Chapter 28

IMPROVING THE PARTICLE SWARM OPTIMIZER BY FUNCTION "STRETCHING"

K. E. Parsopoulos
Department of Mathematics, University of Patras, GR-26110, Patras, Greece.
kostasp@math.upatras.gr

V. P. Plagianakos
Department of Mathematics, University of Patras, GR-26110, Patras, Greece.
vpp@math.upatras.gr

G. D. Magoulas
Department of Information Systems and Computing, Brunel University, Uxbridge, UB8 3PH, United Kingdom.
George.Magoulas@brunel.ac.uk

M. N. Vrahatis
Department of Mathematics, University of Patras, GR-26110, Patras, Greece.
vrahatis@math.upatras.gr

Abstract In this paper a new technique, named Function "Stretching", for the alleviation of the local minima problem is proposed. The main feature of this technique is the usage of a two–stage transformation of the objective function to eliminate local minima, while preserving the global ones. Experiments indicate that combined with the Particle Swarm Optimizer method, the new algorithm is capable of escaping from local minima and effectively locate the global ones. Our experience is that the modified algorithm behaves predictably and reliably and the results were quite satisfactory. The function "Stretching" technique provides stable convergence and thus a better probability of success to the method with which it is combined.

Keywords: Global optimization, Derivative free methods, Evolutionary algorithms, Local minima, Particle swarm optimizer, Swarm intelligence.

1 INTRODUCTION

The global optimization problem is usually stated as finding the *global optimum* x^* of a real valued objective function $f : \mathcal{E} \rightarrow \mathbb{R}$. In many practical optimization problems the search is usually focused on locating the global minimizer, i.e. finding a point $x^* \in \mathcal{E}$ such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{E}, \tag{28.1}$$

where the compact set $\mathcal{E} \subset \mathbb{R}^D$ is a D–dimensional parallelepiped.

There are many Global Optimization (GO) methods developed so far to deal with this problem, which can be classified in two main categories: deterministic methods and probabilistic methods. Most of the deterministic methods apply heuristics, such as modifying the trajectory in trajectory methods or adding penalties in penalty–based methods, to help escape from local minima. On the other hand, probabilistic methods rely on probability to indicate whether the search should move away from the neighborhood of a local minimum, or not. Some of the most important GO strategies, see [11] for details, are listed below:

- *Adaptive partition and search strategies*, e.g., branch–and–bound algorithms, interval arithmetic based methods and Bayesian approaches [4, 7, 8, 12, 13, 14, 19].
- *Enumerative strategies* which are used for solving combinatorial problems, or certain "structured", e.g. concave, optimization problems [3, 7, 8].
- *"Globalized" local search methods* that apply a grid search or random search type global phase, and a local search algorithm [7, 13].
- *Heuristic strategies*, such as deflation, tunneling, filled function methods, approximate convex global underestimation, tabu search, etc. [5, 6, 7, 13].
- *Homotopy (parameter continuation) methods* and analogous approaches. These among others include pivoting algorithms and fixed point methods [7].
- *Passive (simultaneous) strategies*, such as uniform grid search, pure random search [7, 13, 19].

- *Successive approximation (relaxation) methods*, such as cutting plane, more general cuts, minorant construction approaches, certain nested optimization and decomposition strategies [3, 7, 13].
- *Trajectory methods* which are differential equation model–based, or path–following search strategies [7].
- *Adaptive stochastic search algorithms.* These include simulated annealing, random search, evolution and genetic algorithms [7, 10, 13, 17, 19].

Differently from other adaptive stochastic search algorithms, evolutionary computation techniques work on a set of potential solutions, which is called *population*, and find the optimal problem solution through cooperation and competition among the potential solutions. These techniques can often find optima in complicated optimization problems more quickly than traditional optimization methods. The most commonly used population–based evolutionary computation techniques, such as Genetic Algorithms and Artificial Life methods, are motivated from the evolution of nature and the social behavior.

It is worth noting that, in general, GO strategies possess strong theoretical convergence properties, and, at least in principle, are straightforward to implement and apply. Issues related to their numerical efficiency are considered by equipping GO algorithms with a "traditional" local optimization phase. Global convergence, however, needs to be guaranteed by the global–scope algorithm component which, theoretically, should be used in a complete, "exhaustive" fashion. These remarks indicate the inherent computational demand of the GO algorithms, which increases non–polynomially, as a function of problem–size, even in the simplest cases.

In practical applications, most of the above–mentioned methods can detect just *sub–optimal solutions* of the function f. In many cases these sub–optimal solutions are acceptable but there are applications where the optimal solution is not only desirable but also indispensable. Therefore, the development of robust and efficient GO methods is a subject of considerable ongoing research.

Recently, Eberhart and Kennedy [15, 16] proposed the *Particle Swarm Optimization* (PSO) algorithm: a new, simple evolutionary algorithm, which differs from other evolution–motivated evolutionary computation techniques in that it is motivated from the simulation of social behavior. Although, in general, PSO results good solutions, in high–dimensional spaces it stumbles on local minima.

In this paper we propose a new technique, named *Function "Stretching"*, and we show through simulation experiments that this strategy provides a way of escape from the local minima when PSO's convergence stalls.

The paper is organized as follows: the background of the PSO is presented in Section 2. The proposed technique of function "Stretching" for escaping local minima is derived in Section 3. In Section 4, the results of the improved PSO algorithm that incorporates the function "Stretching" strategy are presented and discussed, and finally conclusions are drawn in Section 5.

2 THE PARTICLE SWARM OPTIMIZATION METHOD

As already mentioned, PSO is different from other evolutionary algorithms. Indeed, in PSO the population dynamics simulates a bird flock's behavior where social sharing of information takes place and individuals can profit from the discoveries and previous experience of all other companions during the search for food. Thus, each companion, called *particle*, in the population, which is now called *swarm*, is assumed to "fly" over the search space in order to find promising regions of the landscape. For example, in the minimization case, such regions possess lower functional values than other visited previously. In this context, each particle is treated as a point in a D–dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles (companions).

There are many variants of the PSO proposed so far, after Eberhart and Kennedy introduced this technique [2, 9]. In our experiments we used a new version of this algorithm, which is derived by adding a new inertia weight to the original PSO dynamics [1]. This version is described in the following paragraphs.

First let us define the notation adopted in this paper: the i-th particle of the swarm is represented by the D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by the index g. The best previous position (the position giving the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

The particles are manipulated according to the equations

$$\begin{aligned} v_{id} &= w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id}), && (28.2) \\ x_{id} &= x_{id} + v_{id}, && (28.3) \end{aligned}$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$ and N is the size of population; w is the inertia weight; c_1 and c_2 are two positive constants; r_1 and r_2 are two random values in the range $[0, 1]$.

The first equation is used to calculate i-th particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current position, and, finally, the distance between swarm's best experience (the position of the best particle in the swarm) and i-th particle's current position. Then, following the second equation, the i-th particle flies toward a new position. In general, the performance of each particle is measured according to a predefined fitness function, which is problem–dependent.

The role of the inertia weight w is considered very important in PSO convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, 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 a reduction on the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions, thus a time decreasing inertia weight value is used.

From the above discussion it is obvious that PSO, to some extent, resembles evolutionary programming. However, in PSO, instead of using genetic operators, each individual (particle) updates its own position based on its own search experience and other individuals (companions) experience and discoveries. Adding the velocity term to the current position, in order to generate the next position, resembles the mutation operation in evolutionary programming. Note that in PSO, however, the "mutation" operator is guided by particle's own "flying" experience and benefits by the swarm's "flying" experience. In another words, PSO is considered as performing mutation with a "conscience", as pointed out by Eberhart and Shi [1].

3 THE FUNCTION "STRETCHING" TECHNIQUE

The local minima problem which is considered in this paper can be stated as follows. Let a point $\bar{x}$ such that there exists a neighborhood B

of $\bar{x}$ with

$$f(\bar{x}) \leq f(x), \quad \forall x \in \mathcal{B}. \tag{28.4}$$

This point is a local minimizer of the objective function and many GO methods get stuck in such points. In order to alleviate this problem the following two–stage transformation in the form of the original function $f(x)$ can be applied soon after a local minimum $\bar{x}$ of the function f has been detected:

$$G(x) = f(x) + \frac{\gamma_1}{2}\,\|x - \bar{x}\|\,(\mathrm{sign}(f(x) - f(\bar{x})) + 1), \tag{28.5}$$

$$H(x) = G(x) + \frac{\gamma_2\,(\mathrm{sign}\,(f(x) - f(\bar{x})) + 1)}{2\,\tanh\,(\,\mu(G(x) - G(\bar{x})))}, \tag{28.6}$$

where γ_1, γ_2 and μ are arbitrary chosen positive constants, and $\mathrm{sign}(\cdot)$ defines the well known three valued sign function.

The first transformation stage elevates the function $f(x)$ and makes disappear all the local minima which are located above $\bar{x}$. The second stage 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 unchanged.

It is worth noting that the sign function, which appears in the above transformation, can be approximated by the well known logistic function:

$$\mathrm{sign}(x) \approx \mathrm{logsig}(x) = \frac{2}{1 + \exp(-\lambda_1 x)} - 1 \simeq \tanh(\lambda_2 x),$$

for large values of λ_1 and λ_2. This sigmoid function is continuously differentiable and is widely used as a transfer function in artificial neurons.

At this point it is useful to provide an application example of the proposed technique in order to illustrate its effect. The problem considered is a notorious two dimensional test function, called the *Levy No. 5*:

$$\begin{aligned} f(x) &= \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \times \sum_{j=1}^{5} j\cos[(j+1)x_2 + j] + \\ &\quad + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2, \end{aligned} \tag{28.7}$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are about 760 local minima and one global minimum with function value $f^* = -176.1375$ located at $x^* = (-1.3068, -1.4248)$. The large number of local optimizers makes extremely difficult for any method to locate the global minimizer. In Fig. 28.1, the original plot of the *Levy No. 5* into the cube $[-2, 2]^2$ is shown.

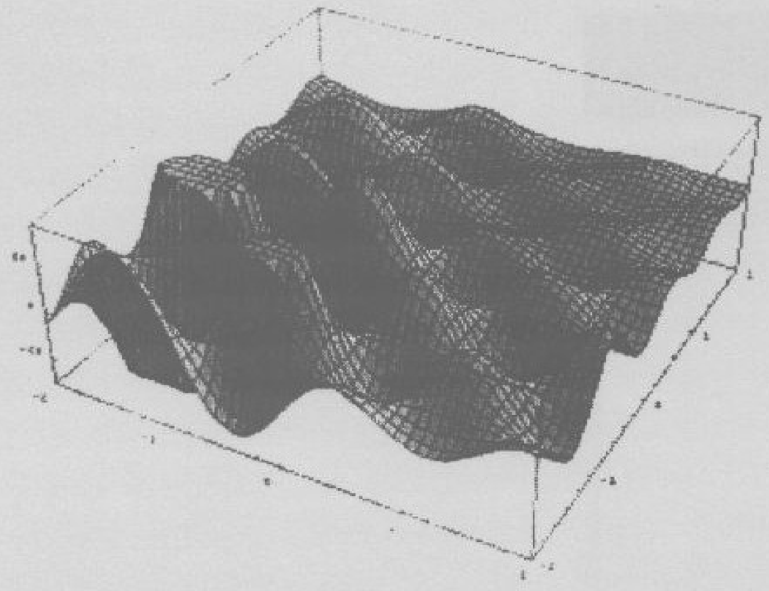

Figure 28.1 The original plot of the function *Levy No. 5*.

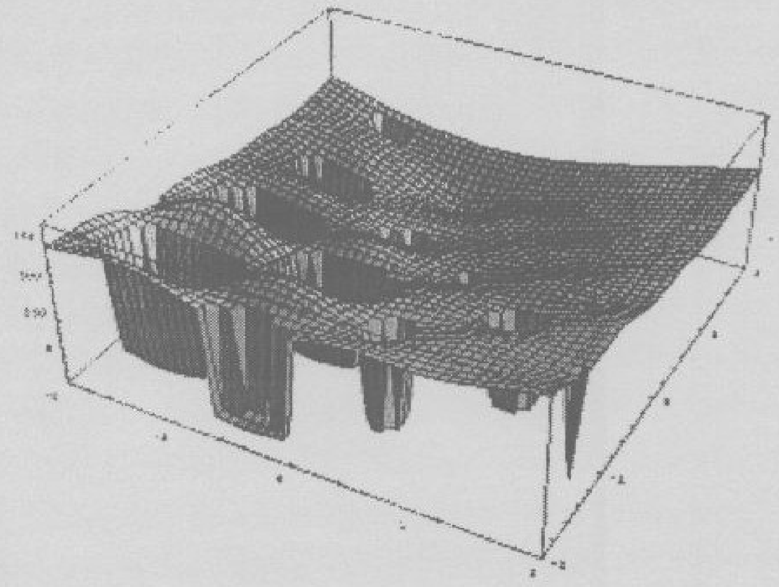

Figure 28.2 Plot of the *Levy No. 5* after the first stage of the "Stretching" technique.

After applying the transformation of Eq. 28.5 (first stage of function "Stretching") to the *Levy No. 5*, the new form of the function is shown in Fig. 28.2. As one can see, 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 been left unaffected.

In Fig. 28.3, the final landscape, derived after applying the second transformation stage to the *Levy No. 5*, is presented. It is clearly shown

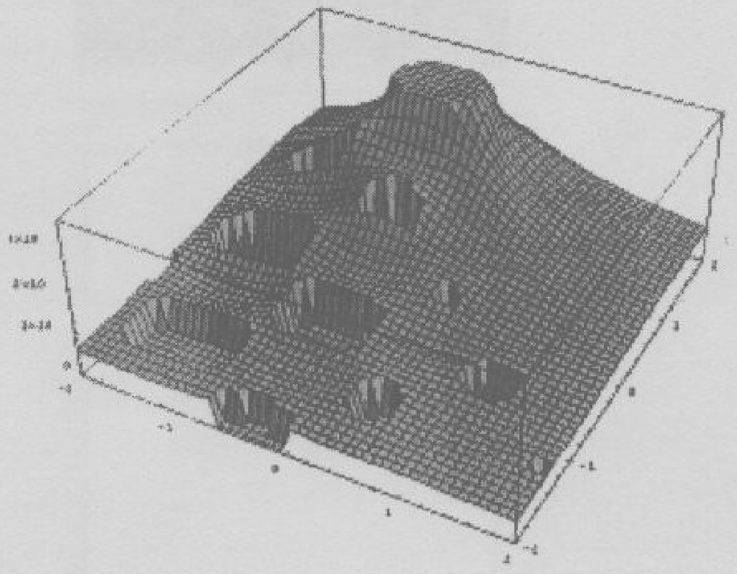

Figure 28.3 Plot of the *Levy No. 5* after the second stage of the "Stretching" technique.

how the whole neighborhood of the local minimum has been elevated; thus, the former local minimum has now turned to be a local maximum of the function. Details on the performance of the function "Stretching" technique on some well known test problems, as well as suggestions for selecting parameter values are presented in the next section.

4 EXPERIMENTAL RESULTS

In this section, we present results from testing a combination of the function "Stretching" technique and the PSO algorithm. This optimization strategy is named "Stretched" PSO (SPSO) and is initialized by applying the PSO algorithm for the minimization of the fitness function. In case the PSO converges to a local minimum, the function "Stretching" technique is applied to the original function and the PSO is re–initialized.

The performance of the SPSO has been evaluated by means of simulation runs in hard–optimization problems, like the minimization of the functions *Levy No. 5* and *Corana*, and in training Artificial Neural Networks (ANNs) for pattern classification problems, like the classification of the classical eXclusive–OR (XOR) patterns.

In all the simulations reported, the values of γ_1, γ_2 and μ were fixed: $\gamma_1 = 10000, \gamma_2 = 1$ and $\mu = 10^{-10}$. Default values for the parameters c_1 and c_2 have been used: $c_1 = c_2 = 0.5$. Although the choice of the parameter values seems to be not critical for the success of the method, 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 weights, since the particles' positions 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, has been found to work better than using a constant value. This is because large inertia weights help to find good seeds at the beginning of the search, while, later, small inertia weights facilitate a finer search.

The first test refers to the minimization of the *Levy No. 5* function that has been described in the previous section. The results of Table 28.1 have been obtained after 100 runs using the SPSO, with a swarm of size 20, initialized into the cube $[-2,2]^2$. The average performance is exhibited in terms of the mean value and standard deviation of the number of function evaluations, and the percentage of SPSO success. Information concerning the performance of the PSO algorithm with and without the function "Stretching" technique is also illustrated. As can be seen from

	"Stretched"	No "Stretched"	Overall
Mean Value	3854.2	1049.4	1245.8
Standard Deviation	1630.1	235.1	854.2
Success	7/7	93/93	100%

Table 28.1 Analysis of the results for the minimization of the Levy No.5 function.

Table 28.1, in 93 out of 100 cases PSO found the global minimum without any help, while in 7 cases it got stuck in a local minimum. In these cases function "Stretching" has been applied and the global minimum had finally been detected. Thus the success rate of PSO increased by 7%.

The second experiment concerns the minimization of the *Corana* function:

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

where $x_j \in [-1000, 1000]$, $d_j = 1, 1000, 10, 100$ and

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

SPSO has been tested in 100 simulation runs with a swarm consisted of 40 particles, initialized and constrained inside the hypercube $[-1,1]^4$. The success rate of SPSO for the minimization of this function is 100% (see Table 28.2), but the success percentage of the plain PSO is just 74%.

	"Stretched"	No "Stretched"	Overall
Mean Value	13704.6	2563.2	5460.0
Standard Deviation	7433.5	677.5	6183.8
Success	26/26	74/74	100%

Table 28.2 Analysis of the results for the minimization of the Corana function with constrained population.

In this case, the "Stetching" technique increased the success percentage by 26%, which is a significant improvement on the performance of the PSO.

Results for the unconstrained–swarm case are exhibited in Table 28.3. If free movement of population in the search space is allowed, the plain PSO has a success of 96%. Regarding the rest of the cases, in 2 of them the "Stetching" technique has been successfully applied leading the SPSO to reach the global minimum within the predefined limit of 2000 iterations. However, in 2 cases the SPSO didn't reach the desirable accuracy within the 2000 limit. This may suggest that in high–dimensional spaces the constrained movement of the swarm helps the algorithm to exhibit better performance; further investigation is necessary to extract useful conclusions.

	"Stretched"	No "Stretched"	Overall
Mean Value	72200.0	3219.1	8737.6
Standard Deviation	12269.1	590.1	19154.5
Success	2/4	96/96	98%

Table 28.3 Analysis of the results for the minimization of the Corana function without constraint.

In the third experiment an ANN has been trained using the SPSO to learn the XOR Boolean classification problem. The XOR function maps two binary inputs to a single binary output and the ANN that was trained to solve the problem had 2 linear input nodes, two hidden nodes with logistic activations and one linear output node. This task corresponds to the minimization of the following objective function [18]:

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

In the context of ANNs, the parameters $x_1, x_2, \ldots, x_9$ are called weights and are usually initialized in the interval $[-1, 1]$. It is well known from the 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 above–mentioned function presents a multitude of local minima. It is obvious from the results reported in Table 28.4 that the function "Stretching" technique helped to increase significantly the success percentage of the PSO, i.e. the success rate has been increased from 77% to 100%.

	"Stretched"	No "Stretched"	Overall
Mean Value	29328.6	1459.7	7869.6
Standard Deviation	15504.2	1143.1	13905.4
Success	23/23	77/77	100%

Table 28.4 Analysis of the results for the XOR problem.

5 CONCLUSIONS

A new technique, named function "Stretching", for the alleviation of the local minima problem was introduced. The main feature of this technique is that it uses a two–stage transformation of the fitness function to eliminate local minima, while preserving the global ones.

Experiments indicate that the PSO method when equipped with the proposed technique is capable to escape from local minima and locate the global one effectively. The function "Stretching" technique provides stable convergence and thus a better probability of success for the PSO.

Further work is focused on optimizing the performance of the proposed modification of the PSO algorithm. In addition, extensive testing on high–dimensional and more complex real–life optimization tasks is necessary to fully investigate the properties and evaluate the performance of the function "Stretching" technique.

References

[1] R.C. Eberhart and Y.H. Shi (1998). Evolving Artificial Neural Networks. *Proc. International Conference on Neural Networks and Brain*, Beijing, P.R. China.

[2] R.C. Eberhart, P.K. Simpson and R.W. Dobbins (1996). *Computational Intelligence PC Tools*, Academic Press Professional, Boston.

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

[4] E.R. Hansen (1992). Global Optimization Using Interval Analysis. Marcel Dekker, New York.

[5] F. Glover (1989). Tabu Search – part I, *ORSA Journal on Computing*, 1(3), 190–206.

[6] F. Glover (1990). Tabu Search – part II, *ORSA Journal on Computing*, 2(1), 4–32.

[7] R. Horst and P.M. Pardalos (1995). Handbook of Global Optimization. Kluwer Academic Publishers, London.

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

[9] J. Kennedy and R.C. Eberhart (1995). Particle Swarm Optimization. *Proc. IEEE International Conference on Neural Networks*, Piscataway, NJ, IV:1942–1948.

[10] Z. Michalewicz (1996). Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York.

[11] A. Neumaier (2000). On the server of the Computational Mathematics group at the University of Vienna, Austria, http://solon.cma.univie.ac.at/~neum/glopt.html, accessed 26/06/2000.

[12] A. Neumaier (1990). Interval Methods for Systems of Equations. Cambridge University Press, Cambridge.

[13] J.D. Pintér (1996). Global Optimization in Action. Kluwer Academic Publishers, Dordrecht, Boston, London.

[14] H. Ratschek and J.G. Rokne (1988). New Computer Methods for Global Optimization. Ellis Horwood, Chichester.

[15] Y.H. Shi and R.C. Eberhart (1998). Parameter Selection in Particle Swarm Optimization. *Proc. Annual Conference on Evolutionary Programming*, San Diego.

[16] Y.H. Shi and R.C. Eberhart (1998). A Modified Particle Swarm Optimizer, *Proc. IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska.

[17] P.J.M. Van Laarhoven and E.H.L. Aarts (1987). Simulated Annealing: Theory and Applications. Kluwer Academic Publishers, London.

[18] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos and G.D. Magoulas (2000), A class of gradient unconstrained minimization algorithms with adaptive stepsize. *Journal of Computational and Applied Mathematics*, 114, 367–386.

[19] A.A. Zhigljavsky (1991). Theory of Global Random Search. Kluwer Academic Publishers, London.