A Novel Global Optimization Technique for High Dimensional Functions

Crina Grosan,[1,†] Ajith Abraham[2,3,*]
[1]*Department of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania*
[2]*Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence, Norwegian University of Science and Technology, Trondheim, Norway*
[3]*Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Auburn, WA 98071*

Several types of line search methods are documented in the literature and are well known for unconstraint optimization problems. This paper proposes a modified line search method, which makes use of partial derivatives and restarts the search process after a given number of iterations by modifying the boundaries based on the best solution obtained at the previous iteration (or set of iterations). Using several high-dimensional benchmark functions, we illustrate that the proposed line search restart (LSRS) approach is very suitable for high-dimensional global optimization problems. Performance of the proposed algorithm is compared with two popular global optimization approaches, namely, genetic algorithm and particle swarm optimization method. Empirical results for up to 2000 dimensions clearly illustrate that the proposed approach performs very well for the tested high-dimensional functions.

1. INTRODUCTION

The objective of global optimization is to find the globally best solution of (possibly nonlinear) models, in the (possible or known) presence of multiple local optima. Formally, global optimization seeks global solution(s) of a constrained optimization model.

In this paper, we assume the following: given a function $f: \Omega \subset \Re^n \rightarrow \Re$:

— the optimization problem is unconstrained;
— f is nonlinear, continuous, and twice differentiable;
— the feasible region Ω is given by a set of lower and upper bounds on each variable, that is, $\Omega = \{\min_i \leq x_i \leq \max_i, i = 1, \ldots, n\}$;
— and that the global minima lies within the interior of Ω.

*Author to whom all correspondence should be addressed: e-mail: ajith.abraham@ieee.org.
†e-mail: cgrosan@cs.ubbcluj.ro.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 24, 421–440 (2009)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20343

As mentioned by Gergel[1] and as evident from other well-established works,[2–8] these class of problems are of substantial interest. Even though there is a huge amount of work dealing with global optimization, there are still not many powerful techniques to be used for dense high-dimensional functions. One of the main reasons is the high-computational cost involved. Usually, the approaches are computationally expensive to solve the global optimization problem reliably. Very often, it requires many function evaluations and iterations and arithmetic operations within the optimization code itself. For practical optimization applications, the evaluation of f is often very expensive to compute and large number of function evaluations might not be very feasible.[9]

Because of the practical demands, there were some attempts in trying to use different methods to solve high-dimensional global optimization problems. One solution is to use parallel global optimization methods.[10] The parallel algorithm for global optimization proposed by Hofinger et al.[11] can approach functions having up to 512 dimensions (according to Ref. 11, it took 3 days for the first 10 iterations). The parallel particle swarm algorithm proposed by Schutte et al.[12] works well for some standard functions (e.g., Griewank and Corona test functions) and was tested for 128 dimensions.

Among the existing metaheuristics for global optimization, we used two popular approaches for comparisons with the line search restart (LSRS) approach. The selected metaheuristics, namely, genetic algorithms (GA) and particle swarm optimization (PSO) are well established and found highly successful and suitable for several classes of optimization problems. As evident from the scientific literature, GA and PSO were improved and adapted in several ways so as to obtain some reasonable results. In spite of all the success stories for several applications and revisions proposed during the last several years, these techniques are still not very much suitable for large-scale global optimization problems involving high dimensions.

There are impressive number of papers reporting results in applying these techniques—either in their original form or in several improved and hybrid versions.[13–28,42–49]

The paper is structured as follows: in Section 2 the modified LSRS technique is presented. In Section 3 the two computational techniques used for comparisons, namely, GA and PSO, are briefly introduced. Section 4 is dedicated to the numerical experiments. We illustrate the performance for higher number of dimensions (between 50 and 2000). Section 5 includes conclusions and further work ideas.

2. MODIFIED LINE SEARCH TECHNIQUE

Line search is a well-established optimization technique. The modification proposed in this paper for the standard line search technique refers to step setting and also the incorporation of a restart approach. To fine tune the performance, the first partial derivative of the function to optimize is also made use of. The proposed three modifications are summarized below and will be described in details in the subsequent sections:

1. The first modification refers to the inclusion of multistart principle within the search process.
2. The second modification is related to the setting of the direction and step.
3. The third modification refers to the restarting of the line search method.

After a given number of iterations, the process is restarted by reconsidering other arbitrary starting point (or other multiple arbitrary starting points), which is generated by taking into account the results obtained at the end of previous set of iterations. In the following subsection, the algorithm is presented in a structured form.

2.1. Generation of the Starting Points

It is known that line search techniques uses a starting point. There are also versions that allow the usage of multiple points, and the search will start separately from each of these points. In the proposed approach, multiple arbitrary starting points are used.

For a function of n variables $(x_1, x_2, \ldots, x_n)$ and the domain of definition given by

$$[\min_1,\ \max_1] \times [\min_2,\ \max_2] \times [\min_n,\ \max_n]$$

where $[\min_i, \max_i]$ is the domain of ith variable, the starting point x_i is randomly generated between the considered limits given by $[\min_i, \max_i]$.

2.2. Direction and Step Settings

Initially, we performed several experiments to set an adequate value for the direction. We used the standard value $+1$ or -1 and, for some functions the value -1 was favorable for very good results. We also performed some experiments by setting the direction value as being a random number between 0 and 1. Using the random number helped to obtain overall very good performance for the entire considered test functions. But usage of the value -1 for direction obtains almost the same performance similar to that obtained with a random value. So, either of these values (the random one and the value -1) may be used for better performance.

The step is set as follows:

$$\alpha_k = 2 + \frac{3}{2^{k^2+1}},$$

where k refers to the iteration number.

The *Line_search()* technique may be written as follows:

Line_search()
et $k = 1$ (Number of iterations)
Repeat
  for $i = 1$ to *No of starting points*
    for $j = 1$ to No of variables

p_k= -1; //or p = random;
$\alpha_k = 2 + \frac{3}{2^{k^2+1}}$
$x_{ij}^{k+1} = x_{ij}^k + p_k \cdot \alpha_k$
endfor
if $f(x_i^{k+1}) > f(x_i^k)$ then $x_i^{k+1} = x_i^k$.

Endfor
k = k+1
Until k = Number of iterations (apriori known).

Remarks

(i) The condition: if $f(x_i^{k+1}) > f(x_i^k)$ then $x_i^{k+1} = x_i^k$ allows us to move to the new generated point only if there is an improvement in the quality of the function.
(ii) Number of iterations for which line search is applied is apriori known and is usually a small number. In our experiments, we set the number of these iterations to 10 even for high-dimensional problems.
(iii) When restarting the line search method (after the insertion of the restart technique), the value of iterations number will again start from 1 (this should not be related to the value of α after the first set of iterations (and after each of the following ones)).

2.3. Restart Insertion

To restart the algorithm, the best result obtained in the previous set of iterations is taken into account and by following the steps given below:

1. Among all the considered points, the solution for which the objective function is obtaining the best value is selected. If there are several such solutions, one of them is randomly selected. This solution will be a multidimension point in the search space and denoted by "x" for an easier reference.
2. For each dimension i of the point x, the first partial derivative with respect to this dimension is calculated. This means the gradient of the objective function is calculated, which is denoted by g. Taking this into account, the bounds of the definition domain for each dimension is recalculated as follows:
   if $g_i = \dfrac{\partial f}{\partial x_i} > 0$ then $\max_i = x_i$;
   if $g_i = \dfrac{\partial f}{\partial x_i} < 0$ then $\min_i = x_i$
3. The search process is restarted by reinitializing a new set of arbitrary points (using Generate starting points () procedure) but between the newly obtained boundaries (between the new $\max_i$ or new $\min_i$).

2.4. General Line Search with Restart Procedure

The line search method presented in the previous subsections combined with the restart technique as described above is expressed by using the following pseudo code:

General Line Search with Re-Start

Set t = 1;
Repeat
Generate starting points (max, min);
Line_search (k);
Re_start (new values for max_i and/or min_i for each dimension will be obtained);
t = t+1;
Until t = Number of applications of the re-start technique (a priori known).
Select the solution x* for which the value of the objective function is minimum.
Print x*.

3. GENETIC ALGORITHMS AND PARTICLE SWARM OPTIMIZATION ALGORITHMS

Because of the fact that these two techniques are now well-established and very popular among the artificial intelligence/problem solving community, some of the fundamental ideas are presented in this section with a focus on the algorithms structure, which would help the reader to follow the experiments.

3.1. Genetic Algorithms

Genetic algorithms are a population-based search technique, using principles from biological evolution that are transposed in a computational scheme. This technique was proposed by Holland[29] in 1975 and was developed further and successfully applied in various domains. The basic components are solution representation, population initialization, fitness function, and genetic operators (such as selection, crossover, and mutation). The main idea is as follows: the genetic pool of a given population potentially contains an approximate solution, to a given problem. During reproduction and crossover, new genetic combination occurs and there is chance to obtain a better solution (either than the ones that were combined or mutated or better than all the existing ones in that pool). By repeating several times, these recombinations between the potential solutions for the problem, usually a very good approximation of the solution is obtained.[30–32]

Genetic Algorithm Scheme

Set t = 1;
Randomly initialize population P(t);
Repeat
Evaluate individuals from P(t);
Selection on P(t). Let P'(t) be the selected individuals;
Crossover on P'(t). Survival between parents and offspring.
Mutation on P'(t). Survival between parent and offspring.
t = t+1;
P(t) = P'(t-1);
Until t = Number of generations.

Remarks:

(i) The selection procedure used is binary tournament.
(ii) Crossover and mutation are performed with a given probability.
(iii) Survival between parents and offspring after crossover will return as results the best two individuals among the four (two parents and two offspring) considered.
(iv) Survival between parents and offspring after mutation is made by direct comparison (in terms of fitness function). The best one between parent and offspring will be accepted.

3.2. Particle Swarm Optimization

Particle swarm optimization is also a population-based method, which can be successfully applied for optimization. The concept of particle swarms, although initially introduced for simulating human social behavior, has become very popular these days as an efficient search and optimization technique. PSO, as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple.

In PSO, a population of conceptual "particles" is initialized with random positions x_i and velocities v_i, and a function, f, is evaluated, using the particle's positional coordinates as input values. In an n-dimensional search space, $x = (x_1, x_2, x_3, \ldots, x_n)$ and $v = (v_1, v_2, v_3, \ldots, v_n)$. Positions and velocities are adjusted, and the function is evaluated with the new coordinates at each timestep. The best position of the particle found during the search process—pbest—as well as the position of the best particle from the entire swarm—gbest (global best)—is stored. The basic update equations for the kth dimension of a current particle $p\_current$ in PSO at iteration t may be given as

$$v_k(t+1) = \omega v_k(t) + c_1 rand_1(pbest_k - p\ current_k(t))$$

$$+ c_2 rand_2(gbest_k - p\ curent_k(t))$$

$$p\ current_k(t+1) = p\ current_k(t) + v_k(t+1)$$

The variables $rand_1$ and $rand_2$ are random positive numbers, drawn from a uniform distribution and defined by an upper limit rand_max, which is a parameter of the system. c_1 and c_2 are called acceleration constants, whereas ω is called inertia weight. $pbest_k$ is the local best solution found so far by the particle, whereas $gbest_k$ represents the positional coordinates of the fittest particle found so far in the entire community (for dimension k). Once the iterations are terminated, most of the particles are expected to converge to a small radius surrounding the global optima of the search space.[33–38,46] The core of the PSO algorithm used in our experiments is presented below:

Particle Swarm Optimization

```
Initialize the population of particles.
For each particle set its pbest.
Set the gbest.
Set t = 1;
Repeat
  For each particle
          update its current position using equations (1);
                  If the value of the function to optimize for the new
                  obtained particle is better than the initial one
                  Then keep the new obtained particle
             Else keep the initial particle
             Update pbest
  Endfor
  Update gbest;
  t = t+1;
Until t = Number of iterations.
```

4. EXPERIMENTAL RESULTS

To demonstrate the performance of the proposed LSRS method, we present the results obtained for a set of benchmark problems that are described in Subsection 4.2. We consider a high number of dimensions varying between 50 and 2000.

4.1. Performance Assessment

According to Baritompa and Hendrix,[39] there are two criteria, which must be taken into account: effectiveness and efficiency. The first one reflects whether we reached what we wished (solution with a given approximation or so) and the second one refers to the (computational) cost required to do this. One measure of effectiveness can be given by the number of times the global optimum has been reached by a certain algorithm. For this purpose, usually several repetitions of the algorithm application are required. For measuring the efficiency, usually the number of function evaluations used is considered. The convergence speed—which is a measure of whether the solution in the next iteration is improved and how—can also be considered as an efficiency criterion. To asses the performances of the new proposed technique, performance graphs are used, and the empirical results are also compared with genetic algorithms and particle swarm optimization algorithms. All the algorithms including LSRS, GA, and PSO have been implemented using C++ Builder 6.0 and were run on a 2.4-GHz Intel Duo Core CPU, with a 2-GB RAM.

4.2. Test Functions

The proposed algorithm is tested by using a set of standard continuous test functions,[4,40] which are widely used in the literature and whose characteristics are

diverse enough to cover many of the problems, which can arise in global optimization problems as mentioned in Ref. 41.

Ackley function

$$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (0,0,\ldots 0)$, $f(x^*) = 0$.

Levy function

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))$$

$$+(y_n - 1)^2(1 + \sin^2(2\pi x_n))$$

$$y_i = 1 + \frac{x_i - 1}{4}, \text{ for } i = 1, 2, \ldots, n$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (1,1,\ldots 1)$, $f(x^*) = 0$.

Quadric function

$$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (1,1,\ldots 1)$, $f(x^*) = 0$.

Rastrigin function

$$f(x) = 10\,n + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Domain of definition: $[-5.12, 5.12]^n$

Optimum point: $x^* = (0,0,\ldots 0)$, $f(x^*) = 0$.

Rosenbrock function

$$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$$

Domain of definition: $[-5, 10]^n$

Optimum point: $x^* = (1,1,\ldots 1)$, $f(x^*) = 0$.

Schwefel function

$$f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (0, 0,\ldots 0)$, $f(x^*) = 0$.

Sphere function

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (0, 0,\ldots 0)$, $f(x^*) = 0$.

Sum Squares function

$$f(x) = \sum_{i=1}^{n} i x_i^2$$

Domain of definition: $[-10, 10]^n$

Optimum point: $x^* = (0, 0,\ldots 0)$, $f(x^*) = 0$.

4.3. The High-Dimensional Experiments

As reported in the scientific literature, it is evident that PSO and GA are good candidates for a low number of dimensions. All the algorithms were also tested for higher number of dimensions. We considered 50, 100, 250, 500, 750, and 1000 dimensions for all the test functions. Also, we test the performance of LSRS for 2000 dimensions, but comparisons with PSO and GA are not reported because of the poor results, which are evident from the results obtained for 1000 dimensions. Table I presents the values of the main parameters used by the three techniques involved in experiments.

4.3.1. Results and Comparisons

Results for 50, 100, 500, and 1000 dimensions obtained by all the three techniques at the end of search process are presented in Tables II–V, respectively. The best and average solution and standard deviation are displayed. Results obtained by LSRS for 2000 dimensions are presented in Table VI. The convergence of LSRS is illustrated in Figures 1–5 for 50, 100, 500, 1000, and 2,000 dimensions, respectively. The best objective function value obtained at the end of each restart application is

Table I. Parameters used by LSRS, PSO, and GA for 50, 100, 250, 500, 750, 1000, and 2000 dimensions.

	Parameter values						
	No. of dimensions						
Parameter	50	100	250	500	750	1000	2000
LSRS							
No. of starting arbitrary points	500	500	500	500	500	500	500
No. of restarting (reinitialization)	50	50	100	100	100	100	100
No. of iterations per each restarting phase	10	10	10	10	10	10	10
Genetic algorithm							
Population size	500	500	500	500	500	500	
No. of generations	20,000	20,000	20,000	20,000	20,000	20,000	
Mutation probability	0.9	0.9	0.9	0.9	0.9	0.9	
Crossover probability	0.5	0.5	0.5	0.5	0.5	0.5	
Particle swarm optimization							
No. of particles	500	500	500	500	500	500	
No. of iterations	20,000	20,000	20,000	20,000	20,000	20,000	
c_1, c_2	2	2	2	2	2	2	

depicted. In Figure 6, a graphical comparison of the convergence for all the three techniques during the first 1000 iterations for functions having 1000 dimensions is provided.

Table II. The performance for 50 dimensions.

	Function							
Approach	Ackley	Levy	Quadric	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum Squares
GA								
Best	2.882	0.327	6.105E+3	104.86	55.34	10.99	6.544	425.57
Average	3.38	0.327	6.113E+3	122.4	55.34	13.46	15.384	431.05
Standard deviation	0.477	0	545.08	14.99	0	1.896	8.111	39.454
PSO								
Best	5.045	11.93	3.02E+4	198.86	1.58E+4	29.491	59.91	1002.2
Average	6.055	18.88	7.81E+4	247.45	5.9E+4	46.80	100.16	2116.6
Standard deviation	1.017	8.38	3.43E+4	52.01	5.23E+4	10.163	36.73	786.29
LSRS								
Best	−6.5E−19	2.9E−39	6.27E−19	0.0	2.47E−28	1.86E−11	1.34E−22	9.86E−21
Average	−6.5E−19	2.9E−39	2.33E−18	0.0	1.38E−18	1.91E−11	1.38E−18	1.42E−18
Standard deviation	0	0	8.11E−19	0	1.29E−18	4.15E−12	1.29E−18	1.25E−18
Actual optimum	0	0	0	0	0	0	0	0

Table III. The performance for 100 dimensions.

	Function							
Approach	Ackley	Levy	Quadric	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum Squares
GA								
Best	4.333	0.6509	2.2E+5	333.5	111.82	37.91	56.50	3.12E+3
Average	4.483	0.6509	2.2E+5	348.37	2.74E+4	40.07	64.83	3.13E+3
Standard deviation	0.353	0	1.96E+5	32.76	8.61	3.78	9.79	279.63
PSO								
Best	6.01	45.72	4.43E+5	642.57	1.05E+5	92.07	221.21	6.02E+3
Average	6.78	55.72	7.81E+5	707.86	2.19E+5	104.86	249.20	9.99E+3
Standard deviation	0.83	10.39	3.04E+5	98.20	9.59E+4	15.81	32.86	3.52E+3
LSRS								
Best	−6.5E−19	2.9E−39	9.2E−16	0	5.83E−28	7.81E−19	5.34E−19	4.68E−18
Average	−6.5E−19	2.9E−39	1.15E−15	0	6.94E−16	3.98E−10	6.94E−16	6.98E−16
Standard deviation	0	0	4.38E−16	0	6.63E−16	4.97E−10	6.63E−16	6.58E−16
Actual optimum	0	0	0	0	0	0	0	0

4.3.2. Discussions on the Performance and Results

As evident from the results presented in tables above, LSRS converges very fast and obtains very accurate results. It can be observed that there are not many differences between the parameters value settings for different number of dimensions.

Table IV. The performance for 500 dimensions.

	Function							
Approach	Ackley	Levy	Quadric	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum Squares
GA								
Best	7.18	3.24	1.25E+8	4.58E+3	563.638	598.12	1.19E+3	4.07E+5
Average	7.21	3.33	1.25E+8	4.59E+3	566.62	600.51	1.20E+3	4.07E+5
Standard deviation	0.01	0.244	1.12E+7	409.93	63.22	53.21	107.67	3.63E+4
PSO								
Best	8.12	544.57	1.41E+8	6.11E+3	2.43E+6	884.67	211.75	4.86E+5
Average	8.14	546.02	1.47E+8	6.13E+3	2.56E+6	891.19	2180.35	5.22E+5
Standard deviation	0.37	42.36	5.61E+7	11.20	2.17E+5	6.55	31.18	3.55E+4
LSRS								
Best	−4.3E−19	2.9E−39	2.12E−11	0	3.4E−27	2.91E−19	4.54E−16	4.05E−35
Average	−4.3E−19	2.9E−39	4.31E−11	0	2.61E−11	4.08E−19	9.0E−16	7.96E−35
Standard deviation	0	0	1.14E−11	0	2.32E−11	3.62E−20	1.52E−16	1.95E−35
Actual optimum	0	0	0	0	0	0	0	0

Table V. The performance for 1000 dimensions.

	Function							
Approach	Ackley	Levy	Quadric	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum Squares
GA								
Best	7.86	6.47	6.28E+8	1.073E+4	1.12E+3	1.49E+3	3.44E+3	2.16E+6
Average	7.87	7.17	6.28E+8	1.075E+4	1.12E+3	1.492E+3	3.45E+3	2.16E+6
Standard deviation	0.0056	1.07	66.81	482.31	10.8	67.0	155.21	9.72E+4
PSO								
Best	8.91	1.61E+3	1.5E+9	1.40E+4	6.46E+6	3.9E+19	5.24E+3	2.65E+6
Average	9.02	1.62E+3	1.61E+6	1.40E+4	6.58E+6	4.1E+47	5.50E+3	2.66E+6
Standard deviation	0.10	10.97	9.46E+7	20.2	1.86E+5	9.2E+48	2.7E+2	8.94E+3
LSRS								
Best	1.3E−18	2.9E−39	5.34E−30	0	6.84E−27	9.30E−18	7.97E−19	3.78E−33
Average	1.3E−18	2.9E−39	1.38E−29	0	7.41E−27	1.12E−17	1.25E−18	7.35E−33
Standard deviation	4.8E−33	0	3.68E−30	0	1.66E−28	7.33E−19	2.05E−19	1.49E−33
Actual optimum	0	0	0	0	0	0	0	0

Even for 2000 dimensions, the same number of restarting and the same number of iterations for each restart were used. Empirical and graphical comparisons with GA and PSO also clearly indicate the big difference between these approaches in terms of quality of solutions and speed of convergence.

For the Rastrigin test function, for example, LSRS obtained the clear 0 for minimum and all the starting points converged to this value, even for 2000 dimensions.

It is obvious that the greater the number of restarts, the faster the convergence and the more accurate the results. A greater number of restarts will increase the computational cost (derivatives to be involved and boundaries are to be modified accordingly). But still the computational cost is very less compared with other two techniques (GA and PSO). It takes about 1 min for LSRS to converge for functions having 2000 dimensions while for GA and PSO, it takes few hours even for 1000

Table VI. The performance for 2000 dimensions.

	Function							
Approach	Ackley	Levy	Quadric	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum Squares
LSRS								
Best	−4.3E−19	2.9E−39	9.37E−8	0	1.40E−26	2.41E−17	9.97E−34	7.58E−31
Average	−4.3E−19	2.9E−39	1.69E−7	0	1.48E−26	3.08E−17	2.35E−33	1.27E−30
Standard deviation	9.6E−35	0	3.42E−8	0	2.44E−28	2.31E−18	6.91E−34	2.02E−31
Actual optimum	0	0	0	0	0	0	0	0

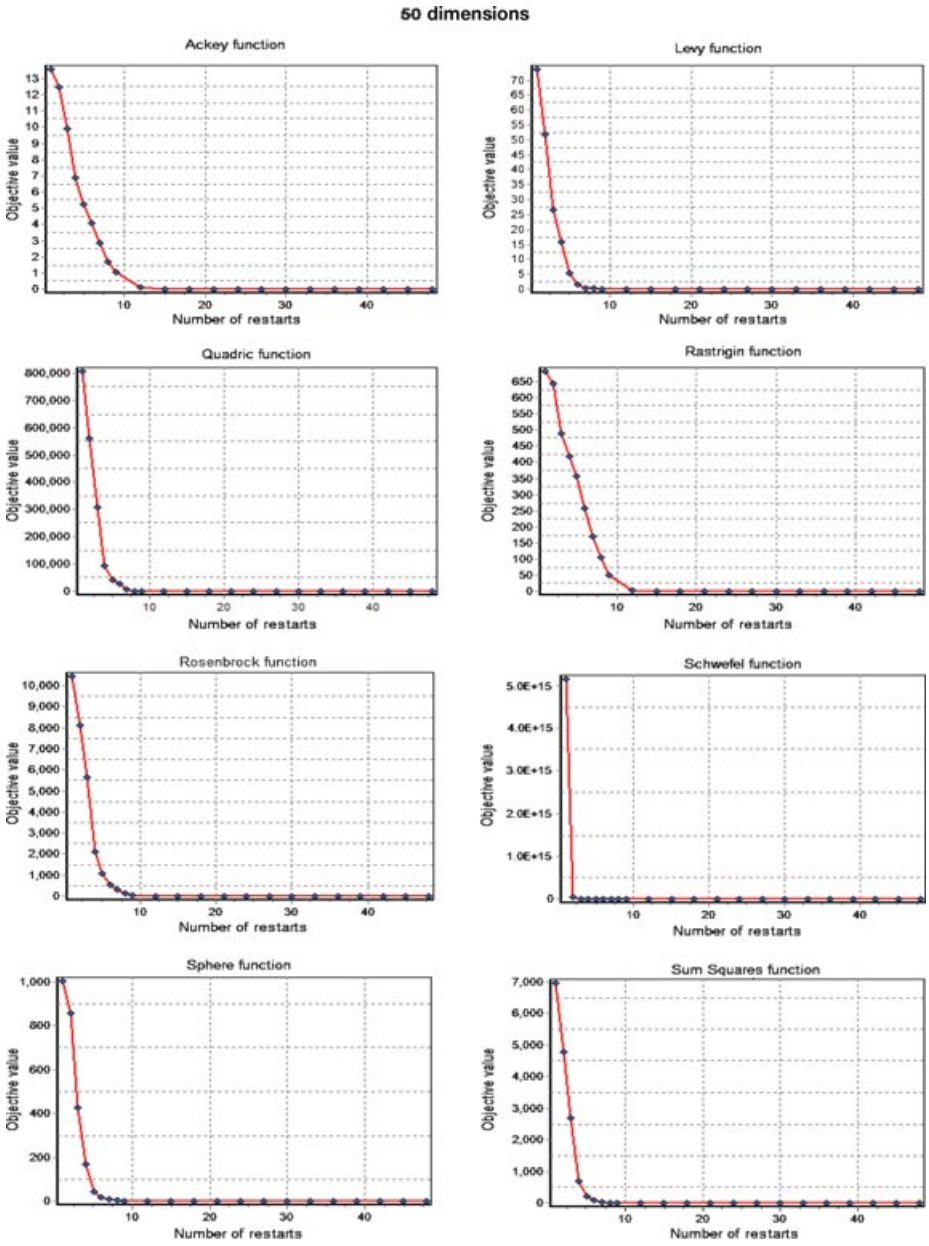

Figure 1. LSRS convergence for 50 dimensions.

dimensions (due to the great number of iterations that these techniques should use to improve the performance). In terms of number of function evaluations, LSRS is more computationally economical than PSO and GA (we refer here to objective function as well as derivatives).

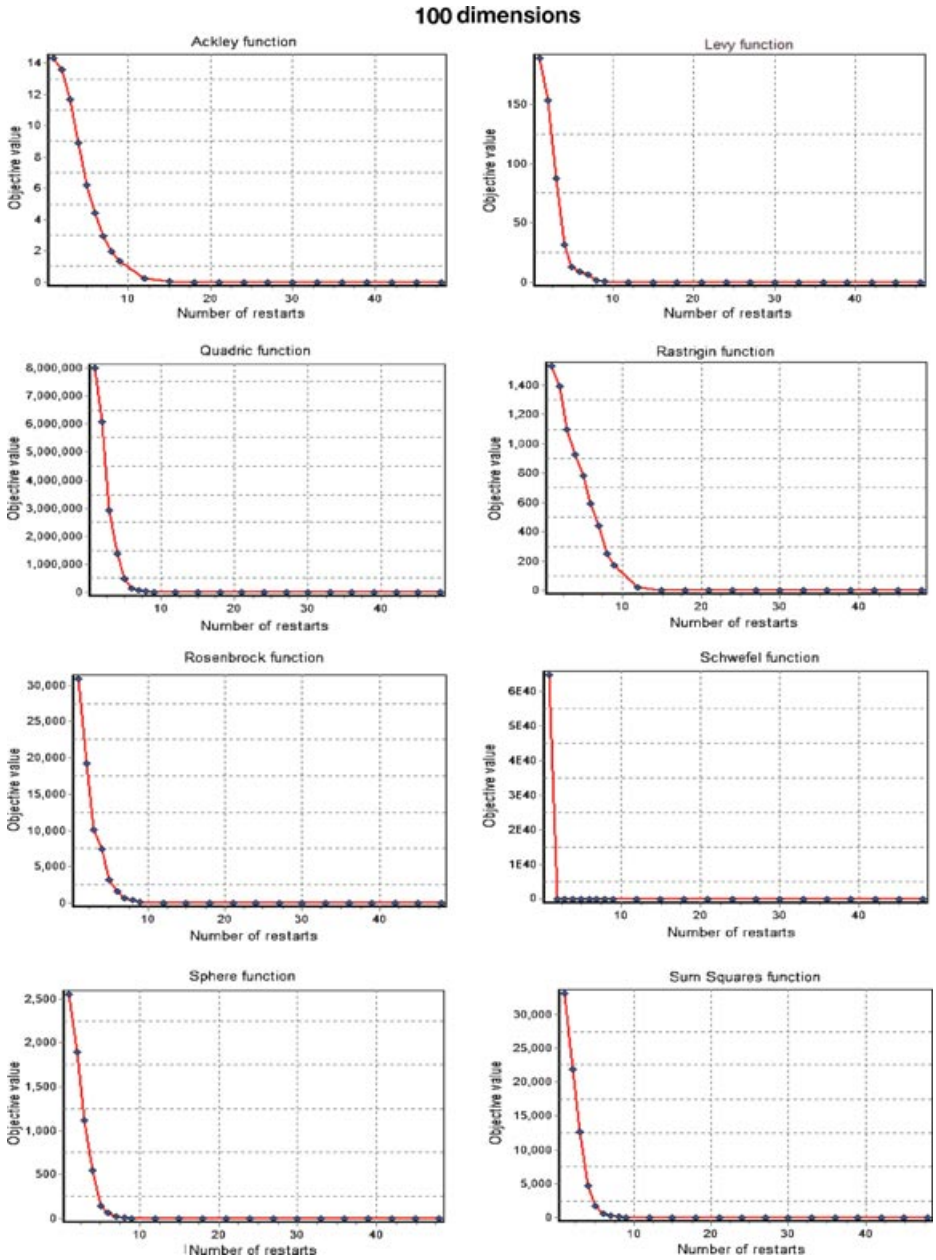

Figure 2. LSRS convergence for 100 dimensions.

5. CONCLUSIONS

A new multidimensional method for solving unconstraint global optimization problems is proposed in this paper. We introduced a modified line search technique

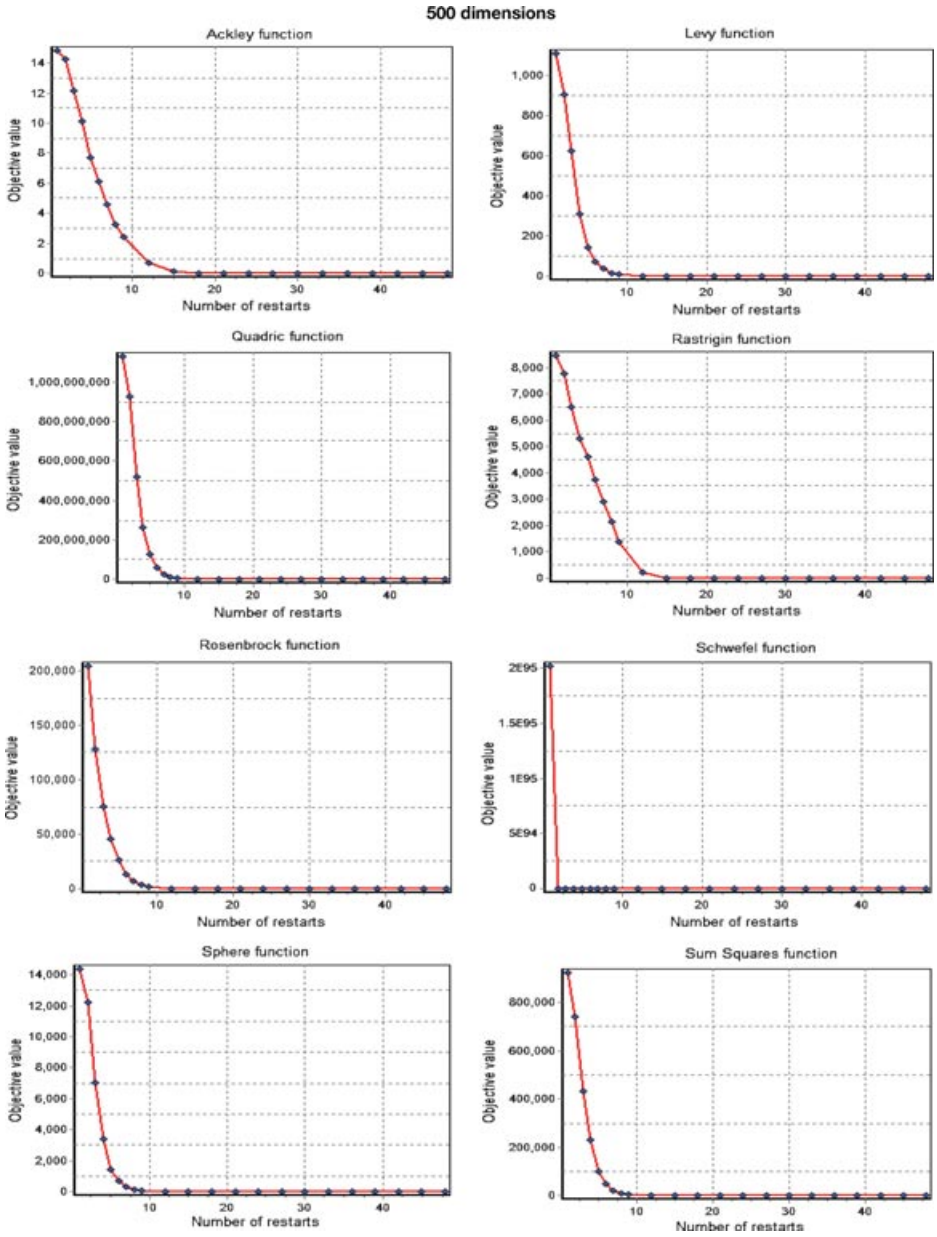

Figure 3. LSRS convergence for 500 dimensions.

called LSRS. The modified classical mathematical technique, which incorporates the multistart method and using the restarting techniques, is found to be computationally efficient for large dimensional functions optimization problems. The computational comparisons with two well-known global optimization techniques—GA and PSO—

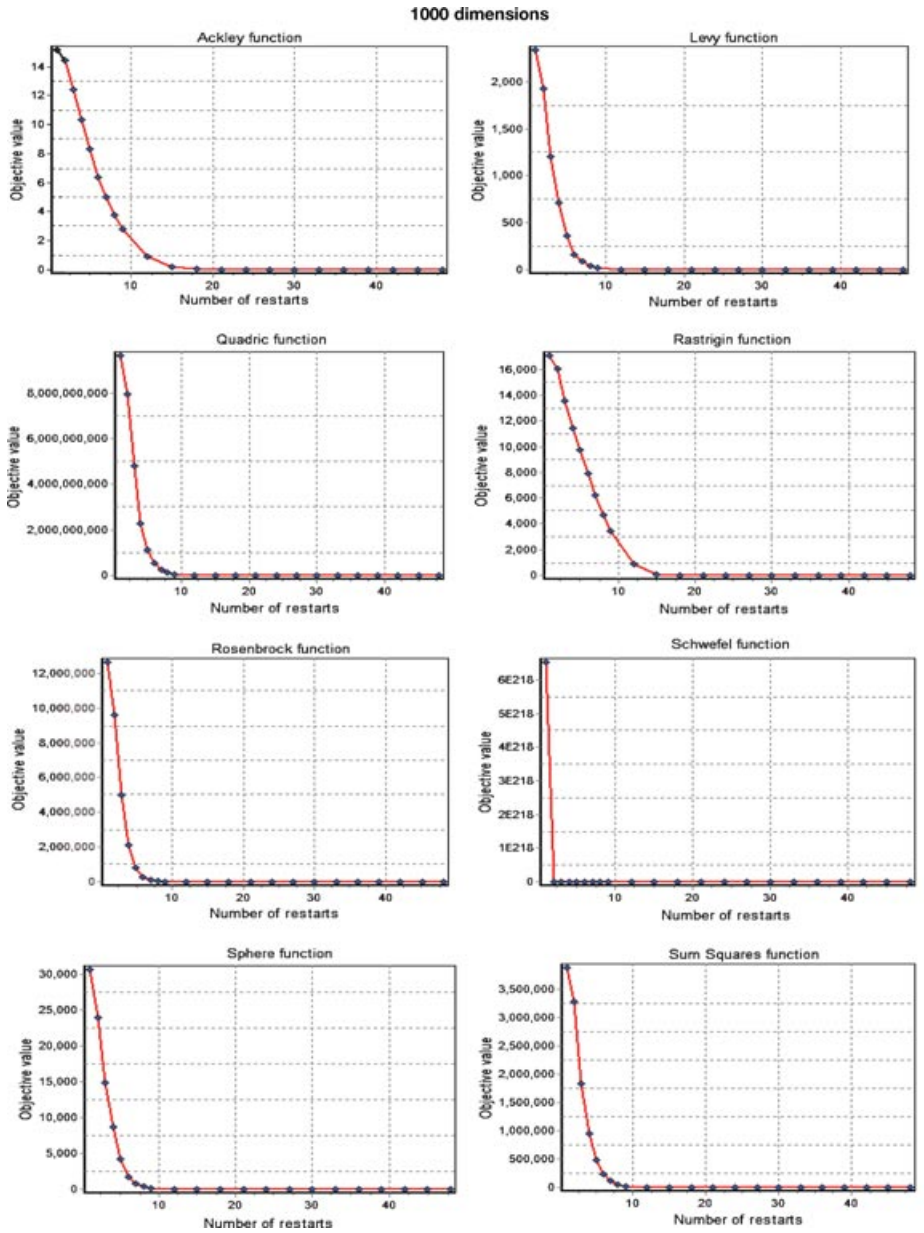

Figure 4. LSRS convergence for 1000 dimensions.

clearly illustrate the superiority of the proposed approach and its independence for the number of dimensions involved.

The LSRS technique makes use of gradient information, which makes it to be restricted to a special class of functions (continuous and differentiable). But it appears to be very efficient for multidimensional optimization problems. We

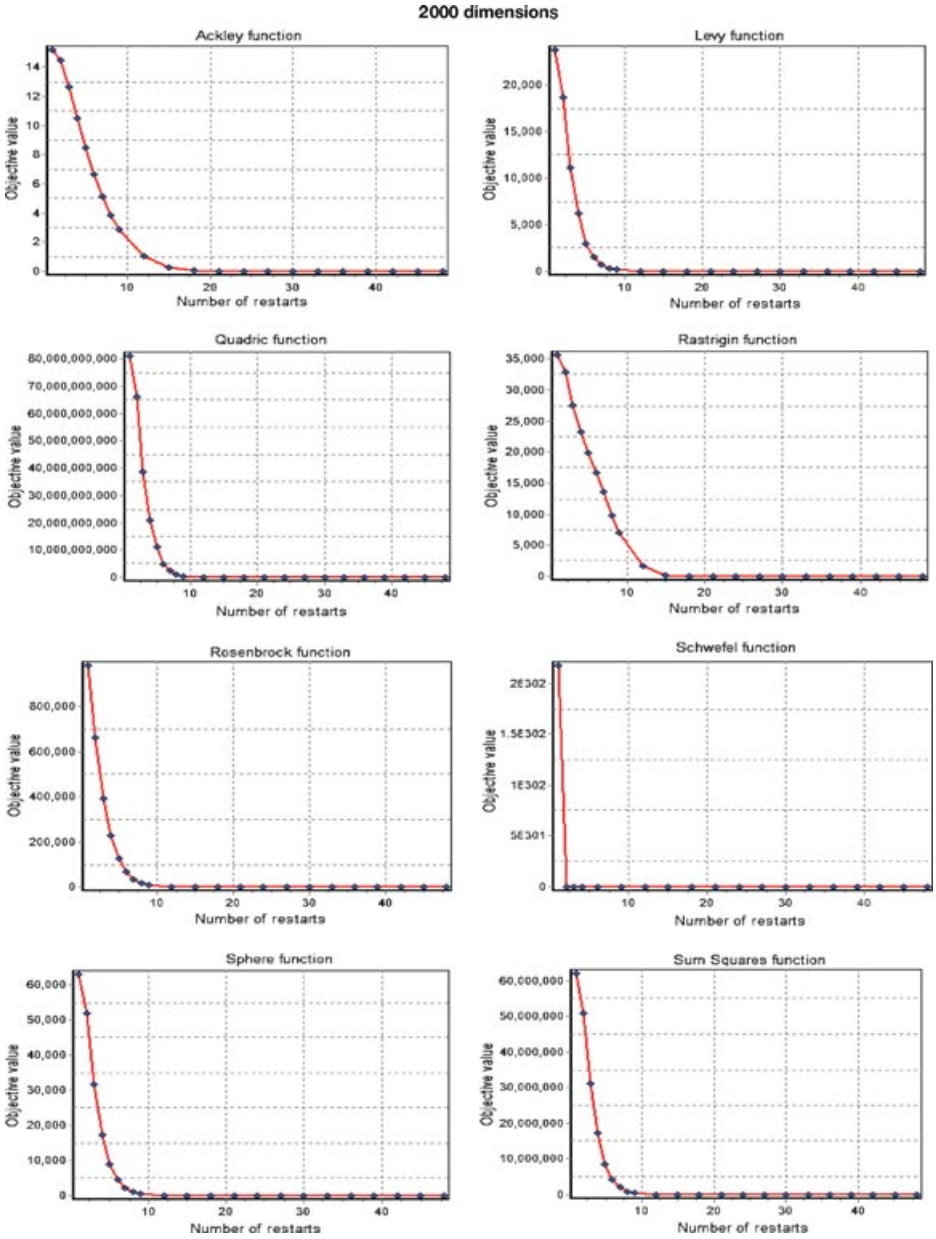

Figure 5. LSRS convergence for 2000 dimensions.

presented the empirical results and graphical illustrations for functions having up to 2000 variables. The proposed technique can be used without any modifications for a higher number of variables. We also intent to develop and adapt the LSRS technique so that it can be further applied for constraint optimization problems, which are also of great interest in some practical optimization problems.

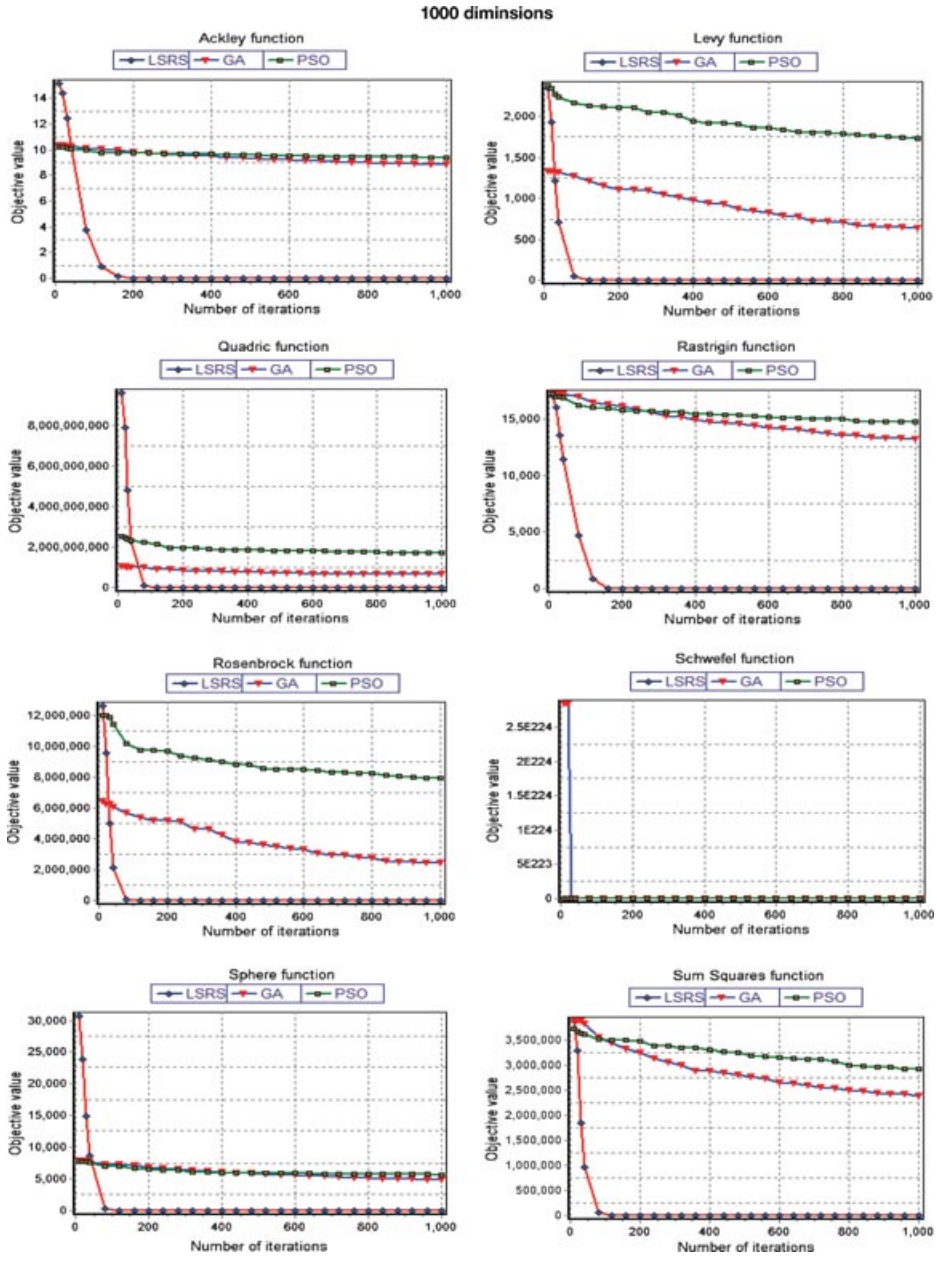

Figure 6. Comparison of LSRS, GA, and PSO for 1000 dimensions for the first 1000 iterations.

Acknowledgment

The first author acknowledges the support from the grant: Scientific computing and optimization for interdisciplinary applications, IDEI-2412, CNCSIS, Romania.

References

1. Gergel V. A global optimization algorithm for multivariate functions with Lipschizian first derivatives. J Global Optim 1997;10:257–281.
2. Bomze IM, Csendes T, Horst R, Pardalos PM, editors. Developments in global optimization. Dordrecht, The Netherlands: Kluwer Academic; 1996.
3. Dixon LCW, Szegö GP, editors. Towards global optimization 2. Amsterdam: North-Holland; 1978.
4. Floudas CA, Pardalos PM. A collection of test problems for constrained global optimization algorithms. In: Goods G, Hartmanis J, editors, Lecture notes in computer science, vol 455. Berlin: Springer-Verlag; 1990.
5. Horst R, Tuy H. Global optimization—Deterministic approaches. Berlin: Springer; 1996.
6. Horst R, Pardalos PM, editors. Handbook of global optimization. Dordrecht, The Netherlands: Kluwer Academic; 1995.
7. Pintér JD. Global optimization in action. Dordrecht, The Netherlands: Kluwer Academic; 1996.
8. Törn AA, Zilinskas A. Global optimization. Lecture notes in computer science, vol 350. Berlin: Springer-Verlag; 1989.
9. Byrd RH, Dert CL, Rinnooy Kan AHG, Schnabel RB. Concurrent stochastic methods for global optimization. Math Programming 1990;45(1–3):1–29.
10. Migdalas A, Pardalos PM, Storoy S, editors. Parallel computing in optimization. Norwell, MA: Kluwer Academic; 1997.
11. Hofinger S, Schindler T, Aszodi A. Parallel Global Optimization of High-Dimensional Problems, Euro PVM/MPI, Kranzlmuller D et al. (Eds.): LNCS 2474, 2002. pp. 148–155.
12. Schutte JF, Reinbolt JA, Fregly BJ, Haftka RT, George AD. Parallel global optimization with the particle swarm algorithm. Int J Numer Methods Eng 2004;61:2296–2315.
13. Abraham A, Grosan C, Ramos V, editors. Stigmergic optimization, studies in computational intelligence. Berlin: Springer-Verlag; 2006. p 300.
14. Dumitrescu D, Grosan C, Oltean M. A new evolutionary adaptive representation paradigm, Studia Universitas Babes-Bolyai, Seria Informatica, 2001;Volume XLVI, No. 2, pp. 19–28.
15. Emmerich MTM, Giannakoglou KC, Naujoks B. Single and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans Evol Comput 2006;10(4): 421–439.
16. Grosan C, Oltean M. Adaptive representation for single objective optimizationsoft comput 2005;9(8):594–605.
17. Grosan C, Abraham A. Hybrid line search for multiobjective optimization. In: Int Conf High Performance Computing and Communications (HPCC-07), Houston, TX, 2007. LNCS, Vol. 4782. Berlin: Springer; 2007. pp 62–73.
18. Grosan C, Abraham A. Modified Line Search Method for Optimization. In: First IEEE Asia Int Conf Modeling and Simulation, AMS-07, Thailand, IEEE Computer Society Press, 2007, pp 415–420.
19. Hirsch MJ, Meneses CN, Pardalos PM, Resende MGC. Global optimization by continuous grasp. Optim Lett 2007;1:201–212.
20. Ismael A, Vaz F, Vicente LN. A particle swarm pattern search method for bound constrained global optimization. J Global Optim 2007.
21. Koumousis VK, Katsaras CP. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 2006;10(1): 19–28.
22. Krishnakumar K. Micro-genetic algorithms for stationary and nonstationary function optimization. In: Proc SPIE Intelligent Control Adaptive Systems, 1989, pp 289–296.
23. Liang JJ, Qin AK, Suganthan PN,Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 2006;10(3): 281–295.
24. Liu H, Abraham A, Zhang W. A fuzzy adaptive turbulent particle swarm optimization. Int J Innovative Comput Appl 2007;1(1):39–47.

25. Maaranen H, Miettinen K, Penttinen A. On initial populations of a genetic algorithm for continuous optimization problems. J Global Optim 2007;37:405–436.
26. Parsopoulos KE, Vrahitis MN. Recent approaches to global optimization problems through. Part Swarm Optim Nat Comput 2002;1:235–306.
27. Stepanenco S, Engels B. Gradient tabu search. J Comput Chem 2007;28(2):601–611.
28. Trafalis TB, Kasap S. A novel metaheuristic approach for continuous global optimization. J Global Optim 2002;23:171–190.
29. Holland JH. Adaptation in natural and artificial system. Ann Arbor, MI: University of Michigan Press; 1975.
30. Bäck T, Fogel D, Michalewicz Z. Handbook of evolutionary computation. New York: IOP Publishing and Oxford University Press; 1997.
31. Bäck T, Fogel D, Michalewicz Z. Evolutionary computation 1: Basic algorithms and operators. Bristol, UK: IOP Publishing; 2000.
32. Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison Wesley; 1989.
33. Hu X, Shi Y, Eberhart RC. Recent advences in particle swarm. In: Proc Congress on Evolutionary Computation (CEC). Portland, Oregon, 2004, pp 90–97.
34. Kennedy J. The particle swarm: social adaptation of knowledge. In: Proc IEEE Int Conf on Evolutionary Computation. Indianapolis, Indiana, IEEE Service Center, Piscataway, NJ, 1997, pp 303–308.
35. Kennedy J. Minds and cultures: particle swarm implications. Socially Intelligent Agents, Papers from the 1997 AAAI Fall Symposium. Technical Report FS-97-02, AAAI Press, Menlo Park, CA, 1997, pp 67–72.
36. Kennedy J. The behavior of particles. In: Proc Seventh Annual Conf on Evolutionary Programming. San Diego, USA, 1998.
37. Kennedy J. Thinking is social: Experiments with the adaptive culture model. J Conflt Resol 1998;42:56–76.
38. Kennedy J, Eberhart R, Shi Y. Swarm intelligence. Morgan Kaufmann Academic Press, 2001.
39. Baritompa B, Hendrix EMT. On the investigation of stochastic global optimization algorithms. J Global Optim 2005;31(4):567–578.
40. Hedar A, Fukushima M. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optim Methods Softw 2004;19:291–308.
41. Hedar AR, Fukushima M. Tabu search directed by direct search methods for nonlinear global optimization. Eur J Operations Res 2006;170:329–349.
42. Addis A, Leyffer S. A trust-region algorithm for global optimization. Comput Optim Appl 2006;35:287–304.
43. Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proc Sixth Int Symposium on Micromachine and Human Science. Nagoya, Japan, 1995, pp 39–43.
44. Floudas CA, Pardalos PM, editors. Encyclopaedia of optimization. Norwell, MA: Kluwer Academic Publishers; 2001.
45. Floudas CA, Pardalos PM, editors. Frontiers in global optimization. Norwell, MA: Kluwer Academic Publishers; 2003.
46. Kennedy J, Eberhart R. Particle swarm optimization. In: Proc IEEE Int Conf on Neural Networks. 1995, pp 1942–1948.
47. Macready WG, Wolpert DH. The no free lunch theorems. IEEE Trans Evol Comput 1997;1(1):67–82.
48. Moré JJ, Garbow BS, Hillstrom KE. Testing unconstrained optimization software. ACM Transac Mathe Softw 1981;7(1):17–41.
49. Pardalos PM, Rosen JB, editors. Computational methods in global optimization. Ann Operations Res 1990;25.