Numerical Algorithms **34:** 393–403, 2003.

Evolutionary operators in global optimization with dynamic search trajectories

E.C. Laskari, K.E. Parsopoulos and M.N. Vrahatis

Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece
E-mail: {elena,kostasp,vrahatis}@math.upatras.gr

Received 13 November 2001; accepted 5 September 2003

One of the most commonly encountered approaches for the solution of unconstrained global optimization problems is the application of multi-start algorithms. These algorithms usually combine already computed minimizers and previously selected initial points, to generate new starting points, at which, local search methods are applied to detect new minimizers. Multi-start algorithms are usually terminated once a stochastic criterion is satisfied. In this paper, the operators of the Differential Evolution algorithm are employed to generate the starting points of a global optimization method with dynamic search trajectories. Results for various well-known and widely used test functions are reported, supporting the claim that the proposed approach improves drastically the performance of the algorithm, in terms of the total number of function evaluations required to reach a global minimizer.

Keywords: global optimization, dynamic search trajectories, differential evolution, hybrid methods, numerical algorithms

AMS subject classification: 65K05, 65K10, 68T20

1. Introduction

Optimization problems are frequently encountered in engineering applications. In numerous applications it is required to detect the globally optimal solutions. Economics, finance, networks and mechanical design, molecular biology, image processing and, chemical engineering are just some of the scientific fields where such applications can be met. Consequently, the development of efficient global optimization (GO) algorithms, remains up-to-date an active area of research [3].

In general, the unconstrained GO problem (without loss of generality we consider only the minimization case) for a continuously differentiable objective function $F : X \subset \mathbb{R}^n \to \mathbb{R}$, can be stated as finding a point $x^* \in X$, such that,

$$F^* = F(x^*) = \min_{x \in X} F(x), \tag{1}$$

where X is a compact set and $F \in C^1$. This is an ill-posed problem in the sense that a lower bound for F^* cannot be estimated after any finite number of evaluations, unless the function F satisfies the Lipschitz condition with known Lipschitz constant L [2]:

$$\left\| F(x) - F(y) \right\| \leqslant L\|x - y\|, \quad \forall x, y \in X.$$

Unfortunately, the value of L is rarely known in most applications, rendering the algorithms which are heavily dependent on such assumptions, of limited applicability. Thus, most of the GO methods are of heuristic nature, i.e. the lowest minimum among several detected minima is considered as the estimate of F^*. In this sense, GO should be considered as a problem of locating sufficiently low local minima [2].

One of the most common approaches for solving GO problems is the *multi-start* technique. According to this technique, several starting points are sampled and a local search procedure is applied to each one. This method has been widely discussed [1,16], especially with respect to its regions of attraction. The algorithm considered in this study is the *Multi-start Global Minimization Algorithm with Dynamic Search Trajectories*, developed by Snyman and Fatti [13] (henceforth denoted as SF). The SF algorithm probes the search space using trajectories that are generated by integrating the differential equation

$$\ddot{x}(t) = -\nabla F\big(x(t)\big), \tag{2}$$

which describes the motion of a unit mass particle in a conservative force field, where $F(x(t))$ represents the potential energy of the particle. The trajectories are modified in a manner that ensures convergence to a lower minimum, compared to the one obtained using a common gradient descent local search method. Therefore, the region of convergence of the global minimum is increased. The global scope phase of the algorithm involves randomly selecting the starting point of the first trajectory, and combining already computed minimizers as well as previously selected initial points to generate the starting points for the trajectories considered in later steps.

Alternatively, the GO problem can be tackled by modern stochastic methods, like *Evolutionary Algorithms* [5,11] and *Differential Evolution* [14,15] (DE). In this context, a population of potential solutions is employed to probe the search space synchronously. The individuals of the population are combined by applying *recombination* operators, and they are perturbed through *mutation* operators. These operators give rise to a new population consisting of the most promising solutions, at each iteration of the algorithm, by selecting the best individuals, i.e. the individuals with the lowest function values. The stochastic combination of potential solutions is one of the main characteristics of such algorithms and a crucial aspect of their performance.

In this paper, a hybrid algorithm that combines the SF and DE algorithms is introduced. Specifically, the recombination and mutation operators of the DE algorithm are utilized to generate the starting points of the trajectories of the SF algorithm. Experimental results on various widely used test functions support the claim that the proposed approach improves significantly the performance of the plain SF algorithm.

The rest of the paper is organized as follows: the SF and the DE algorithms are briefly described in sections 2 and 3, respectively. The proposed approach is presented in section 4 and experimental results are presented in section 5. The paper closes with conclusions and ideas for future work in section 6.

2. The multi-start global minimization algorithm with dynamic search trajectories

SF is a multi-start numerical algorithm that utilizes trajectories derived by (2), for tackling GO problems. Assuming $x(0) = x_0$ and $\dot{x}(0) = 0$ to be the initial conditions, multiplying (2) by $\dot{x}(t)$, and integrating from time 0 to time t, implies the energy conservation relationship:

$$\frac{1}{2}\left\|\dot{x}(t)\right\|^2 + F\big(x(t)\big) = \frac{1}{2}\left\|\dot{x}(0)\right\|^2 + F\big(x(0)\big) = F(x_0). \tag{3}$$

The first term on the left-hand side of (3) represents the kinetic energy, whereas the second term represents the potential energy of a particle of unit mass, at any time instant t. Obviously, the particle will start moving in the direction of the steepest descent and its kinetic energy will continue to increase (and thus F will decrease) as long as the following relation holds:

$$-\nabla F^{\mathrm{T}}\dot{x} > 0.$$

If the descent condition is not met along the generated path then the magnitude of the velocity decreases and its direction changes in a way that ensures motion towards a local minimizer [12]. If more than one local minimizers exist and the global minimizer is required, a potential strategy is to let the particle continue its motion, while recording the point x_m at which the minimum along this path occurs, as well as the corresponding quantities $\dot{x}_m$ and F_m. In such cases, the path may surmount a ridge of height F_r, where $F_m < F_r < F(x_0)$, detecting probably the basin of attraction of a lower minimum. On the other hand, the trajectory shall be stopped before it retraces itself in an indefinite motion. A proper termination condition is to stop the trajectory once it reaches a point with function value close to $F(x_0)$, while at the same time the relation $\nabla F^{\mathrm{T}}\dot{x} > 0$ is satisfied, i.e. the movement is uphill [13]. Auxiliary trajectories are then generated to better utilize the knowledge of x_m and its associated quantities. The initial point of the second trajectory can be selected as:

$$x_0^2 = \frac{1}{2}\big(x_0^1 + x_b^1\big), \tag{4}$$

with initial velocity $\dot{x}_0^2 = \frac{1}{2}\dot{x}_m^1$, where the superscripts denote the trajectory's number, and $x_b^1 \leftarrow x_m^1$. The termination criterion of this trajectory is the same as before. If $F(x_m^2) < F(x_b^1)$ then $x_b^2 \leftarrow x_m^2$, otherwise $x_b^2 \leftarrow x_b^1$. Further trajectories are generated by repeating the procedure, if it is necessary.

The integration method that is used for the generation of the trajectories is the *Leap-Frog* algorithm. Given x_0, $\dot{x}_0$, and a time step Δt, the Leap-Frog scheme is defined as follows:

$$x_{k+1} = x_k + \dot{x}_k \Delta t,$$
$$\dot{x}_{k+1} = \dot{x}_k - \nabla F(x_{k+1}) \Delta t,$$

where $k = 0, 1, 2, \ldots$. The initial velocity over the first step is taken as $\dot{x}_0 = -\nabla F(x_0)\Delta t/\gamma$, where γ is usually set equal to 2. The time step, Δt, is chosen so as to ensure descent at the first step. If this condition is not met, Δt is successively halved and the trajectory is restarted until descent is achieved.

In the algorithm given in [13], the termination condition for a trajectory is

$$F\big(x(t)\big) - F_T > \alpha\big(F(x_0) - F_T\big),$$

where F_T is the lowest available value of F, and α is a relaxation parameter almost equal but less than one. A trajectory is terminated on the uphill, provided that

$$T = \frac{1}{2}\big\|\dot{x}(t)\big\|^2 < (1-\alpha)\big(F(x_0) - F_T\big).$$

This prevents the trajectory from becoming almost endless. A typical choice for α is $\alpha = 0.95$ [13].

For each starting point, the minimization procedure is provisionally ended if, for some x_k, the inequality

$$\big\|\nabla F(x_k)\big\| < \varepsilon$$

holds for some prescribed small positive number $\varepsilon > 0$. If $F(x_k)$ is the smallest function value obtained so far for the corresponding starting point, then the procedure terminates with final values $x_{\mathrm{f}} = x_k$ and $F(x_{\mathrm{f}})$. If the smallest function value occurs at another point $x_q \neq x_k$, then the procedure is restarted by using the value of x_q as the starting point. In order to prevent excessive computation, an upper limit k_m is posed on the number of integration steps for each starting point.

The global scope component of the algorithm involves a stochastic criterion that reports the probability of the lowest obtained minimum to be the global one. To this end, let R_j denote the region of convergence of a local minimum $\widehat{F}_j$ in the search space, and α_j denote the probability that a randomly selected point falls into R_j. If R^* and α^* are the corresponding quantities of the global minimum, then a usual assumption is

$$\alpha^* = \Pr\big[R^*\big] = \max_j \alpha_j. \tag{5}$$

Then, the following theorem is proved [13].

Theorem 1. Let r be the number of sample points falling within the region of convergence of the current overall minimum $\widetilde{F}$, after n points have been sampled. Then,

under the assumption given in (5) and a statistically noninformative prior distribution, the probability that $\widetilde{F}$ be equal to F^* satisfies the following relationship:

$$\Pr\left[\widetilde{F} = F^*\right] \geqslant q(n, r) = 1 - \frac{(n+1)!(2n-r)!}{(2n+1)!(n-r)!}.$$

For each starting point, the quantity $q(n, r)$ is compared with a prescribed value q^*, and the algorithm is terminated if and only if $q(n, r) > q^*$. A commonly used value is $q^* = 0.99$ [13].

An analytic description of all the aforementioned concepts, as well as the pseudocode of the SF algorithm can be found in [13]. In the next section, the DE algorithm is briefly described.

3. The differential evolution algorithm

The DE algorithm was developed by Storn and Price [14,15]. It is a parallel direct numerical search method, which utilizes NP, D-dimensional parameter vectors $x_{i,G}$, $i = 1, 2, \ldots, NP$, as a population to probe the search space. The index G denotes the generation (iteration) number of the algorithm. The initial population is taken to be uniformly distributed in the search space. At each generation, the *mutation* and *recombination* operators are applied on the individuals, and a new population arises. Then, *selection* takes place, and the corresponding individuals from both populations compete to comprise the next generation.

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

$$v_{i,G+1} = x_{r_1,G} + K(x_{r_2,G} - x_{r_3,G}),$$

where $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ are mutually different random indices and $K \in (0, 2]$. The indices r_1, r_2, r_3, also need to differ from the current index i and, consequently, mutation can be applied only if NP is at least 4.

Following the mutation phase, the *recombination* operator is applied on the population. Thus, a *trial vector*

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

is generated, where

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

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

To decide whether the vector $u_{i,G+1}$ will be a member of the population of the next generation, it is compared to the initial vector $x_{i,G}$, and

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{if } F(u_{i,G+1}) < F(x_{i,G}), \\ x_{i,G}, & \text{otherwise,} \end{cases}$$

where F is the function under consideration.

The procedure described above is considered as the standard variant of the DE algorithm, and it is denoted as DE/rand/1/bin. Different kinds of mutation and recombination operators have also been applied on minimization problems with promising results [15]. A parallel version of the DE algorithm is reported in [10]. In the next section, the proposed (hybrid) algorithm is described.

4. The proposed algorithm

The combination operator used by Snyman and Fatti in the SF algorithm, to generate the starting points of the auxiliary trajectories, is very simple as exhibited in (4) [13]. The central idea of the proposed algorithm is to provide the SF algorithm with a more sophisticated recombination technique so as to increase its success rate and its overall performance. The DE mutation and recombination operators have proved very promising in developing efficient GO techniques and have been selected for this purpose. Thus, a population of already computed minimizers and previously selected initial points is maintained. The best individual of the population, after the application of recombination and mutation, is selected as the initial point for the new auxiliary trajectory. Then, a new trajectory is generated, and its best point is considered as a potential member of the population. The DE operators are re-applied to determine the initial point of the next auxiliary trajectory. The process is thoroughly described in the rest of this section.

Initially, a randomly selected starting point x_0^1 is introduced into an (initially empty) population P, and the first trajectory is generated using the standard SF algorithm. The best point x_m^1 of this trajectory, i.e. the point of the trajectory with the smallest function value, is also included into the maintained population P. The two available points x_0^1 and x_m^1 are then combined, according to (4), to give the starting point x_0^2 of the second trajectory. The SF operator is employed at this initial stage because the DE mutation and recombination operators can be applied only on populations of size $popsize \geqslant 4$. Consequently, they cannot be applied at this stage since the population consists of only two points.

The second trajectory is generated by starting from x_0^2. The initial point x_0^2 as well as the best point x_m^2 of the second trajectory are also included in P. It should be noted here that a point is introduced into the population P, if and only if it is not already a member of P.

Beyond this step, the population consists of four points, and the DE operators are applicable. Thus, mutation and recombination are applied on the population and four new points are generated. Then, the selection phase takes place, and a new population P' is formed. The best point of P' is selected as the initial point for the new (third)

Table 1
The modified trajectory generation procedure.

Step 1.	Given x_0^1, compute the trajectory T^1 by solving (2) subject to $\dot{x}_0^1 = 0$. Record x_m^1, $\dot{x}_m^1$, F_m^1. Let $x_b^1 \leftarrow x_m^1$ and $i \leftarrow 2$. Insert x_0^1 and x_m^1 in the population P. Set *popsize* $\leftarrow 2$.
Step 2.	If *popsize* < 4 then compute the trajectory T^i with $x_0^i = \frac{1}{2}(x_0^{i-1} + x_b^{i-1})$, $\dot{x}_0^i = \frac{1}{2}\dot{x}_m^{i-1}$, and set $P = P \cup \{x_0^i\}$. Otherwise apply the mutation and recombination operators on the population P, to find the point x_0^i, and take $\dot{x}_0^i = \frac{1}{2}\dot{x}_m^{i-1}$. Record x_m^i, $\dot{x}_m^i$, F_m^i.
Step 3.	If $F(x_m^i) < F(x_b^{i-1})$, then set $x_b^i \leftarrow x_m^i$ and if x_b^i differs from all the points into the population P, then set $P = P \cup \{x_b^i\}$ and *popsize* $\leftarrow$ *popsize* $+ 1$. Otherwise set $x_b^i \leftarrow x_b^{i-1}$.
Step 4.	If *popsize* $>$ *maxps*, then evaluate the population and take the *maxps* better points as the new population P.
Step 5.	Set $i \leftarrow i + 1$ and go to step 2.

trajectory, and it is included in P. The remaining points of P' are of no interest and they are omitted. The whole procedure is repeated as long as the SF algorithm generates new trajectories. Obviously, this procedure results in a huge population, since in many problems hundreds or even thousands of trajectories may be required. To address this problem, a threshold *maxps* is posed on the maximum number of points included in P. If there is a potential new member of the population but *popsize* is equal to *maxps*, i.e. the population's size has already reached its maximum value, then the new point is included in the population and all the individuals are sorted according to their function values. The individual with the worst (highest) function value is excluded from the population, in order to reduce its size to *maxps*.

Following the technique described above, the *Procedure 2.1* of the SF algorithm, which is described in [13], is modified as reported in table 1. The rest of the SF algorithm is unaltered and it is applied following the guidelines provided by Snyman and Fatti in [13].

In the next section, experimental results from the application of the proposed approach as well as the plain SF algorithm are reported and compared.

5. Experimental results

Ten well-known test functions of various dimensions have been used to investigate the performance of the proposed algorithm (denoted as Hybrid SF), and to compare it with that of the plain SF algorithm. For each test function, 100 runs were performed for four different sets of the parameters *CR* and *K* of the DE algorithm. The name and dimension of each test function; the range in which the initial point x_0^1 of the first trajectory was randomly selected; the maximum size of the population; as well as the

Table 2
The results for the Rosenbrock function.

Function: Rosenbrock [6], dimension: 2 Initial interval: [−3, 3], maximum population size: 10				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	10156.38	2249.22	77.85%
0.5	0.5	8486.66	2349.46	72.32%
0.9	0.1	8299.88	2664.12	67.90%
0.5	0.9	5408.02	2334.06	56.84%

Table 3
The results for the Freudenstein–Roth function.

Function: Freudenstein–Roth [6], dimension: 2 Initial interval: [−1, 1], maximum population size: 10				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	182163.08	16500.32	90.94%
0.5	0.5	158424.40	3294.64	97.92%
0.9	0.1	170561.80	18457.88	89.18%
0.5	0.9	139451.04	17594.44	87.38%

Table 4
The results for the Hellical Valley function.

Function: Hellical Valley [6], dimension: 3 Initial interval: [−2, 2], maximum population size: 10				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	3800.36	3076.74	19.04%
0.5	0.5	3872.14	3106.98	19.76%
0.9	0.1	4204.72	2944.46	29.97%
0.5	0.9	3843.52	3082.48	19.80%

obtained results are reported in tables 2–11. The results indicate the mean values of the required function evaluations for the plain SF and the proposed Hybrid SF algorithm, as well as the percentage of the improvement in the mean number of function evaluations that is gained by using the Hybrid SF. The values of the parameters of the SF algorithm that were used in all experiments, were identical to the values reported in [13], namely, $\gamma = 2$, $\alpha = 0.95$, $\varepsilon = 10^{-3}$, $q^* = 0.99$, $\omega = 10^{-2}$ and $k_m = 5000$.

The reported results support the claim that the proposed hybrid approach outperforms the plain SF algorithm. The improvement in performance in some cases exceeds 90%. Moreover, there were cases where the proposed algorithm converged, while the plain SF algorithm failed. Furthermore, the proposed algorithm can be easily implemented, through a minor modification of the plain SF algorithm.

Table 5
The results for the Levy No. 8 function.

Function: Levy No. 8 [4], dimension: 3 Initial interval: [−1, 1], maximum population size: 10				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	28867.75	17458.75	39.53%
0.5	0.5	29221.50	17182.00	41.17%
0.9	0.1	21078.00	16605.00	18.18%
0.5	0.9	25434.50	18064.00	28.39%

Table 6
The results for the Wood function.

Function: Wood [6], dimension: 4 Initial interval: [−3, 3], maximum population size: 10				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	30667.64	5679.84	81.48%
0.5	0.5	26058.16	5679.36	78.21%
0.9	0.1	28224.32	6044.60	78.58%
0.5	0.9	25842.24	5750.80	77.75%

Table 7
The results for the Watson function.

Function: Watson [6], dimension: 6 Initial interval: [−0.5, 0.5], maximum population size: 5				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	22176.10	14984.20	32.43%
0.5	0.5	22747.80	14985.80	34.12%
0.9	0.1	32634.40	13612.30	58.29%
0.5	0.9	38189.00	17748.00	53.53%

Table 8
The results for the Hyper-Ellipsoid function.

Function: Hyper-Ellipsoid [15], dimension: 6 Initial interval: [−1, 1], maximum population size: 5				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	1742.42	1436.68	17.55%
0.5	0.5	2049.52	1450.60	29.22%
0.9	0.1	1997.02	1415.36	29.13%
0.5	0.9	1750.68	1453.86	16.95%

Table 9
The results for the 6-dimensional Rastrigin function.

Function: Rastrigin 6D [15], dimension: 6 Initial interval: [−1, 1], maximum population size: 5				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	14013.84	3742.52	73.29%
0.5	0.5	16721.84	4398.80	73.69%
0.9	0.1	16316.04	5182.68	68.24%
0.5	0.9	17709.92	4199.28	76.29%

Table 10
The results for the 10-dimensional Rastrigin function.

Function: Rastrigin 10D [15], dimension: 10 Initial interval: [−0.5, 0.5], maximum population size: 5				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	7079.10	3528.70	50.15%
0.5	0.5	7486.25	4477.75	40.19%
0.9	0.1	8043.60	3878.70	51.78%
0.5	0.9	7843.30	2706.80	65.49%

Table 11
The results for the 15-dimensional Rastrigin function.

Function: Rastrigin 15D [15], dimension: 15 Initial interval: [−0.5, 0.5], maximum population size: 5				
CR	*K*	Plain SF	Hybrid SF	Imrovement
0.1	0.9	21309.90	9576.00	55.06%
0.5	0.5	18991.20	9338.90	50.83%
0.9	0.1	25605.30	9879.00	61.42%
0.5	0.9	19449.10	3984.70	79.51%

6. Conclusions and future research directions

A hybrid modification of the global optimization algorithm with dynamic search trajectories, that utilizes differential evolution operators for the selection of the initial points of the auxiliary trajectories, has been introduced. The hybrid algorithm exhibits superior performance relative to the plain algorithm, on various well-known test functions. Specifically, the mean number of function evaluations required to detect the global minimizer is significantly reduced. In many cases the efficiency of the algorithm is also increased and the number of the generated trajectories is reduced.

Future work will include the combination of the hybrid approach with techniques for avoiding local minima, such as the "stretching" technique [7–9], and the utilization of other variants of the DE operators as well as operators of different evolutionary and

swarm intelligence algorithms that might further improve the algorithm's performance. The performance of parallel approaches using the parallel DE [10] will be also considered.

References

[1] L.C.W. Dixon, J. Gomulka and G.P. Szegö, Reflections on the global pptimization problem, in: *Optimization in Action*, ed. L.C.W. Dixon, 1975, pp. 29–54.

[2] A.O. Griewank, Generalized descent for global optimization, J. Optim. Theory Appl. 34 (1981) 11–39.

[3] R. Horst, P.M. Pardalos and N.V. Thoai, *Introduction to Global Optimization* (Kluwer Academic, Dordrecht, 1995).

[4] A. Levy, A. Montalvo and S. Gomez, *Topics in Global Optimization* (Springer, Berlin, 1981).

[5] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1999).

[6] J.J. More, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, ACM Trans. Math. Software 7(1) (1981) 17–41.

[7] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas and M.N. Vrahatis, Objective function "stretching" to alleviate convergence to local minima, Nonlinear Anal. TMA 47(5) (2001) 3419–3424.

[8] K.E. Parsopoulos and M.N. Vrahatis, Modification of the particle swarm optimizer for locating all the global minima, in: *Artificial Neural Networks and Genetic Algorithms*, eds. V. Kurkova et al. (Springer, Berlin, 2001) pp. 324–327.

[9] K.E. Parsopoulos and M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1(2/3) (2002) 235–306.

[10] V.P. Plagianakos and M.N. Vrahatis, Parallel evolutionary training algorithms for "hardware-friendly" neural networks, Natural Computing 1(2/3) (2002) 307–322.

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

[12] J.A. Snyman, A new and dynamic method for unconstrained minimization, Appl. Math. Modelling 6 (1982) 449–462.

[13] J.A. Snyman and L.P. Fatti, A multi-start global minimization algorithm with dynamic search trajectories, J. Optim. Theory Appl. 54(1) (1987) 121–141.

[14] R. Storn, Sytem design by constraint adaptation and differential evolution, IEEE Trans. Evolutionary Comput. 3(1) (1999) 22–34.

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

[16] A. Törn and A. Žilinskas, *Global Optimization* (Springer, Berlin, 1989).