Applied Mathematics and Computation 216 (2010) 2988–3001

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Enhancing PSO methods for global optimization

Ioannis G. Tsoulos [a,*], Athanassios Stavrakoudis [b]

[a] Technological Education Institute of Epiros, Department of Communications, Informatics and Management, University of Ioannina, Greece
[b] Department of Economics, University of Ioannina, Ioannina, Greece

ARTICLE INFO

Keywords:
Global optimization
Particle swarm optimization
Stochastic methods
Stopping rules

ABSTRACT

The Particle Swarm Optimization (PSO) method is a well-established technique for global optimization. During the past years several variations of the original PSO have been proposed in the relevant literature. Because of the increasing necessity in global optimization methods in almost all fields of science there is a great demand for efficient and fast implementations of relative algorithms. In this work we propose three modifications of the original PSO method in order to increase the speed and its efficiency that can be applied independently in almost every PSO variant. These modifications are: (a) a new stopping rule, (b) a similarity check and (c) a conditional application of some local search method. The proposed were tested using three popular PSO variants and a variety test functions. We have found that the application of these modifications resulted in significant gain in speed and efficiency.

1. Introduction

The problem of locating the global minimum of a continuous and differentiable function f can be formulated as: Determine

$$x^* = \arg\min_{x \in S}(x), \tag{1}$$

where the hyper box $S \subset R^n$ is defined as:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \cdots [a_n, b_n].$$

The above problem is applicable in many scientific fields such as chemistry [1,5,6], physics [2,4], architectural synthesis [3], economics [7,8] etc. During the past years many methods have been proposed for tackling the problem of global optimization. These methods can be divided in two main categories namely deterministic and stochastic. Methods belonging to the first category are more difficult to implement and they depend on a priory information about the objective function. Therefore, they are not further examined in this paper. On the other side, stochastic methods are implemented more easily and they do not require a priory information about the objective function. Among the stochastic methods for global optimization we refer to Random Line Search [9], Adaptive Random Search [10], Competitive Evolution [11], Controlled Random Search [12], Simulated Annealing [13–16], Genetic Algorithms [17,18], Differential Evolution [19,20], Tabu Search [21] etc. A stochastic method for global optimization which has recently attracted considerable attention by researchers is the Particle Swarm Optimization (PSO) algorithm. The PSO was initially suggested by Kennedy and Eberhart [22].

PSO is an evolutionary algorithm and it is based on population of candidate solutions (swarm of particles) which move in an n-dimensional search space. Every particle i is assigned its current position x_i and the so called velocity u_i. The two vectors are repeatedly updated, until a predefined convergence criterion is met (see Algorithm 1). The PSO method has been applied

* Corresponding author.
E-mail addresses: itsoulos@cs.uoi.gr, itsoulos@gmail.com (I.G. Tsoulos).

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.04.011

on a wide range of applications [23–26]. Recently many variations of the original method have been proposed in order to increase the speed and the efficiency of the method. Some of these methods aim to develop an automatic mechanism for the estimation of the parameters of the PSO method [27–29], while some others conjunct the PSO method with different type of stochastic techniques [30–32].

This article proposes three modifications of the PSO method: (a) a new stopping rule, (b) a similarity check and (c) a periodically application of a local optimization procedure. The generality of these modifications allows their application to any PSO variant.

The rest of this article is organized as follows: in Section 2 a general description of a typical PSO method is given as well as a detailed description of the proposed modifications. In Section 3 we describe the optimization problems used in our comparison and we give the results from the application of the proposed modifications to three PSO variants. Finally, in Section 4 a discussion is made about the experimental results.

2. Method description

In this section an outline of the general PSO method is given followed by the description of the three proposed modifications.

2.1. Description of the PSO method

The main steps of a generic PSO algorithm are presented in Algorithm 1. In **Initialization** step the algorithm: (a) sets the number of particles, (b) initializes the iteration counter and (c) assigns the initial positions (x_i) of the particles and their velocities (u_i) with uniformly distributed random numbers. The vector p_i holds the best visited position (the one with the lowest function value) for the particle i and the vector p_{best} is the best among $\{p_1, p_2, \ldots, p_m\}$. In the **Termination Check** step, the algorithm checks some predefined criteria such as the maximum number of iterations ($k \geqslant k_{\max}$) or how close is the best and the worst function values of the particles ($|f_{\max} - f_{\min}| < e, e > 0$) or some other stopping rules. In step 2 the main loop of the algorithm is performed: for every particle (a) the velocity is updated, (b) the current position is modified as a function of its associated velocity, (c) the fitness is calculated and (d) the best position p_i is updated if a better position is found. After the main loop the best position among p_i is assigned to p_{best}. The most common update mechanism for the current position of a particle is given by:

$$x_i = x_i + u_i, \tag{2}$$

where

$$u_{ij} = \omega u_{ij} + r_1 c_1 (p_{ij} - x_{ij}) + r_2 c_2 (p_{\text{best},j} - x_{ij}). \tag{3}$$

The j parameter denotes the jth element of the vector, where $j \in [1, \ldots, n]$. The parameters r_1 and $r_2$are random numbers in $[0, 1]$ and the constants c_1 and c_2 stands for the cognitive and the social parameters. Usually, the values for $c_1$and c_2 are in $[1, 2]$. The parameter ω is called inertia with $0 \leqslant \omega \leqslant 1$. In [36] an update mechanism is proposed for the inertia following the formula:

$$\omega = \omega_{\max} - \frac{k}{k_{\max}} (\omega_{\max} - \omega_{\min}), \tag{4}$$

where $k_{\max}$ is the maximum number of iterations allowed and $\omega_{\min}$, $\omega_{\max}$ are defined by the user. Common values for these parameters are $\omega_{\min} = 0.4$ and $\omega_{\max} = 0.9$.

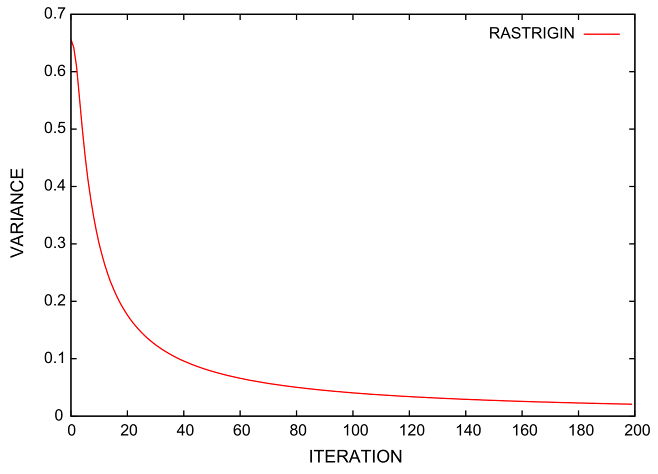

Fig. 1. Plot of variance of best value for the function Rastrigin.

Algorithm 1. The generic PSO algorithm

```
1. Initialization.
   (a) Set $k = 0$ (iteration counter).
   (b) Specify the parameter $m$ (number of particles)
   (c) Randomly initialize the positions of the $m$ particles $x_1, x_2, \ldots, x_m$, with $x_i \in S \subset R^n$
   (d) Randomly initialize the velocities of the $m$ particles $u_1, u_2, \ldots, u_m$, with $u_i \in S \subset R^n$
   (e) For $i = 1, \ldots, m$ do $p_i = x_i$.
   (f) Set $p_{\text{best}} = \arg\min_{i \in 1,\ldots,m} f(x_i)$
2. Termination Check. If the termination criteria hold stop. The final outcome of the algorithm will be $p_{\text{best}}$
3. For $i = 1, \ldots, m$ Do
   (a) Update the velocity $u_i$
   (b) Update the position $x_i$ as a function of $u_i$, $p_i$ and $p_{\text{best}}$
   (c) Evaluate the fitness of the particle $i$, $f(x_i)$
   (d) If $f(x_i) \leqslant f(p_i)$ then $p_i = x_i$
4. End For
5. Set $p_{\text{best}} = \arg\min_{i \in 1,\ldots,m} f(x_i)$
6. Set $k = k + 1$.
7. Goto Step 2
```

2.2. The proposed modifications

2.2.1. Stopping rule

The first, and probably the most essential modification, is a new stopping rule based on asymptotic considerations. Consider the function Rastrigin, which is given by:

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2).$$

The function has 49 local minima in the $[-1,1]^2$ and the global minimum is located at $(0,0)$. Now let a PSO algorithm for the location of the global minimum of the above function with 100 particles. The algorithm terminates when $k \geqslant 200$, where 200 is the maximum number of iterations allowed. In Fig. 1 the variance of the function value of p_{best} is plotted against the number of iterations k. As we can see, the variance decreases smoothly towards to zero. Nevertheless, the algorithm discovers the global minimum at iteration 10 and the rest 190 iterations are spent in order to fulfill the termination criteria. We propose a new stopping rule, which is based on the above variance. The new basic PSO algorithm using the proposed stopping rule is displayed in Algorithm 2. As we see, at every iteration the algorithm calculates the variance (variable v) of the best discovered value (using the auxiliary variables v_1, v_2) and stops when $v \leqslant s$, where s is the variance of the best value, when a new best value was last discovered. The reasoning for this stopping rule is that, when there is no progress for a number of iterations, we could believe that we have found the global minimum and so we must stop iterating.

Algorithm 2. The basic PSO algorithm using the new stopping rule

```
1. Initialization.
   (a) Set $k = 1$ (iteration counter).
   (b) Specify the parameter $m$ (number of particles)
   (c) Randomly initialize the positions of the $m$ particles $x_1, x_2, \ldots, x_m$, with $x_i \in S \subset R^n$
   (d) Randomly initialize the velocities of the $m$ particles $u_1, u_2, \ldots, u_m$, with $u_i \in S \subset R^n$
   (e) For $i = 1, \ldots, m$ do $p_i = x_i$.
   (f) Set $p_{\text{best}} = \arg\min_{i \in 1,\ldots,m} f(x_i)$
   (g) Set $v_1 = 0$, $v_2 = 0$, $p_{\text{old}} = p_{\text{best}}$
2. For $i = 1, \ldots, m$ Do
   (a) Update the velocity $u_i$
   (b) Update the position $x_i$ as a function of $u_i$, $p_i$ and $p_{\text{best}}$
   (c) Evaluate the fitness of the particle $i$, $f(x_i)$
   (d) If $f(x_i) \leqslant f(p_i)$ then $p_i = x_i$
3. End For
4. Termination Check Step
   (a) Set $p_{\text{best}} = \arg\min_{i \in 1,\ldots,m} f(x_i)$
   (b) Set $v_1 = v_1 + |p_{\text{best}}|,\ v_2 = v_2 + p_{\text{best}}^2$
   (c) Set $v = \frac{v_2}{k} - \left(\frac{v_1}{k}\right)^2$
   (d) If $p_{\text{best}} \neq p_{\text{old}}$ Then Set $s = \frac{v}{2},\ p_{\text{old}} = p_{\text{best}}$
   (e) If $v \leqslant s$ Then Stop
5. Set $k = k + 1$.
6. Goto Step 2.
```

Table 1
Variations of the implementation of the proposed PSO modifications. VN is the sequence number of variation, SR is the stopping rule as described at 2.2.1, SC is the similarity check as described at 2.2.2 and LS is the application of local search as described at 2.2.3.

VN	SR	SC	LS	Explanation
1	+			Only the stopping rule modification was applied
2		+		Only the similarity check was applied
3	+	+		Both stopping rule and similarity check were applied
4	+	+	+	All the proposed modifications were applied

2.2.2. Similarity check

The second proposed modification is the so called similarity check. As we can see from the Algorithms 1 and 2 at every iteration the objective function is evaluated at the point x_i even though the point have remained intact during the last iteration. These unnecessary function evaluations can be avoided by simply checking the point x_i for changes, i.e. evaluate $f(x_i)$ if and only if $|x_i^{(k+1)} - x_i^{(k)}| \geqslant e$, where $x_i^{(k)}$ is the point x_i at the k iteration and e a small positive number, i.e. $e = 10^{-5}$.

2.2.3. Application of local search

We propose as a final modification a new operator for the population of particles: at every iteration we decide to apply a local search procedure to the members of population with probability. In order to prevent the algorithm to being trapped in some local minima and to avoid unnecessary function calls we set this probability to $p_l = \frac{1}{m}$, where m is the number of particles. The local search procedure used was a BFGS variant due to Powell [38].

2.3. Implementation

We have applied the previously described modifications of PSO method in four variations, aiming to investigate the particular influence of each one to the resulted speed and efficiency. The four variations are presented in Table 1. In the first variation we have applied only the stopping rule as described in 2.2.1. In the second variation we have applied only the similarity check (2.2.2). In the third variation we have applied a combined application of both stopping rule and similarity check. Finally, in the fourth variation we have applied the local search procedure along with stopping rule and similarity check. We have used the test functions described in Appendix A.

3. Experiments

In this section we list the results from the application of the proposed modifications to four variants of the PSO technique. All PSO methods have been applied to a series of test problems presented in Appendix A.

3.1. PSO variants

We have tested the effect of the proposed modifications to the efficiency and the speed of the method on four PSO variants:

1. **LDWPSO**: proposed by Shi and Eberhart [36], using the velocity update mechanism of Eq. (3).
2. **Center PSO**: proposed by Liu et al. [37], which utilizes the addition of an extra particle called Center Particle. The position X_c of this extra particle is calculated using the formula:

$$X_c = \frac{1}{M}\sum_{i=1}^{M} x_i. \tag{5}$$

The center particle has no velocity, but it is used as an ordinary particle in every computational step of the algorithm. The velocity update of other particles follows the mechanism of **LDWPSO**.

3. **Simple PSO**: This is a simple PSO method, that has no inertia weight in the velocity's update rule. So, the velocity of particle i is updated using the equation:

$$u_{ij} = r_1 c_1 (p_{ij} - x_{ij}) + r_2 c_2 (p_{\text{best},j} - x_{ij}). \tag{6}$$

In contrast to LDWPSO and Center Pso, here the particle has no memory of the previous velocity.

4. **Dynamic PSO** (**DPSO**): This is a modified version of the PSO proposed in [39], that utilizes a different update mechanism for the velocity of the particles.

3.2. Experimental methodology

Four different experimental procedures as given in Table 1 have been applied in order to check the effect of the proposed modifications in the efficiency and the speed of four PSO variants. In all four cases a local search procedure has been applied to the best resulted particle after the satisfaction of the stopping rule. This was performed in order to guarantee that the located point is a true local minimum. All experiments were performed 100 times on every test problem, using different seed for the random number generator. Each PSO variant had a population of 100 particles and the maximum number of generations allowed was set to 200. The same random number set was used among the four PSO variants for each experiment. For the cases of LDWPSO and Center PSO the velocity for each particle was bound to the limits $[-u_{\mathrm{j,max}}, u_{\mathrm{j,max}}]$, with $u_{\mathrm{j,max}}$ defined by:

$$u_{\mathrm{j,max}} = \gamma(a_j - b_j), \quad j = 1, \dots, n. \tag{7}$$

The parameter γ for our experiments was set to 0.5. The parameters c_1, c_2 used in all PSO variants were set to 1. For the case of Dynamic PSO we have used the proposed values for the constant parameters as suggested in the associate manuscript.

3.3. Experimental results

3.3.1. Variation 1

The results from the application of the stopping rule, described at 2.2.1, to the four PSO variants are listed in Table 2. The last row in all tables (denoted by TOTAL) is the total number of function calls for listed test problems. The numbers in parentheses denote the fraction of runs that located the global minimum and were not trapped in one of the local minima. Absence of parentheses denotes that the global minimum has been recovered in every single run (100% success). Application of the

Table 2
Experimental results for the PSO variants using only the proposed stopping rule (variation 1 of Table 1).

PROBLEM	LDWPSO	CENTER PSO	SIMPLE PSO	DPSO
CAMEL	3675	3678	1573	1903
RASTRIGIN	5398(0.96)	5335(0.96)	2211	2464(0.97)
GKLS250	3418	3445	1480	1680
GKLS350	7661(0.98)	7926(0.99)	2479(0.99)	2443(0.98)
AP	4133	3989	1639	1922
BF1	8720(0.76)	8612(0.79)	2478	3147
BF2	9338(0.81)	9184(0.78)	2492(0.97)	3171(0.99)
BRANIN	3366(0.96)	3335(0.96)	1859(0.99)	1803(0.94)
CB3	4572	4415	1664	1925
CM	9918(0.99)	9808(0.99)	3056	2834(0.99)
EASOM	881	885	858	844
EXP8	10,395	9919	3176	4250
EXP32	19,972	20,311	5124	10,280
EXP64	18,852	20,327	5289	9392
GRIEWANK2	7469(0.87)	7949(0.87)	2746(0.88)	3160(0.97)
HANSEN	8885(0.99)	9130(0.99)	4406(0.98)	3819(0.99)
HARTMAN3	4833	4902	2448	2277
HARTMAN6	10,590(0.44)	10,660(0.44)	4430(0.44)	7472(0.44)
ROSENBROCK8	18,311	19,083	7260	16,191
ROSENBROCK32	19,508	20,556	10,520	20,319
ROSENBROCK64	18,366	20,583	11,315	20,471
SHEKEL5	11,669(0.43)	11,682(0.43)	5037(0.43)	3895(0.45)
SHEKEL7	12,845(0.51)	13,337(0.52)	5283(0.48)	4000(0.51)
SHEKEL10	14,685(0.62)	15,186(0.63)	5328(0.49)	3703(0.45)
SHUBERT	5958(0.94)	6055(0.94)	3879(0.98)	3058(0.99)
SINU8	15,878(0.97)	15,065(0.97)	5524(0.94)	7859(0.93)
SINU32	20,127(0.55)	20,324(0.86)	7625(0.95)	13,284(0.90)
SINU64	20,227(0.14)	20,343(0.94)	11,457(0.91)	13,777(0.72)
TEST2N4	22,263	22,423	10,677(0.90)	3563(0.67)
TEST2N5	14,735(0.87)	14,897(0.83)	7756(0.40)	4611(0.42)
TEST2N6	15,509(0.69)	16,151(0.67)	8264(0.33)	5876(0.33)
TEST2N7	16,398(0.53)	17,148(0.54)	8451(0.18)	6977(0.14)
POTENTIAL3	7888	8184	3318	2997
POTENTIAL5	17,027	17,936	19,241	20,166
POWER10	4712	4902	9968	4729
POWER20	13,212	13,183	8151	15,285
TRID50	16,712	20,185	12,797	20,183
TRID100	11,974	20,468	14,534	20,229
TOTAL	**439,765(0.87)**	**461,471(0.90)**	**225,793(0.88)**	**275,959(0.86)**

proposed stopping rule modification resulted in efficiently stopping of the algorithm in all cases. This stopping rule uses information of the objective problem and it is more efficient than using an arbitrary number of maximum steps as a stopping rule. Consequently, it resulted in significant acceleration of the applied method. For example, setting the maximum number of generations to 200 instead of applying the stopping rule needed approximately 20,000 (200 generations $\times$ 100 number of particles) function calls approximate for each optimization problem. As we can see from Table 2, in most of the cases the required function calls were significantly lower than 20000.

The acceleration effect of the stopping rule is similar in LDWPSO, Center PSO and DPSO methods. However, in the case of Simple PSO the application of the stopping rule resulted in dramatically decrease of function calls. For example, in the case of Rastrigin, LDWPSO and Center PSO required more than 5000 function calls, while Simple PSO required 2211 function calls. Moreover, in the case of Griewank2 LDWPSO and Center PSO required more than 7000 function calls, while Center PSO required 2746.

We have described here the effectiveness of the application of a modified stopping rule in four PSO methods. This stopping rule which is described in 2.2.1 is entirely independent of the choice of the PSO variant as it does not use any other information than the best located value of the target function. So it is proposed that it can *in principle* work in any other PSO variant.

3.3.2. Variation 2

The results from the application of the similarity check (variation 2 of Table 1) described at 2.2.2, are listed at Table 3. As in variation 1, the average function calls are significantly lower than 20,000 function calls required by the PSO methods. Decrement of function calls ranged from approximately 40% in the cases of LDWPSO, Center PSO and DPSO to 70% in the case of Simple PSO. The speed gain is relatively higher correspondingly to variation 1. For example, if SINU8 is to be considered, it can be seen that the speed gain from the application of the similarity check ranges from 60% in the cases of LDWPSO and Center PSO to 70% in the case of DPSO and more than 90% in the case of Simple PSO. The corresponding speed gain from

Table 3
Experimental results using the similarity check (variation 2 of Table 1).

PROBLEM	LDWPSO	CENTER PSO	SIMPLE PSO	DPSO
CAMEL	8502	8695	1306	1669
RASTRIGIN	7130	7333	3535	2968
GKLS250	6565	6753	888	1140
GKLS350	7028(0.98)	7205(0.99)	1058(0.99)	1280(0.98)
AP	8267	8461	1137	1566
BF1	9216	9416	1377	1890
BF2	9264	9455	1425	1904(0.99)
BRANIN	8496(0.97)	8698(0.97)	3088	2664(0.97)
CB3	7769	7963	1036	1414
CM	7223	7388	1221	1395(0.99)
EASOM	17,999	18,205	14,478	13,446
EXP8	7913	7978	1455	2278
EXP32	11,926	10,953	1818	7156
EXP64	16,458	15,332	1949	6985
GRIEWANK2	10,145	10,336	3859(0.92)	3379(0.99)
HANSEN	9858	10,076	5796(0.98)	4779
HARTMAN3	6393	6574	1005	1406
HARTMAN6	7253(0.44)	7395(0.44)	1433(0.44)	3788(0.44)
ROSENBROCK8	12,554	12,572	1995	4972
ROSENBROCK32	17,489	16,881	2477	9769
ROSENBROCK64	19,167	18,739	2744	9721
SHEKEL5	7358(0.43)	7853(0.43)	1891(0.43)	1933(0.45)
SHEKEL7	7520(0.51)	8055(0.52)	1909(0.48)	2123(0.51)
SHEKEL10	9233(0.65)	9309(0.65)	2072(0.49)	2010(0.45)
SHUBERT	9958	10,224	5838(0.99)	4510
SINU8	8088	8205	1730(0.94)	2428(0.93)
SINU32	12,739(0.55)	12,483(0.86)	2522(0.95)	8260(0.90)
SINU64	171,017(0.10)	16,998(0.93)	2727(0.91)	8411(0.72)
TEST2N4	8336(0.97)	8542(0.98)	2103(0.78)	2091(0.73)
TEST2N5	8920(0.89)	9005(0.85)	2471(0.48)	2247(0.44)
TEST2N6	9006(0.71)	9198(0.68)	2325(0.33)	2429(0.33)
TEST2N7	9195(0.55)	9349(0.55)	2284(0.18)	2593(0.15)
POTENTIAL3	10,707	10,866	7469	6652
POTENTIAL5	11,260	10,834	8237	8758
POWER10	13,308	13,494	2821	4363
POWER20	16,765	16,806	3205	8563
TRID50	20,044	20,041	3799	12,466
TRID100	20,216	20,359	4306	12,597
TOTAL	**416,375(0.89)**	**418,029(0.92)**	**112,789(0.88)**	**178,003(0.87)**

the application of the previously analyzed variation 1 was ranged from 25% to 75%. However, the better performance of the application of similarity check compared to the application of the stopping rule (variation 1) can not be generalized in all of the cases. A notable exception is for example the Camel function, where the application of variation 1 resulted in less function calls.

Regarding the efficiency, the two variations do not showed notable differentiations. For example, in EXP and Rosenbrock sets of functions both variations resulted in located the global minimum in 100% considering all PSO methods. In cases where global minimum was not founded in all runs (Shekel, Sinu test sets) the two variations of the algorithm resulted in similar success ratios.

The application of the similarity check (see Section 2.2.2) does not depend directly on the PSO variant. Each PSO variant might adopt a different process for the generation of $x_i^{(k)}$, where $x_i^{(k)}$ is the point x_i at the k iteration. However, when the generated points become available, it is not effective to utilize extra function call if the $x_i^{(k+1)}$ points are very similar to $x_i^{(k)}$ points.

3.3.3. Variation 3

The results from the application of both the proposed stopping rule and the similarity check for the four PSO variants are listed in Table 4. As expected function calls are significantly lower than 20,000 calls. Moreover, the simultaneous application of both modifications yielded in increased speed compared to variation 1 or variation 2. Decrement of function calls ranged from approximately 60% in the cases of LDWPSO, Center PSO and DPSO to 90% in the case of Simple PSO. For example, in the Shekel5 test case it took more than 11,000 function calls to locate the global minimum when variation 1 was applied for LDWPSO and Center PSO methods, while similarity check needed approximately 7500 function calls. Likewise, Simple PSO required 5500 function calls in variation 1 and 1900 function calls in variation 2. Interestingly, the combination of the two modifications resulted in even better performance and it took only 6500 function calls for LDWPSO and Center PSO and 1600 function calls for Simple PSO.

Table 4
Experimental results using the proposed stopping rule and the similarity check (variation 3 of Table 1).

PROBLEM	LDWPSO	CENTER PSO	SIMPLE PSO	DPSO
CAMEL	3385	3362	1033	1420
RASTRIGIN	4431(0.96)	4375(0.96)	1544	1773(0.97)
GKLS250	3167	3192	852	1106
GKLS350	5613(0.98)	5821(0.99)	1039(0.99)	1263(0.98)
AP	3968	3891	1105	1512
BF1	6086(0.76)	5987(0.79)	1364	1879
BF2	6571(0.81)	6440(0.78)	1296(0.97)	1887
BRANIN	3203(0.96)	3191(0.96)	1261(0.99)	1506(0.94)
CB3	4182	4053	995	1345
CM	6805(0.99)	6908(0.99)	1208	1383(0.99)
EASOM	875	879	850	842
EXP8	7731	7727	1443	2174
EXP32	11,824	10,953	1811	7043
EXP64	15,305	15,548	1943	7217
GRIEWANK2	6076(0.87)	6535(0.87)	2036(0.88)	2382(0.97)
HANSEN	6674(0.99)	6777(0.99)	2952(0.98)	2794(0.99)
HARTMAN3	4306	4447	988	1383
HARTMAN6	6957(0.44)	7208(0.44)	1343(0.44)	3731(0.44)
ROSENBROCK8	11,429	11,999	1989	4961
ROSENBROCK32	16,809	16,881	2472	9769
ROSENBROCK64	17,230	18,287	2739	9770
SHEKEL5	6505(0.43)	6720(0.43)	1624(0.43)	1752(0.45)
SHEKEL7	7084(0.51)	7550(0.52)	1613(0.48)	1922(0.51)
SHEKEL10	8169(0.62)	8541(0.63)	1791(0.49)	1769(0.45)
SHUBERT	4685(0.94)	4747(0.94)	2599(0.98)	2388(0.99)
SINU8	7804(0.97)	7727(0.97)	1725(0.94)	2410(0.93)
SINU32	12,739(0.55)	12,483(0.86)	2518(0.95)	8234(0.90)
SINU64	17,017(0.14)	16,998(0.94)	2725(0.91)	8050(0.72)
TEST2N4	7273	7541	1984(0.90)	1896(0.67)
TEST2N5	8412(0.87)	8513(0.83)	2214(0.40)	2155(0.42)
TEST2N6	8524(0.69)	8926(0.67)	2260(0.33)	2383(0.33)
TEST2N7	8645(0.53)	9229(0.54)	2222(0.18)	2478(0.14)
POTENTIAL3	6386	6631	2285	2377
POTENTIAL5	9689	9812	7541	8745
POWER10	4644	4782	2821	3151
POWER20	12,071	12,277	3099	8133
TRID50	16,599	19,858	3796	12,392
TRID100	11,953	20,348	4320	12,628
TOTAL	**310,826(0.88)**	**327,144(0.91)**	**79,500(0.87)**	**150,003(0.86)**

In some cases, where one of the variations 1 and 2 did not resulted in speed gain (required function calls remained approximately 20,000) as in SINU32 test function, the combination of stopping rule and similarity check did not perform better than the best of the two methods. As it can be seen from Tables 3 and 4 it took approximately 20,000 for LDWPSO and Center PSO when only stopping rule was applied and approximately 12,500 when similarity check was applied. The combination of stopping rule and similarity check did not improve the performance so as in the similarity check case it took approximately 12,500 function calls to locate the global minimum.

3.3.4. Variation 4

While stopping rule and similarity checked aimed mainly at improving the speed of the algorithm, local search was applied in order to enhance both the speed and the efficiency of the optimization process. As the combination of stopping rule and similarity check gave the best results concerning the speed, we applied the local search method (the third proposed modification) only to this algorithm variation.

Results of this algorithm variation are listed at Table 5. As it can be derived from Tables 2–4 application of local search is needless for some particular cases, where the global minimum is found in all trials (100%). Anyway in the majority of "difficult" cases, such as SINU, HARTMAN6, TEST, SHEKEL etc., local search application greatly improves the efficiency of all PSO methods tested. For example, global minimum of the SHEKEL5 was found approximately in 45% without local search application. On the contrary, as it can be seen from Table 5, the efficiency of the algorithm raised the global finding percentage to 74%. Similarly, global finding was found to be only 14% for the SINU64 test problem with LDWPSO while after the application of the local search the corresponding percentage was found to be 79%.

Interestingly, application of local search yielded in some speed enhancement. As can be derived form table average function calls were approximately 80% (LDWPSO, Center PSO, DPSO) or 90% (Simple PSO) lower than the 20,000 function calls needed without the applied modifications. These values were found to be 40–60% with the application of algorithm variations 1, 2 and 3. One can explain this behavior if we consider the combined application of local search and stopping rule.

Table 5
Experimental results using the three proposed modifications (variation 4 of Table 1).

PROBLEM	LDWPSO	CENTER PSO	SIMPLE PSO	DPSO
CAMEL	1168	1172	954	1073
RASTRIGIN	2148(0.95)	2132(0.95)	1470(0.99)	1678(0.99)
GKLS250	1242	1238	849	1018
GKLS350	2318(0.98)	2286(0.98)	1043(0.99)	1299(0.97)
AP	1542	1522	1169	1370
BF1	5545(0.95)	5611(0.95)	1606	2194
BF2	5563(0.95)	5635(0.96)	1583	2093
BRANIN	942	949	856	902
CB3	4268	4367	984	1259
CM	4514(0.95)	4195(0.95)	1450	1777(0.99)
EASOM	807	813	793	806
EXP8	1160	1158	996	1057
EXP32	2034	1709	1190	1364
EXP64	2111	2028	1243	1413
GRIEWANK2	5535(0.76)	5362(0.76)	2129(0.92)	2607(0.95)
HANSEN	2613(0.98)	2640(0.98)	1886(0.96)	1908(0.98)
HARTMAN3	1055	1062	896	1015
HARTMAN6	1254(0.61)	1259(0.61)	1117(0.61)	1234(0.64)
ROSENBROCK8	9483	9622	1771	2297
ROSENBROCK32	15,022	15,133	2978	3659
ROSENBROCK64	20,180	20,229	4042	4791
SHEKEL5	2886(0.74)	2853(0.74)	1932(0.74)	1648(0.72)
SHEKEL7	3516(0.82)	3318(0.81)	1622(0.72)	1885(0.73)
SHEKEL10	4342(0.93)	4239(0.92)	1726(0.78)	1768(0.79)
SHUBERT	2512(0.95)	2535(0.95)	1794	1805(0.99)
SINU8	1977	1953	1233	1409
SINU32	5681(0.81)	5797(0.85)	2438(0.96)	2554(0.93)
SINU64	8911(0.79)	8536(0.94)	3554(0.88)	3481(0.80)
TEST2N4	1551(0.90)	1577(0.90)	1227(0.75)	1322(0.70)
TEST2N5	1796(0.78)	1810(0.79)	1437(0.61)	1573(0.51)
TEST2N6	2121(0.61)	2100(0.66)	1451(0.42)	1544(0.29)
TEST2N7	2539(0.51)	2511(0.51)	1446(0.12)	1634(0.13)
POTENTIAL3	1231	1238	1247	1250
POTENTIAL5	2041	2047	1943	1997
POWER10	5170	5240	3719	3979
POWER20	12,079	11,873	3619	8449
TRID50	1788	1753	1578	1727
TRID100	2084	2082	2129	2135
TOTAL	**152,729(0.92)**	**151,584(0.93)**	**65,100(0.91)**	**76,974(0.90)**

So, if the local search helps in locating the global minimum, then the algorithm is terminated by the stopping rule, thus there is no need to perform needles function calls inside the main loop of the algorithm just to complete a predefined umber of cycles.

4. Conclusions

In this paper three modifications have been proposed in order to enhance the speed and efficiency of the PSO methodology. The modifications were namely: (a) stopping rule, (b) similarity check and (c) local search. These modifications are general enough and they can be incorporated in almost every PSO variant. A wide area of optimization problems was tested and four different PSO methods were used. The experimental results clearly demonstrated that all three proposed modifications enhance the speed and efficiency of PSO algorithms. Especially, the application of the stopping rule and the similarity check variations, when applied independently, showed significant acceleration of the applied PSO methods. The application of the similarity check showed slightly a better performance than the application of the stopping rule variation. More interestingly, the combined application of the two proposed modifications resulted in even bigger gain in speed. This indicates a synergistic effect on locating the global minimum. Finally, the application of the third proposed modification, namely the local search, resulted in even better speed and significantly raised the efficiency of PSO methods.

The superior performance of the proposed modifications of virtually any PSO methodology implies that they can be used in many future optimization applications where speed and efficiency matters. The proposed modifications are very simple to be implemented in almost every PSO variant.

Appendix A

For the evaluation of the proposed method a series of well - known optimization problems were utilized. These problems can be found in [33] and in [34]. The description of the test problems is given below.

A.1. Ap function

The objective function for the problem of Alluffi–Pentiny is given by

$$f(x) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2,$$

with $x \in [-10, 10]^2$. The value of global minimum is -0.352386.

A.2. Bf1 function

The function Bohachevsky 1 is given by the equation

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10},$$

with $x \in [-100, 100]^2$. The value of global minimum is 0.0.

A.3. Bf2 function

The function Bohachevsky 2 is given by the equation

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10},$$

with $x \in [-50, 50]^2$. The value of the global minimum is 0.0.

A.4. Branin function

$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ with $-5 \leqslant x_1 \leqslant 10, 0 \leqslant x_2 \leqslant 15$. The value of global minimum is 0.397887.

A.4.1. Camel function

The function is given by

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2.$$

The global minimum has the value of $f(x^*) = -1.0316$

A.5. Cb3 function

The Three Hump function is given by the equation

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2,$$

with $x \in [-5, 5]^2$. The value of the global minimum is 0.0.

A.6. Cosine mixture function (CM)

The function is given by the equation

$$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i),$$

with $x \in [-1,1]^n$. The value of the global minimum is 0.4 and in our experiments we have used $n = 4$.

A.6.1. Easom function

The function is given by the equation

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2-\pi)^2 - (x_1-\pi)^2\right),$$

with $x \in [-100, 100]^2$. The value of the global minimum is -1.0

A.6.2. Shubert function

The function is given by $f(x) = -\sum_{i=1}^{2}\sum_{j=1}^{5} j\{\sin((j+1)x_i) + 1\}, x \in [-10,\ 10]^2$. The value of global minimum is -24.06249.

A.6.3. Exponential function

The function is given by

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leqslant x_i \leqslant 1.$$

The global minimum is located at $x^* = (0,0,\ldots,0)$ and -1. In our experiments we used this function with $n = 8,32,64$ and they are denoted by the labels EXP8, EXP32 and EXP64.

A.6.4. Gkls function

$f(x) = \text{Gkls}\ (x,n,w)$, is a function with w local minima, described in [35], $x \in [-1,1]^n$, $n \in [2, 100]$. In our experiments we use $n = 2,3$ and $w = 50$.

A.6.5. Griewank2 function

The function is given by

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100,\ 100]^2.$$

The global minimum is located at the $x^* = (0,0,\ldots,0)$ with value 0.

A.7. Hansen function

$f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j],\ x \in [-10,\ 10]^2$. The global minimum of the function is -176.541793.

A.7.1. Rastrigin function

The function is given by

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2.$$

The global minimum is located at $x^* = (0, 0)$ with value -2.0.

A.7.2. Rosenbrock function

This function is given by

$$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \leqslant x_i \leqslant 30.$$

The global minimum is located at the $x^* = (0,0,\ldots,0)$ with $f(x^*) = 0$. In our experiments we used this function with $n = 8, 32, 64$.

A.7.3. Sinusoidal function

The function is given by

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \leqslant x_i \leqslant \pi.$$

The global minimum is located at $x^* = (2.09435, 2.09435, \ldots, 2.09435)$ with $f(x^*) = -3.5$. In our experiments we used $n = 8, 32, 64$ and $z = \frac{\pi}{6}$ and the corresponding functions are denoted by SINU8, SINU32, and SINU64 respectively.

A.8. Shekel 5

$$f(x) = -\sum_{i=1}^{5}\frac{1}{(x - a_i)(x - a_i)^T + c_i},$$

with $x \in [0, 10]^4$ and

$$a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$$

and

$$c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}.$$

The function has 5 local minima in the specified range.

A.9. Shekel 7

$$f(x) = -\sum_{i=1}^{7}\frac{1}{(x - a_i)(x - a_i)^T + c_i},$$

with $x \in [0, 10]^4$ and

$$a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$$

and

$$c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}.$$

A.10. Shekel 10

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i},$$

with $x \in [0, 10]^4$ and

$$a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$$

and

$$c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}.$$

A.10.1. Test2N function

This function is given by the equation

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5, 5].$$

The function has 2^n in the specified range and in our experiments we used n = 4,5,6,7

A.10.2. Potential function

The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard–Jones potential is determined for the case of N = 3 and N = 5 atoms.

A.11. Hartman 3 function

The function is given by

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right),$$

with $x \in [0, 1]^3$ and

$$a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$$

and

$$c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$$

and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}.$$

The function has 3 minima in the specified range.

A.12. Hartman 6 function

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right),$$

with $x \in [0,1]^6$ and

$$a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix},$$

$$c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix},$$

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}.$$

A.13. Powersum function

$$f(x) = \sum_{k=1}^{n}\left(\sum_{i=1}^{n} x_i^k - b_k\right)^2,$$

where the parameter b is chosen as

$$b_k = \sum_{i=1}^{n} z_i^k$$

and

$$z_i = \frac{1}{i}, \ i = 1, 2, \ldots, n,$$

with $x \in [\min z_i, \max z_i]$ for $i = 1,2,\ldots,n$ The global minimum is $f(x^*) = 0$. In our experiments we have used the values $n = 10,20$ (Examples POWER10, POWER20).

A.14. Trid function

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

with $x_i \in [-n^2, n^2]$, $i = 1,2,\ldots,n$ The global minimum is given by

$$f(x^*) = -\frac{n(n+4)(n-1)}{6}.$$

References

[1] M.M. Ali, C. Storey, A. Törn, Applications of some stochastic global optimization algorithms to practical problems, Journal of Optimization Theory and Applications 95 (1997) 545–563.
[2] M.K. Sen, P.L. Stoffa, Global Optimization Methods in Geophysical Inversion, Elsevier, 1995.
[3] C.H. Gebotys, M.I. Elmasry, Global optimization approach for architectural synthesis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 12 (1993) 1266–1278.
[4] A. Gottvald, K. Preis, C. Magele, O. Biro, A. Savini, Global optimization methods for computational electromagnetics, IEEE Transactions on Magnetics 28 (1992) 1537–1540.
[5] M. Locatelli, F. Schoen, Fast global optimization of difficult Lennard–Jones clusters, Computational Optimization and Applications 21 (2002) 55–70.
[6] C.D. Maranas, C.A. Floudas, Global optimization for molecular conformation problems, Annals of Operations Research 42 (1993) 85–117.
[7] C.D. Maranas, I.P. Androulakis, C.A. Floudas, A.J. Berger, J.M. Mulvey, Solving long-term financial planning problems via global optimization, Journal of Economic Dynamics and Control 21 (1997) 1405–1425.
[8] P.H. Chen, H.C. Chang, Large-scale economic dispatch by genetic algorithm, IEEE Transactions on Power Systems 10 (1995) 1919–1926.
[9] M. Gaviano, Some general results on the convergence of random search algorithms in minimization problems, in: L.C. W Dixon, G.P. Szegö (Eds.), Towards Global Optimization, North-Holland, Amsterdam, 1975, pp. 149–157.
[10] H.A. Bremermann, A method for unconstrained global optimization, Mathematical Biosciences 9 (1970) 1–15. 4, 8.
[11] R.A. Jarvis, Adaptive global search by the process of competitive evolution, IEEE Transactions on System, Man and Cybergenetics 75 (1975) 297–311.
[12] W.L. Price, Global optimization by controlled random search, Computer Journal 20 (1977) 367–370.
[13] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[14] P.J.M. van Laarhoven, E.H.L. Aarts, Simulated Annealing: Theory and Applications, D. Riedel, Boston, 1987.
[15] A. Corana, M. Marchesi, C. Martini, S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, ACM Transactions on Mathematical Software 13 (1987) 262–280.
[16] W.L. Goffe, G.D. Ferrier, J. Rogers, Global Optimization of Statistical Functions with Simulated Annealing, Journal of Econometrics 60 (1994) 65–100.
[17] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
[18] Z. Michaelewizc, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.
[19] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[20] M.M. Ali, L.P. Fatti, A differential free point generation scheme in the differential evolution algorithm, Journal of Global Optimization 35 (2006) 551–572.
[21] D. Cvijoivic, J. Klinowski, Taboo search. an approach to the multiple minima problems, Science 667 (1995) 664–666.
[22] J. Kennedy, R.C. Eberhart, The particle swarm: social adaptation in information processing systems, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, Cambridge, UK, 1999, pp. 11–32.
[23] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltagecontrol considering voltage security assessment, IEEE Transactions on Power Systems 15 (2000) 1232–1239.
[24] J. Robinson, Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, IEEE Transactions on Antennas and Propagation 52 (2004) 397–407.
[25] M.A. Abido, Optimal power flow using particle swarm optimization, International Journal of Electrical Power & Energy Systems 24 (2002) 563–571.
[26] Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems 18 (2003) 1187–1195.
[27] X. Yang, Jinsha Yuan, Jiangy Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation 189 (2007) 1205–1213.
[28] Y. Jiang, T. Hu, C. Huang, X. Wu, An improved particle swarm optimization algorithm, Applied Mathematics and Computation 193 (2007) 231–239.
[29] J. Yisu, J. Knowles, L. Hongmei, L. Yizeng, D.B. Kell, The landscape adaptive particle swarm optimizer, Applied Soft Computing 8 (2008) 295–304.
[30] Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing 8 (2008) 849–857.
[31] B. Liu, L. Wang, Y.H. Jin, F. Tang, D.X. Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons and Fractals 25 (2005) 1261–1271.
[32] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, L.M. Wang, An improved GA and a novel PSO-GA based hybrid algorithm, Information Processing Letters 93 (2005) 255–261.
[33] M. M Ali, C. Khompatraporn, Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, Journal of Global Optimization 31 (2005) 635–672.
[34] C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers., Dordrecht, 1999.
[35] M. Gaviano, D.E. Ksasov, D. Lera, Y.D. Sergeyev, Software for generation of classes of test functions with known local and global minima for global optimization, ACM Transactions on Mathematical Software 29 (2003) 469–480.
[36] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Evolutionary Programming VII, Lecture Notes in Computer Science, 1447, Springer, Berlin, 1998, pp. 591–600.
[37] Y. Liu, Z. Qin, Z. Shi, J. Lu, Center particle swarm optimization, Neurocomputing 70 (2007) 672–679.
[38] M.J.D. Powell, A tolerant algorithm for linearly constrained optimization calculations, Mathematical Programming 45 (1989) 547–566.
[39] B. Jiao, Z. Liana, X. Gu, A dynamic inertia weight particle swarmnext term optimization algorithm, Chaos, Solitons and Fractals 37 (2008) 698–705.