minima (all global) in $\mathbb{R}^2$, at the points $(\kappa \frac{\pi}{2}, \lambda \pi)$, where $\kappa = \pm 1, \pm 3, \pm 5, \ldots$ and, $\lambda = 0, \pm 1, \pm 2, \pm 3, \ldots$. We assume that we are interested only in the subset $[-5, 5]^2$ of $\mathbb{R}^2$. In this hypercube, the function f has 12 global (equal to zero) minima. The plot of f is given in Figure 16.

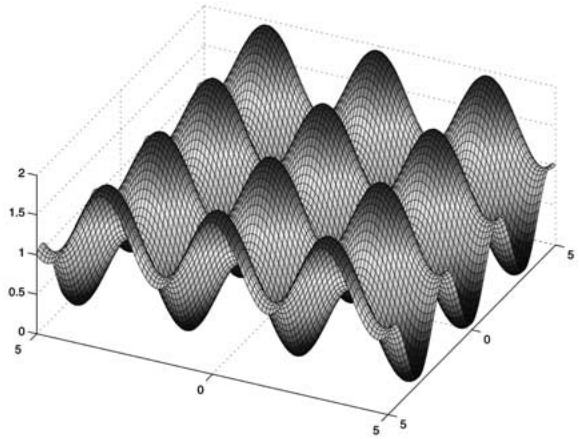

Figure 16. Plot of the function $f(x_1, x_2) = \cos(x_1)^2 + \sin(x_2)^2$, in $[-5, 5]^2$.

If we try to find a single global minimizer of f then we realize that the swarm moves back and forth as described in the previous section, until the maximum number of iterations is attained, failing to detect the minimizer. The trajectory of a single particle, at the first iterations, can be seen in Figure 17. As already mentioned, this happens due to the same information (i.e., functional value) that each global minimizer has. Thus, we could say that the swarm is so "excited" that it cannot decide where to land. Applying the algorithm given in Table 34, we are able to detect all the global minimizers, with accuracy 10^{-5}, even without the need for further local search. The minimizers detected by the algorithm are plotted in Figure 18.

In a second experiment, the algorithm was applied on the *Levy No. 5* test function, defined in Equation (9). Using the presented algorithm, we were able to detect any desired number of minima (local or the global one), in cpu time that does not surpass 760 times the mean cpu time needed to compute each minimizer separately.

In another experiment a neural network has been trained using the PSO to learn the XOR Boolean classification problem. It is well known from the

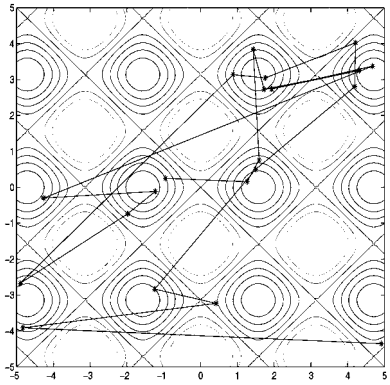

Figure 17. The trajectory of a single particle for some iterations. The particle cannot "land" on a minimizer, although it passes through the neighborhood of several of them. Dark contour lines denote lower function values.

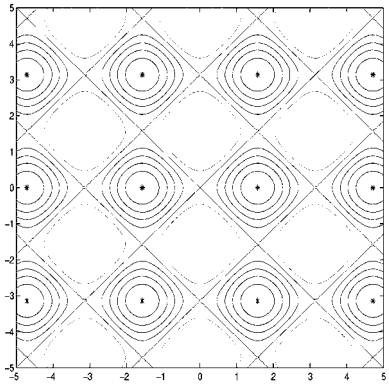

Figure 18. The detected minimizers (denoted as stars).

neural networks literature that successful training in this case, i.e., reaching a global minimizer, strongly depends on the initial weight values and that the error function of the network presents a multitude of local minima. To solve this problem, we use the new algorithm as follows: we set a threshold of 0.1 and start the algorithm as above, but once the standard deviation of the population in an iteration was too close to zero without having functional value close to the threshold, (e.g., if the error value of the network is around 0.5, where there is a well known local minimum of the function), we removed the best particle of the population and isolated it. This particle was potentially a local minimizer (or close to one). Thus, we provided it some new particles (in our simulation the size of the population was 40 thus we were adding 10 particles to the isolated one) and performed a local search in its vicinity (we considered an area of radius 0.01 around it) while the rest of the big swarm continued searching the rest of the space. If the local search yielded a

global minimizer, we added it to our list of found global minima, otherwise it was just a local minimizer that we had avoided. In this way, we were able to detect an arbitrarily large number of global minimizers, while simultaneously avoiding the local ones.

9.3 *Synopsis*

A strategy for locating efficiently and effectively all the global minimizers of a function with many global and local minima has been introduced. Experimental results indicate that the proposed modification of the PSO method is able to detect effectively all the global minimizers instead of rambling over the search space or being attracted by the local minima.

The algorithm provides stable and robust convergence and thus a higher probability of success for the PSO. Also, it can be straightforwardly parallelized.

10. Determination of the heuristic parameters of the Particle Swarm Optimization method

10.1 *Motivation and problem formulation*

All the evolutionary algorithms include a plethora of heuristic parameters that control several of their features. These parameters are usually problem-dependent and defined by the user. A proper choice is a matter of the user's experience and the available information on the problem under consideration. Most importantly, these heuristic parameters may impact the convergence properties of the algorithm. Due to the fact that, even experienced users may provide the algorithm with an inappropriate set of parameters, causing failure, there is an ongoing demand for numerical algorithms that involve effective mechanisms for the parameters' manipulation.

In this section, the Differential Evolution (DE) algorithm (Storn and Price, 1997) is used to manipulate the heuristics of the PSO method (Laskari et al., 2001). Although DE itself contains heuristics, they are not of critical importance for the convergence of the overall method. As shown in experimental results, the mean values of the parameters selected by the DE algorithm, are similar to the values proposed in other studies through numerous experiments (Carlisle and Dozier, 2001; Eberhart and Shi, 2000).

10.2 *The Differential Evolution algorithm*

The DE algorithm has been developed by Storn and Price (Storn and Price, 1997). It is a numerical algorithm for GO, that utilizes a population of NP,

D-dimensional parameter vectors $x_{i,G}$, $i = 1, 2, \ldots, NP$, to probe the search space simultaneously. The initial population is uniformly distributed within the search space. At each iteration, the so-called in this context, *mutation* and *crossover* operators are applied on the population and a new population is generated. After that, the selection phase starts, and the best NP points are selected to comprise the next generation's population.

Applying the *mutation* operator, for each vector $x_{i,G}$, $i = 1, 2, \ldots, NP$, a mutant vector is determined according to the equation

$$v_{i,G+1} = x_{r_1,G} + F\ (x_{r_2,G} - x_{r_3,G}), \tag{51}$$

where $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$, are mutually different random indexes, and $F \in [0, 2]$. The indexes r_1, r_2, r_3, also need to differ from the current index i. Consequently, NP must be greater than or equal to 4, to apply the mutation operator.

Once mutation is completed, the *crossover* operator is applied on the population. A trial vector

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}), \tag{52}$$

is generated, where

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & if\ (\text{randb}(j) \leqslant CR)\ \ or\ \ j = \text{rnbr}(i), \\ x_{ji,G} & if\ (\text{randb}(j) > CR)\ \ and\ \ j \neq \text{rnbr}(i), \end{cases} \tag{53}$$

where $j = 1, 2, \ldots, D$; randb(j) is the j-th evaluation of a uniform random number generator within the range of [0, 1]; CR is the crossover constant within the range [0, 1] (user defined); and rnbr(i) is a randomly chosen index from the set $\{1, 2, \ldots, D\}$.

The vector $u_{i,G+1}$ is defined as $x_{i,G+1}$, i.e., it is included in the population comprising the next generation, if and only if it has a lower function value than $x_{i,G}$. Otherwise $x_{i,G+1}$ equals $u_{i,G+1}$.

The procedure described above is the standard variant of the DE algorithm. Different types of mutation and crossover operators have been studied with promising results (Storn and Price, 1997). In the next section, the proposed algorithm is exposed.

10.3 *The Composite PSO algorithm*

In initial experiments, the tuning of the heuristic PSO parameters was assigned to a second PSO. In effect, there was one PSO solving the GO problem, and a second PSO running as a background procedure, tuning the heuristics of the first. The results for this approach were not very promising. It seems that the utilization of a method of the same dynamic could not provide PSO with information concerning the appropriate values of its

parameters. Thus experiments on this idea were abandoned. A potential theoretical explanation for this phenomenon is being studied, and may appear in future work.

Alternatively, in a second experiment, we developed a composite numerical algorithm that utilizes the DE method to determine the appropriate set of parameters for the PSO at each iteration, during the optimization. Thus, a population of 3-dimensional vectors is used, where each vector assigns values to w, c_1 and c_2. The technique is described in the following paragraphs.

Suppose that we are at the n-th iteration of the PSO technique and the appropriate values of w, c_1 and c_2 have to be determined, in order to update velocities and generate the new swarm. The swarm S_{n-1} of the $(n-1)$-th iteration is already known. The DE algorithm generates randomly a population, P, of 3-dimensional vectors, where each consists of the values of the three PSO parameters. For each vector, p_l, of this population, the new velocities are determined and the corresponding new swarm is obtained. The new swarm is then evaluated, and the function value of its best particle characterizes the vector p_l, i.e. it is taken as its fitness function value. The same procedure is applied to all vectors p_l of the population, P. Afterwards, the DE operators are applied to P, and transform it. Finally, selection takes place and a new population is generated. The DE algorithm performs a small number of iterations, and the best vector of the population is the final choice for the generation of the n-th iteration's swarm. The algorithm is exhibited in Table 35.

A common termination criterion for the DE algorithm is the performance of a predefined maximum number of iterations. If a satisfactory solution is obtained during the DE iterations, i.e., a solution of the GO problem is obtained during the evaluation process of the DE algorithm, the overall algorithm is immediately terminated. The termination condition for the PSO is usually the achievement of a solution with the desired accuracy or the loss of the swarm's diversity. A maximum number of allowed iterations is also set for the PSO, and the algorithm stops as soon as this number is reached.

Clearly, the proposed algorithm is slower than the plain PSO, due to the DE iterations that are performed for each PSO iteration. However, the time needed for these iterations is usually short, since their number is small and DE is a very fast algorithm. The advantage is that the convergence of the PSO algorithm becomes independent of the heuristic parameters and consequently the user need not be concerned with the parameters' configuration. Most importantly, the proposed technique exhibits higher success rates than the plain PSO.

It is very interesting to compare the mean values of the parameters w, c_1 and c_2, which are determined by the DE algorithm, with the values that are presented in other extensive studies, and considered as good default values if

Table 35. The Composite PSO algorithm.

The Composite PSO algorithm	
Input:	Initial swarm S_0 and velocities V_0. Set $i \leftarrow -1$.
Step 1.	**Set** $i \leftarrow i + 1$ and $j \leftarrow 0$.
Step 2.	**Generate** a random population P^j of 3-dimensional vectors p_l^j, $l = 1, 2, \ldots, popsize$.
Step 3.	**For** each p_l^j, **Set** $w \leftarrow p_l^j(1)$, $c_1 \leftarrow p_l^j(2)$ and $c_2 \leftarrow p_l^j(3)$ and **Determine** V_{i+1} and S_{i+1} using Equations (4) and (5).
Step 4.	**Evaluate** S_{i+1}. Let s_g be its best particle. Use $F(s_g)$ as the fitness value of p_l^j.
Step 5.	**Apply** mutation, crossover and selection on P^j, according to Equations (51), (52) and (53), and generate a new population P^{j+1}. Let p^* be the best individual of P^{j+1}.
Step 6.	**Check** the DE stopping criterion. If it is not satisfied, **Set** $j \leftarrow j + 1$ and go to Step 3. Otherwise go to Step 7.
Step 7.	**Take** $w \leftarrow p^*(1)$, $c_1 \leftarrow p^*(2)$ and $c_2 \leftarrow p^*(3)$ and **Determine** V_{i+1} and S_{i+1} using Equations (4) and (5).
Step 8.	**Check** the PSO stopping criterion. If it is not satisfied, go to Step 1. Otherwise terminate the algorithm.

the user has no prior information of the problem under consideration. These results form the core of the next section.

10.4 *Experimental results*

We used the proposed algorithm to find the global minimum of twelve test functions (Laskari et al., 2001). The names of the test functions, their dimensions as well as their corresponding references, are reported in Table 36.

For each test function 25 runs were performed. The size of the DE population and its maximum number of iterations were fixed for all experiments, equal to 5 and 15 respectively. The DE parameters were fixed, $F = 0.5$, $CR = 0.1$. These values are considered as good default values. Different sets of parameters were also tested, but no significant changes were observed. The swarm's size, the hypercube inside which the initial swarm was taken, and the maximum allowed number of PSO iterations were problem-dependent. Their values for the various test functions, are exhibited in Table 37.

The values for w, c_1 and c_2, were bounded in $1.2 \leqslant w \leqslant 0.4$ and $4 \leqslant c_1, c_2 \leqslant 0.1$. The iterations performed by the proposed algorithm were far less than the iterations required by the plain PSO method. As expected,

Table 36. Test functions used in experiments.

Function's Name	Dimension	Reference
Banana Valley	2	(Storn and Price, 1997)
Rosenbrock	2	(More et al., 1981)
Six-Hump Camel Valley	2	(Snyman and Fatti, 1987)
Goldstein-Price	2	(Snyman and Fatti, 1987)
Griewangk 2D	2	(Snyman and Fatti, 1987)
Branin	2	(Storn and Price, 1997)
Levy No. 5	2	(Levy et al., 1981)
Hellical Valley	3	(More et al., 1981)
Levy No.8	3	(Levy et al., 1981)
Hyperbolic Ellipsoid	6	(Storn and Price, 1997)
Rastrigin 6D	6	(Storn and Price, 1997)
Griewangk 10D	10	(Storn and Price, 1997)

Table 36. PSO parameters.

Function's Name	Swarm Size	Init. Hyperc.	Max. Iter.
Banana Valley	20	$[-5, 5]^2$	200
Rosenbrock	20	$[-5, 5]^2$	200
Six-Hump Camel Valley	20	$[-5, 5]^2$	200
Goldstein-Price	20	$[-5, 5]^2$	200
Griewangk 2D	20	$[-5, 5]^2$	200
Branin	20	$[-5, 5]^2$	200
Levy No. 5	20	$[-5, 5]^2$	200
Hellical Valley	20	$[-2, 2]^3$	300
Levy No.8	20	$[-5, 5]^3$	200
Hyperbolic Ellipsoid	40	$[-0.5, 0.5]^6$	300
Rastrigin 6D	40	$[-0.2, 0.2]^6$	300
Griewangk 10D	60	$[-0.2, 0.2]^{10}$	300

the total number of function evaluations was larger. The success rates of the proposed algorithm were higher than those of plain PSO, more specifically the success rates were always 100%, even in cases where the plain PSO performed badly (e.g., the Hellical Valley function).

The aspect of critical importance is the comparison of the mean values of the PSO parameters chosen by the DE algorithm, and their corresponding values presented in related studies (Carlisle and Dozier, 2001; Eberhart and

Shi, 2000; Shi and Eberhart, 1998b). Both the mean values for all test functions, as well as the success rates for both the proposed and the plain PSO algorithm, are exhibited in Table 38.

Table 38. Mean Values for w, c_1, c_2, and success rates.

Function's Name	Mean w	Mean c_1	Mean c_2	Plain PSO	Comp. PSO
Banana Valley	0.7145	2.0967	1.2919	84%	100%
Rosenbrock	0.7337	2.0421	1.1943	88%	100%
Six-Hump Camel Valley	0.7153	2.1574	1.2706	100%	100%
Goldstein-Price	0.7085	2.0378	1.2712	100%	100%
Griewangk 2D	0.7082	2.0171	1.3926	96%	100%
Branin	0.7329	2.0197	1.3148	100%	100%
Levy No. 5	0.7023	2.0145	1.2098	84%	100%
Hellical Valley	0.7053	2.1062	1.1694	36%	100%
Levy No.8	0.7739	1.9722	1.3063	100%	100%
Hyperbolic Ellipsoid	0.7179	1.9796	1.3812	100%	100%
Rastrigin 6D	0.7027	2.0462	0.9881	90%	100%
Griewangk 10D	0.6415	2.1291	1.4534	100%	100%

As reported in Table 38, the DE algorithm always chose a larger c_1 than c_2. This tends to enforce the global perspective of the PSO, i.e., to maintain the population's diversity for a large number of iterations, and consequently, to avoid premature convergence to local, instead of global, minima. Parameters c_1 and c_2 appear to be interrelated, and their sum is around 3.5. The mean value of w was always within the range $[0.6, 0.8]$ and the deviation from these values was not large. These results are in line with the values reported in the related literature (Carlisle and Dozier, 2001; Eberhart and Shi, 2000; Shi and Eberhart, 1998b). The contribution of the proposed algorithm lies in the fact that the determination of the heuristic parameters was assigned to the DE algorithm, in contrast to the previous common practice of running numerous experiments.

10.5 *Synopsis*

A variant of the PSO algorithm that utilizes the DE algorithm to manipulate its heuristic parameters has been exposed. The utilization of the DE algorithm was chosen following the failure to manipulate these parameters by a second PSO, running as a background procedure. Different values of the DE heuristic parameters seem not to impact the convergence of the overall method.

The social parameter, c_2, of the PSO was always assigned smaller values than the cognitive parameter, c_1. The implication of this assignment of values

is the maintenance of the diversity of the swarm, which in turn, tends to alleviate local minima. Therefore, our results indicate that the original assignment of equal values to c_1 and c_2, may be misleading and result in inefficient solutions. A value of w within the range [0.6, 0.8] appears to be a good alternative if gradually decreasing values are not used. The results are in line with those reported in the related literature (Carlisle and Dozier, 2001; Eberhart and Shi, 2000; Shi and Eberhart, 1998b). The central contribution of the proposed algorithm lies in the fact that the manipulation of the heuristic parameters was assigned to the DE algorithm, in contrast to the previous common practice of running numerous experiments to determine these parameters. Furthermore, the proposed algorithm surpassed the success rates of the plain PSO, detecting the global minima in all experiments.

11. Conclusions

An overview of recent results of ours, regarding the PSO method was presented. Techniques for the alleviation of local minima, and for detecting multiple minimizers were described and algorithm models were given. Moreover, the ability of PSO to solve problems in noisy and continuously changing environments was investigated, and results were reported for test functions as well as for the Light Scattering problem. Multiobjective, Minimax, Integer Programming and ℓ_1 errors-in-variables problems were also tackled using PSO with very promising results. Finally, a Composite PSO, in which a DE strategy was incorporated to control the heuristic parameters of the plain PSO during the optimization, was described, and results for many well-known test functions were given with respect to it.

Conclusively, PSO appears to be a very useful technique for solving GO problems, and a good alternative in cases where other techniques fail, although further research is required to fully comprehend the dynamics and the potential limits of this technique.

Acknowledgements

The authors wish to thank Prof. Hans-Paul Schwefel for his constructive comments on an early draft of this paper. This material was partially supported by the Deutsche Forschungsgemeinschaft-DFG (German National Research Foundation) as a part of the collaborative research center "Computational Intelligence" (SFB 531).

Part of this work was done while the authors were at the Department of Computer Science, University of Dortmund, D–44221 Dortmund, Germany.

Appendix

A SIMPLE IMPLEMENTATION OF THE
Particle Swarm Optimization IN MATLAB

```
function [xmin, fxmin, iter] = PSO

% Initializing variables
success = 0; % Success flag
PopSize = 20; % Size of the swarm
MaxIt = 5000; % Maximum number of iterations
iter = 0; % Iterations'counter
fevals = 0; % Function evaluations' counter
maxw = 1.2; % Maximum inertia weight's value
minw = 0.1; % Minimum inertia weight's value
weveryit = floor(0.75*MaxIt); % Inertia decr. step
c1 = 0.5; % PSO parameter C1
c2 = 0.5; % PSO parameter C2
inertdec = (maxw-minw)/weveryit; % Inertia weight's decrement
w = maxw; % initial inertia weight
f = "DeJong"; % Objective Function
dim = 2; % Dimension of the problem
upbnd = 5; % Upper bound for init. of the swarm
lwbnd = –5; % Lower bound for init. of the swarm
GM = 0; % Global minimum (used in the stopping criterion)
ErrGoal = 1e–3; % Desired accuracy

% Initializing swarm and velocities
popul = rand(dim, PopSize)*(upbnd-lwbnd) + lwbnd;
vel = rand(dim, PopSize);

% Evaluate initial population
for i = 1:PopSize,
    fpopul(i) = feval(f, popul(:,i));
    fevals = fevals + 1;
end

% Initializing Best positions' matrix and
% the corresponding function values
bestpos = popul;
fbestpos = fpopul;

% Finding best particle in initial population
[fbestpart,g] = min(fpopul);
lastbpf = fbestpart;

% SWARM EVOLUTION LOOP * START *
while (success == 0) & (iter < MaxIt),
    iter = iter + 1;
```

```
    % Update the value of the inertia weight w
    if (iter<=weveryit)
       w = maxw – (iter–1)*inertdec;
    end

    % VELOCITY UPDATE
    for i=1:PopSize,
       A(:,i) = bestpos(:,g);
    end
    R1 = rand(dim, PopSize);
    R2 = rand(dim, PopSize);
    vel = w*vel + c1*R1.*(bestpos-popul) + c2*R2.*(A-popul);

    % SWARM UPDATE
    popul = popul + vel;

    % Evaluate the new swarm
    for i = 1:PopSize,
       fpopul(i) = feval(f,popul(:, i));
       fevals = fevals + 1;
    end

    % Updating the best position for each particle
    changeColumns = fpopul < fbestpos;
    fbestpos = fbestpos.*( changeColumns) + fpopul.*changeColumns;
    bestpos(:, find(changeColumns)) = popul(:, find(changeColumns));

    % Updating index g
    [fbestpart, g] = min(fbestpos);
    % Checking stopping criterion
    %if abs(fbestpart-lastbpf) <= ErrGoal
    if abs(fbestpart-GM) <= ErrGoal
       success = 1;
    else
       lastbpf = fbestpart;
    end
end
% SWARM EVOLUTION LOOP * END *

% Output arguments
xmin = popul(:,g);
fxmin = fbestpos(g);

function DeJong=DeJong(x)
DeJong = sum(x.^2);
```

References

Ahonen H, Desouza PA and Garg VK (1997) A Genetic Algorithm for Fitting Lorentzian Line-Shapes in Mossbauer-Spectra 124(4): 633–638

Angeline PJ (1997) Tracking Extrema in Dynamic Environments. Proceedings of International Conference on Evolutionary Programming

Angeline PJ (1998) Evolutionary optimization versus Particle Swarm Optimization: Philosophy and performance differences. In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds) Evolutionary Programming VII, pp. 601–610. Springer

Arnold DV (2001) Local Performance of Evolution Strategies in the Presence of Noise. Ph.D. thesis, Department of Computer Science, University of Dortmund, Germany

Bäck T (1996) Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York

Bäck T, Fogel D and Michalewicz Z (1997) Handbook of Evolutionary Computation. IOP Publishing and Oxford University Press, New York

Bandler JW and Charalambous C (1974) Nonlinear programming using minimax techniques. J. Optim. Th. Appl. 13: 607–619

Banzhaf W, Nordin P, Keller RE and Francone FD (1998) Genetic Programming – An Introduction. Morgan Kaufman, San Francisco

Bertsekas DP (1976) Minimax Methods Based on Approximations. Proceedings 1976 John Hopkins Conf. Inform. Sciences and Systems

Bertsekas DP (1976) A new algorithm for solution of nonlinear resistive networks involving diodes. IEEE Trans. Circ. Th. 23: 599–608

Bertsekas DP (1977) Approximation procedures based on the method of multipliers. J. Optim. Th. Appl. 23: 487–510

Beyer H-G (2000) Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. Comput. Methods Appl. Mech. Engrg. 186: 239–269

Beyer H-G (2001) The Theory of Evolution Strategies. Springer, Berlin

Beyer H-G and Schwefel H-P (2002) Evolution Strategies: A Comprehensive Introduction. Natural Computing, to appear

Blum EK (1989) Approximation of boolean functions by sigmoidal networks, Part I: XOR and other two-variable functions. Neural Computation 1: 532–540

Bohren CF and Huffman DR (1983) Absorption and Scattering of Light by Small Particles. Wiley, New York

Boutsinas B and Vrahatis MN (2001) Artificial nonmonotonic neural networks. Artificial Intelligence 132: 1–38

Box GEP and Muller ME (1958) A note on the generation of random normal deviates. Ann. Math. Statistics 29: 610–611

Britt HI and Luecke RH (1973) The estimation of parameters in nonlinear, implicit models. Technometrics 15: 233–247

Burke J and Xu S (2000) An non-interior predictor-corrector path-following algorithm for the monotone linear complementarity problem. Math. Progr. 87: 113–130

Bush TS, Catlow CRA and Battle PD (1995) Evolutionary programming techniques for predicting inorganic crystal-structures. J. Materials Chemistry 5(8): 1269–1272

Carlisle A and Dozier G (2001) An Off-The-Shelf PSO. Proceedings of the Particle Swarm Optimization Workshop, pp. 1–6

Charalambous C and Conn AR (1978) An efficient method to solve the minimax problem directly. SIAM J. Numerical Analysis 15: 162–187

Corana A, Marchesi M, Martini C and Ridella S (1987) Minimizing multimodal functions of continuous variables with the 'simulated annealing algorithm'. ACM Transactions on Mathematical Software 13(3): 262–280

Demyanov VF and Molozemov VN (1974) Introduction to Minimax. Wiley, New York

Drossos L, Ragos O, Vrahatis MN and Bountis TC (1974) Method for computing long periodic orbits of dynamical systems. Physical Review E 53: 1206–1211

Du DZ and Pardalos PM (1995) Minimax and Applications. Kluwer, Dordrecht

Eberhart RC and Kennedy J (1995) A New Optimizer Using Particle Swarm Theory, Proceedings Sixth Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Service Center, Piscataway, NJ

Eberhart RC and Shi Y (1998) Comparison between genetic algorithms and Particle Swarm Optimization. In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds) Evolutionary Programming VII, pp. 611–616. Springer

Eberhart RC and Shi Y (2000) Comparing inertia weights and constriction factors in Particle Swarm Optimization. Proceedings of the Congress on Evolutionary Computating, pp. 84–88

Eberhart RC, Simpson P and Dobbins R (1996) Computational Intelligence PC Tools. Academic Press

Elster C and Neumaier A (1995) A grid algorithm for bound constrained optimization of noisy functions. IMA Journal of Numerical Analysis 15: 585–608

Elster C and Neumaier A (1997) A method of trust region type for minimizing noisy functions. Computing 58: 31–46

Facchinei F, Jiang H and Qi L (1999) A smoothing method for mathematical programs with equilibrium constraints. Math. Progr. 85: 107–134

Fletcher R (1987) Practical Methods of Optimization. Wiley, New York

Fogel D (1996) Evolutionary Computation: Towards a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, NJ

Forgó F (1988) Nonconvex Programming. Akadémiai Kiadó, Budapest

Gall DA (1966) A practical multifactor optimization criterion. In: Levi A and Vogl TP (eds) Recent Advances in Optimization Techniques, pp. 369–386

Gilmore T and Keeley CT (1995) An implicit filtering algorithm for optimization of functions with many local minima. SIAM J. Optim. 5: 269–285

Glankwahmdee A, Liebman JS and Hogg GL (1979) Unconstrained discrete nonlinear programming. Engineering Optimization 4: 95–107

Goldberg D (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, Reading, MA

Hansen ER (1992) Global Optimization Using Interval Analysis. Marcel Dekker, New York

Hansen N (1998) Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie. Mensch & Buch, Berlin

Hansen N and Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9: 159–195

Heppner F and Grenander U (1990) A stochastic nonlinear model for coordinate bird flocks. In: Krasner S (ed) The Ubiquity of Chaos. AAAS Publications, Washington, DC

Hirriart JB (1978) On optimality conditions in non-differentiable programming. Mathematical Programming 14: 73–86

Hodgson RJW (2000) Genetic algorithm approach to particle identification by light scattering. J. Colloid and Interface Science 229: 399–406

Hooke R and Jeeves TA (1961) Direct search solution of numerical and statistical problems. J. ACM 8: 212–229

Horst R and Pardalos PM (1995) Handbook of Global Optimization. Kluwer Academic Publishers, London

Horst R and Tuy H (1996) Global Optimization – Deterministic Approaches. Springer, New York

Jaynes E (1979) Where do we stand on maximum entropy? In: Levine R and Tribus M (eds) The Maximum Entropy Formalism, pp. 15–118. MIT Press, Cambridge

Jin Y, Olhofer M and Sendhoff B (2001) Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? Proceedings GECCO 20001 Conference, to appear

Kalantonis VS, Perdios EA, Perdiou AE and Vrahatis MN (2001) Computing with certainty individual members of families of periodic orbits of a given period. Celestial Mechanics and Dynamical Astronomy 80: 81–96

Kelahan RC and Gaddy JL (1978) Application of the adaptive random search to discrete and mixed integer optimization. Int. J. Num. Meth. Engin. 12: 289–298

Kelley R (1999) Iterative Methods for Optimization. SIAM, Philadelphia

Kennedy J 1998, The behavior of particles. In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds) Evolutionary Programming VII, pp. 581–590. Springer

Kennedy J and Eberhart RC (1995) Particle Swarm Optimization. Proceedings IEEE International Conference on Neural Networks, IV: pp. 1942–1948.. IEEE Service Center, Piscataway, NJ

Kennedy J and Eberhart RC (2001) Swarm Intelligence. Morgan Kaufmann Publishers

Knowles JD and Corne DW (2000) Approximating the nondominated front using the pareto archived evolution strategies. Evolutionary Computation 8(2): 149–172

Kort BW and Bertsekas DP (1972) A new penalty function algorithm for constrained minimization. Proceedings 1972 IEEE Conf. Decision and Control

Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA

Laskari EC, Parsopoulos KE and Vrahatis MN (2001) Determination of the heuristic parameters of the Particle Swarm Optimization method. Proceedings Numerical Algorithms Conference submitted

Laskari EC, Parsopoulos KE and Vrahatis MN (2002a) Particle Swarm Optimization for Minimax Problems. IEEE Congress on Evolutionary Computation, pp. 1576–1581

Laskari EC, Parsopoulos KE and Vrahatis MN (2002b) Particle Swarm Optimization for Integer Programming. IEEE Congress on Evolutionary Computation, pp. 1582–1587

Levy A, Montalvo A, Gomez S and Galderon A (1981) Topics in Global Optimization. Springer-Verlag, New York

Li XS (1991) An aggregate function method for nonlinear programming. Science in China (A) 34: 1467–1473

Lukšan L and Vlček J (2000) Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report No. 798, Institut of Computer Science, Academy of Sciences of the Czech Republic

Michalewicz Z (1994) Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin

Millonas MM (1994) Swarms, phase transitions, and collective intelligence. In: Palaniswami M, Attikiouzel Y, Marks R, Fogel D and Fukuda T (eds) Computational Intelligence: A Dynamic System Perspective, pp. 137–151. IEEE Press, Piscataway, NJ

More JJ, Garbow BS and Hillstrom KE (1981) Testing unconstrained optimization software. ACM Transactions on Mathematical Software 7(1): 17–41

Murray W and Overton ML (1980) A projected lagrangian algorithm for nonlinear minimax optimization. SIAM J. Scient. Stat. Comp. 1: 345–370

Nelder JA and Mead R (1965) A simplex method for function minimization. Computer Journal 7: 308–313

Nemhauser GL, Rinooy Kan AHG and Todd MJ (eds) (1989) Handbooks in OR & MS, Vol. 1: Optimization. Elsevier

Nocedal J (1991) Theory of algorithms for unconstrained optimization. In: Iserles A (ed) Acta Numerica, pp. 199–242. Cambridge University Press, Cambridge

Osborne MR and Watson GA (1969) An algorithm for minimax approximation in the non-linear case. Comput. J. 12: 63–68

Parsopoulos KE, Plagianakos VP, Magoulas GD and Vrahatis MN (2001a) Improving the Particle Swarm Optimizer by Function "Stretching". In: Hadjisavvas N and Pardalos PM (eds) Advances in Convex Analysis and Global Optimization, pp. 445–457. Kluwer Academic Publishers

Parsopoulos KE, Plagianakos VP, Magoulas GD and Vrahatis MN (2001b) Objective Function "Stretching" to alleviate convergence to local minima. Nonlinear Analysis, TMA 47(5): 3419–3424

Parsopoulos KE, Plagianakos VP, Magoulas GD and Vrahatis MN (2001c) Stretching technique for obtaining global minimizers through Particle Swarm Optimization. Proceedings of the Particle Swarm Optimization workshop, pp. 22–29

Parsopoulos KE, Laskari EC and Vrahatis MN (2001d) Solving ℓ_1 norm errors-in-variables problems using Particle Swarm Optimization. In: Hamza MH (ed) Artificial Intelligence and Applications, pp. 185–190. IASTED/ACTA Press

Parsopoulos KE and Vrahatis MN (2001a) Modification of the Particle Swarm Optimizer for locating all the global minima. In: Kurkova V, Steele NC, Neruda R and Karny M (eds) Artificial Neural Networks and Genetic Algorithms, pp. 324–327. Springer, Wien

Parsopoulos KE and Vrahatis MN (2001b) Particle Swarm Optimizer in noisy and continuously changing environments. In: Hamza MH (ed) Artificial Intelligence and Soft Computing, pp. 289–294. IASTED/ACTA Press

Parsopoulos KE and Vrahatis MN (2001c) Particle Swarm Optimization for imprecise problems. Proceedings of the 5th International Workshop on Mathematical Methods In Scattering Theory and Biomedical Technology, in press

Parsopoulos KE and Vrahatis MN (2002) Particle Swarm Optimization method in multiobjective problems. Proceedings ACM Symposium on Applied Computing (SAC 2002), pp. 603–607

Paszkowicz W (1996) Application of the smooth genetic algorithm for indexing powder patterns: Test for the orthorhombic system. Materials Science Forum 228(1 & 2): 19–24

Plagianakos VP and Vrahatis MN (1999) Training neural networks with 3-bit integer weights. In: Bahnzaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M and Smith RE (eds) Proceedings of Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 910–915. Morgan Kaufmann

Plagianakos VP and Vrahatis MN (2000) Training neural networks with threshold activation functions and constrained integer weights. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2000)

Peng J-M and Lin Z (1999) A non-interior continuation method for generalized linear complementarity problems. Math. Progr. 86: 533–563

Pierre DA (1986) Optimization Theory and Applications. Dover Publications

Pintér JD (1996) Global Optimization in Action. Academic Publishers

Polak E (1987) On the mathematical foundations of nondifferentiable optimization. SIAM Review 29: 21–89

Polak E (1997) Optimization: Algorithms and Consistent Approximations. Springer-Verlag, New York

Powell MJD (1988) A review of algorithms for nonlinear equations and unconstrained optimization. Proceedings ICIAM, pp. 220–232

Powell MJD (1992) A direct search optimization method that models the objective and constraint functions by linear interpolation. DAMTP 1992/NA5, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England

Rao SS (1996) Engineering optimization-theory and practice. Wiley

Rechenberg I (1994) Evolution Strategy. In: Zurada JM, Marks RJ II and Robinson C (eds) Computational Intelligence: Imitating Life. IEEE Press, Piscataway, NJ

Reeves WT (1983) Particle systems – a technique for modelling a class of fuzzy objects. ACM Transactions on Graphics 2(2): 91–108

Reynolds CW (1987) Flocks, herds, and schools: a distributed behavioral model. Computer Graphics 21(4): 25–34

Rudolph G (1994) An evolutionary algorithm for integer programming. In: Davidor Y, Schwefel H-P, Männer R (eds), pp. 139–148. Parallel Problem Solving from Nature 3

Rudolph G (1997) Convergence properties of evolutionary algorithms. Verlag Dr. Kovač, Hamburg

Salomon R (1998) Evolutionary search and gradient search: Similarities and differences. IEEE Trans. Evolutionary Computation 2: 45–55

Schaffer JD (1985) Genetic Algorithms and their Applications: Proceedings of the first Int. Conf. on Genetic Algorithms, pp. 93–100

Schwefel H-P (1975) Evolutionsstrategie und numerische Optimierung. Technical University of Berlin, Department of Process Engineering, Dr.-Ing. Thesis

Schwefel H-P (1981) Numerical Optimization of Computer Models. Wiley, Chichester

Schwefel H-P (1995) Evolution and Optimum Seeking. Wiley, New York

Schwefel H-P and Rudolph G (1995) Contemporary evolution strategies. In: Morana F, Moreno A, Merelo J and Chacon P (eds), pp. 893–907. Advances in Artificial Life, Proceedings 3rd ECAL

Shi Y and Eberhart RC (1998) Parameter selection in Particle Swarm Optimization. In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds) Evolutionary Programming VII, pp. 611–616. Springer

Shi Y and Eberhart RC (1998) A modified Particle Swarm Optimizer. Proceedings of the 1998 IEEE Conference on Evolutionary Computation. AK, Anchorage

Skinner AJ and Broughton JQ (1995) Neural networks in computational materials science: Training algorithms. Modelling and Simulation in Material Science and Engineering 3(3): 371–390

Snyman JA and Fatti LP (1987) A multi-start global minimization algorithm with dynamic search trajectories. J. of Optimization Theory and Applications 54(1): 121–141

Spall JC (1992) Multivariate stochastic approximation using a simultaneous perturbation gradient approximation'. IEEE Trans. Automatic Control 37: 332–341

Spall JC (1998a) Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans. Aerospace and Electronic Systems 34: 817–823

Spall JC (1998b) An overview of the simultaneous perturbation method for efficient optimization. Johns Hopkins APL Technical Digest 19: 482–492

Storn R and Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11: 341–359

Swinehart K, Yasin M and Guimaraes E (1996) Applying an analytical approach to shop-floor scheduling: A case-study. Int. J. of Materials & Product Technology 11(1–2): 98–107

Torczon V (1989) A direct search algorithm for parallel machines. Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, USA

Torczon V (1991) On the convergence of the multidimensional search algorithm. SIAM J. Optimization 1: 123–145

Törn A and Žilinskas A (1989) Global Optimization. Springer-Verlag, Berlin

Vrahatis MN, Androulakis GS, Lambrinos JN and Magoulas GD (2000) A class of gradient unconstrained minimization algorithms with adaptive stepsize. Journal of Computational and Applied Mathematics 114: 367–386

Vrahatis MN, Androulakis GS and Manoussakis ME (1996) A new unconstrained optimization method for imprecise function and gradient values. Journal of Mathematical Analysis and Applications 197: 586–607

Vrahatis MN, Perdiou AE, Kalantonis VS, Perdios AE, Papadakis K, Prosmiti R and Farantos SC (2001) Application of the characteristic bisection method for locating and computing periodic orbits in molecular systems. Computer Physics Communications 138: 53–68

Waren AD, Lasdon LS and Suchman DF (1967) Optimization in engineering design. Proceedings IEEE 55: 1885–1897

Xu S (2001) Smoothing method for minimax problems. Comp. Optim. Appl. 20: 267–279

Watson GA (1997) The use of the ℓ_1 norm in nonlinear errors-in-variables problems. In: Van Huffel (ed) Proceedings of Recent Advances on Total Least Squares Techniques & Errors-in-Variables Modeling

Watson GA (1998) Choice of norms for data fitting and function approximation. Acta Numerica, pp. 337–377

Watson GA and Yiu KFC (1991) On the solution of the errors in variables problem using the ℓ_1 norm. BIT 31: 697–710

Wilson EO (1975) Sociobiology: The New Synthesis. Belknap Press, Cambridge, MA

Zitzler E (1999) Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. Thesis, Swiss Federal Institute of Technology Zurich

Zitzler E, Deb K and Thiele L (2000) Comparison of multiobjective evolution algorithms: empirical results. Evolutionary Computation 8(2): 173–195

Zuhe S, Neumaier A and Eiermann MC (1990) Solving minimax problems by interval methods. BIT 30: 742–751