International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol. 7, No. 6 (2014), pp. 183-200
http://dx.doi.org/10.14257/ijsip.2014.7.6.15

A Self-adaptive Global Particle Swarm Optimization Algorithm for Unconstrained Optimization Problems

Dexuan Zou

School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou 221116, China
zoudexuan@163.com

Abstract

This paper aims to present a self-adaptive global particle swarm optimization (SGPSO) algorithm for solving unconstrained optimization problems. In the new algorithm, the inertia weights are generated based on Gaussian distribution, which is helpful to improve the diversity of the population. In addition, the worst particle is updated by averaging the other particles, which is beneficial to improving the quality of the population. Finally, a global disturbance is adopted to increase the convergence rate of SGPSO. In the disturbance process, a disturbance factor is utilized to control the searching ranges of the population, which can effectively keep a balance between the global exploration and local exploitation. Twenty well-known benchmark functions are considered to evaluate the performance of SGPSO, and 50 runs are implemented in each case. Numerical experiments and comparisons demonstrate that SGPSO is superior to the other three algorithms according to means, standard deviations and convergence rate.

Keywords: *self-adaptive global particle swarm optimization, gaussian distribution, global disturbance, convergence rate, global exploration, local exploitation*

1. Introduction

Global optimization problems are frequently arisen in a variety of engineering applications. In some instances, these optimization problems are non-differentiable, which is hard to solve for the gradient-based methods. Thus, researchers have to rely on some stochastic global optimization technologies such as particle swarm optimization algorithm (PSO) [1], differential evolution algorithm (DE) [2], genetic algorithm (GA) [3], harmony search algorithm (HS) [4, 5] etc., because they do not need to compute the gradients of the objective function, and they do not require the continuity of problem variables. In consideration of the preferable advantages of these stochastic global optimization algorithms, researchers have payed closer attention to them recently, and they have implemented these technologies on many complex optimization problems including reliability problems [6, 7], tile manufacturing process [8], adiabatic styrene reactor [9], Off-Centre bracing system [10], T-S fuzzy models [11] and so on.

The particle swarm optimization algorithm [1] is a simple stochastic global optimization technology which generates new candidate solutions by combining particles' personal best positions and the most successful particle's position in each generation. Due to its simple structure and few parameters, the use of the original particle swarm optimization algorithm for optimization problems has draw much attention from researchers in recent years. Shi and Eberhart [12] presented a particle swarm optimization algorithm based on inertia weight, and this improved algorithm is

ISSN: 2005-4254 IJSIP

called standard particle swarm optimization (PSO). To increase the convergence rate of the original particle swarm optimization algorithm, inertia weight is adopted in particles' velocity updating. Moreover, it decreases linearly in the whole iteration process. In the early evolutionary process, it keeps to be a set of large values, which can enhance the global searching capacity of PSO, and in the late evolutionary process, it keeps to be a set of small values, which can enhance the local searching capacity of PSO. Kennedy [13] proposed a novel version of PSO called bare bones particle swarm optimization (BBPSO) algorithm. BBPSO eliminates the velocity updating and position updating of PSO. Instead, each particle updates its position by Gaussian distribution whose mean and standard deviation are associated with its personal best position and the personal best position. This new position updating method has improved the developing ability of PSO. Zhao *et al.* [14] proposed a modified particle swarm optimization via particle visual modeling analysis (PSO-VMA), which describes particle's dimensional vector behavior. More specifically, they analyzed the reason why premature convergence and diversity loss happen in PSO, and introduced a new method to ensure the rational flight of every particle's dimensional component. Additionally, PSO-VMA adopts two parameters to avoid premature convergence, and they are particle-distribution-degree and particle- dimension-distance, respectively. Based on simulation results, PSO-VMA has demonstrated a better capacity of finding the best solution. Ma [15] analyzed the migratory behavior in the nature, and proposed a novel particle swarm optimization algorithm based on particle migration (MPSO). In MPSO, the swarm is randomly divided into several sub-swarms, where inertia weight and acceleration coefficients are dynamically adjusted. In each fixed generations, some particles migrate from one sub-swarm to another, which is helpful to enhance the diversity of the swarm and overcome premature convergence. Liu *et al.* [16] proposed a new improved PSO called center particle swarm optimization (CPSO) algorithm. CPSO generates a position for a center particle by averaging the positions of the other particles. Meanwhile, the updating of the other particles' velocities and positions of CPSO are the same as those of PSO. In short, the position of the center particle is a potential alternative, and it often guides the search direction of the population.

The paper is organized as follows. In Section 2, the PSO algorithm is briefly illustrated, and three improved PSO algorithms are also introduced. In Section 3, a self-adaptive global particle swarm optimization (SGPSO) algorithm is presented, and the SGPSO procedure is fully explained. In Section 4, twenty functions are chosen to evaluate the performance of SGPSO on solving unconstrained optimization problems. We end this paper with some conclusions in Section 5.

2. Four Particle Swarm Optimization Algorithms

The original particle swarm optimization algorithm was proposed by Kennedy and Eberhart [1] in 1995. It has many advantages such as convenience, legibility and applicability etc. So far, its many improved versions have been proposed, and some of them will be presented as follows:

2.1. The Original Particle Swarm Optimization Algorithm

In original particle swarm optimization algorithm, new velocity vectors and position vectors are generated according to the previous velocity vectors and position vectors. Furthermore, its velocity updating and position updating are explained as follows:

$$v_{i,j}^{k+1} = v_{i,j}^{k} + c_1 r_1 (pb_{i,j} - x_{i,j}^{k}) + c_2 r_2 (gb_j - x_{i,j}^{k}) \quad (1)$$

$$x_{i,j}^{k+1} = x_{i,j}^{k} + v_{i,j}^{k+1} \quad (2)$$

For the ith particle, $v_{i,j}^{k}$ and $x_{i,j}^{k}$ denote its jth velocity dimension and jth position dimension at generation k. $v_{i,j}^{k+1}$ and $x_{i,j}^{k+1}$ denote its jth velocity dimension and the jth position dimension at generation $k+1$. $pb_{i,j}$ represents the jth dimension of the personal best particle, and gb_j represents the jth dimension of the global best particle. Additionally, c_1 and c_2 are cognitive parameter and social parameter, respectively. Moreover, r_1 and r_2 are the random numbers generated uniformly in the range of $[0,1]$.

2.2. The Particle Swarm Optimization Algorithm based on Linearly Decreased Inertia Weight

In order to improve the convergence of the original particle swarm optimization, Shi and Eberhart [12] introduced a parameter ω into its velocity updating, and the improved velocity updating is given by:

$$v_{i,j}^{k+1} = \omega v_{i,j}^{k} + c_1 r_1 (pb_{i,j} - x_{i,j}^{k}) + c_2 r_2 (gb_j - x_{i,j}^{k}) \quad (3)$$

Here, ω is defined as inertia weight, and it decreases linearly in particle's evolutionary process. Moreover, the particle swarm optimization algorithm with parameter ω is called standard particle swarm optimization (PSO) algorithm, and its convergence rate is faster than the original particle swarm optimization algorithm in most cases.

2.3. Bare Bones Particle Swarm Optimization (BBPSO)

In 2003, Kennedy proposed a bare bones particle swarm optimization (BBPSO) algorithm [13]. Different from PSO, BBPSO only adopts position updating, moreover, it updates particles' positions by using Gaussian distribution. In detail, the modified position updating is given by:

$$x_{i,j} = Gd(\mu_{i,j}, \sigma_{i,j}^2) \quad (4)$$

Here, Gd represents Gaussian distribution. Its mean is $\mu_{i,j} = (gb_j + pb_{i,j})/2$, and its standard deviation is $\sigma_{i,j} = \left| gb_j - pb_{i,j} \right|$. Due to the utilization of the Gaussian distribution related to the global best solutions and the personal best solutions, BBPSO can find better solutions than the original particle swarm optimization algorithm in most instances.

2.4. Center Particle Swarm Optimization Algorithm

Liu *et al.* [16] proposed an improved version of PSO called center particle swarm optimization (CPSO) algorithm. CPSO generates a position for a center particle by averaging the positions of the other particles. In detail, $N-1$ particles update their positions as the usual PSO algorithms at every iteration, and the generation of center particle is given by:

$$X_{cd}^{k+1} = \frac{1}{M-1}\sum_{i=1}^{M-1} X_{id}^{k+1} \tag{5}$$

The position of center particle is a potential and promising alternative, and it often guides the search direction of the population which is beneficial to producing solutions of high quality.

3. Self-adaptive Global Particle Swarm Optimization Algorithm

In order to further improve the performance of PSO, we modify its control parameter, and change its algorithm structure. The detailed modification steps are presented as follows:

3.1. Adjust Inertia Weight by using a Self-adaptive Strategy

Regarding inertia weight, a thorny issue is the selecting of suitable values, because it is usually a problem-dependent task. In this section, we propose a self-adaptive method to adjust inertia weight, which can effectively avoiding the inconvenience brought by trial-and-error method. Furthermore, each particle has its own inertia weight in each generation, and it is calculated as:

$$\omega_i^{k+1} = \begin{cases} N(\omega_m', \omega_{std}'), & \text{If} \quad r < \gamma \\ N(\omega_m, \omega_{std}), & \text{Otherwise} \end{cases} \tag{6}$$

Here, r is a uniform random value in $[0,1]$. $N(\omega_m', \omega_{std}')$ denotes Gaussian distribution with mean ω_m' and standard deviation ω_{std}'. $N(\omega_m, \omega_{std})$ denotes Gaussian distribution with mean ω_m and standard deviation ω_{std}. In our experiments, we set $\omega_m' = 0.5$, $\omega_{std}' = 0.01$, $\omega_{std} = 0.01$, and $\gamma = 0.1$. With respect to ω_m, it is given by:

$$\omega_m = \frac{\sum_{i \in S} \omega_i^k}{N_S} \tag{7}$$

where S represents the set of all successful inertia weights at generation k. By adopting these successful inertia weights $\omega_i^k \in S$, suitable mean ω_m can be obtained to generate self-adaptive inertia weights $\omega_i^{k+1} (i = 1,2,\cdots,M)$.

3.2. Update the Worst Particle

In order to improve the quality of swarm, we update the worst particle in terms of the other particles, and the updating formula is expressed as follow:

$$x_{i_{worst}}^{k+1} = \frac{1}{M-1}\sum_{i \neq i_{worst}} x_i^k \tag{8}$$

This formula is inspired by center particle swarm optimization (CPSO) algorithm presented by Liu *et al.* [16]. Nevertheless, it is different from CPSO. In detail, the solution that will be replaced has no fixed index, and it depends on the comparison among the objective function values of all solutions. Furthermore, the benign competition is helpful to enhance the capacity of developing solution space for SGPSO. By eliminating poor particles

in evolutionary process, the quality of swarm will be improved step by step. According to Eq. (7), the calculation of mean ω_m may include the i_{worst}th particle's inertia weight $\omega_{i_{worst}}$. However, this inertia weight is non-existent, because Eq. (8) does not contain the term $\omega_{i_{worst}}$. Thus, to finish the generation of ω_m, $\omega_{i_{worst}}$ is set to 0.5 in each iteration.

3.3. Disturb the Global Best Particle

Usually, particles have fast convergence rate at the beginning of evolutionary process. However, when it comes to the end of evolutionary process, particles are easy to be at a standstill, which indicates their poor convergence in the late optimization. To overcome this shortcoming, a disturbing formula is proposed, and it is stated as follow:

$$Gb_j = gb_j \times (1 + \lambda_j \times N(0,1)) \tag{9}$$

Here, $N(0,1)$ denotes Gaussian distribution with mean 0 and standard deviation 1. gb_j denotes the jth dimension of the global best solution. In addition, λ_j is defined as disturbance factor, and it is calculated as:

$$\lambda_j = \begin{cases} \lambda_j^U - (\lambda_j^U - \lambda_j^L) \times 2k/K, & \text{If} \quad k < K/2 \\ \lambda_j^L, & , \quad \text{Otherwise} \end{cases} \tag{10}$$

Here, $\lambda_j^U (j = 1,2,\cdots,N)$ and λ_j^L are respectively the upper bound and the lower bound of λ_j. In our experiments, we set $\lambda_j^U = 10^{-3}(x_j^U - x_j^L)$, and $\lambda_j^L = 10^{-6}(x_j^U - x_j^L)$, where x_j^U and x_j^L represent the upper bound and the lower bound of the jth dimension, respectively. In the early optimization process, SGPSO has strong capacity of global searching due to the utilization of large values of λ_j, and it has strong capacity of local searching in the late optimization process due to the utilization of small values of λ_j. In short, disturbance factor λ_j is dynamically changed with generation number, and it is shown in Figure 1. Finally, it should be noticed that Gb_j will be truncated to $[x_j^L, x_j^U]$ if overflowing happens.

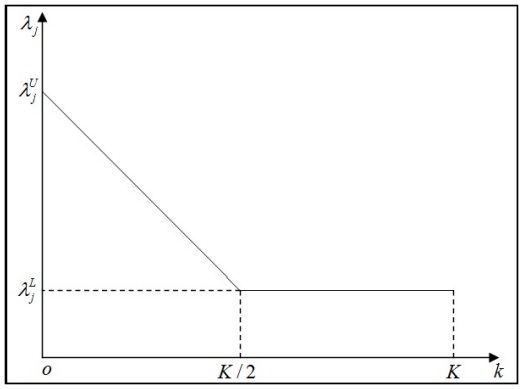

Figure 1. Disturbance Factor λ_j

Based on the above detailed explanation, we can get the general procedure of SGPSO in Table 1:

Table 1. Pseudo code of SGPSO

Line	Procedure of SGPSO
1	**Begin**
2	Set $M = 40$, K, $\omega'_m = 0.5$, $\omega'_{std} = 0.01$, $\omega_m = 0.5$, $\omega_{std} = 0.01$, $\gamma = 0.1$;
3	Set $\omega_i = 0 \ \ (i = 1, \cdots, M)$, x^U, x^L, $\lambda_j^U = 10^{-3}(x_j^U - x_j^L)$, $\lambda_j^L = 10^{-6}(x_j^U - x_j^L)$.
4	Generate a random swarm P with M particles in $[x^L, x^U]$
5	**For** $k = 1$ to K
6	Generate $\lambda_j (j = 1, \cdots, N)$ according to Eq. (10)
7	$Count = 0, \omega_{sum} = 0$;
8	**For** $i = 1$ to M
9	**If** $f(x_i^k) < f(pb_i)$
10	$pb_i = x_i^k, Count = Count + 1, \ \omega_{sum} = \omega_{sum} + \omega_i$
11	**If** $f(pb_i) < f(gb)$
12	$gb = pb_i$
13	**End If**
14	**End If**
15	**End For**
16	**If** $Count \neq 0$
17	$\omega_m = \omega_{sum} / Count$
18	**End If**
19	$x_{i_{worst}}^{k+1} = \sum_{i \neq i_{worst}} x_i / (M-1), \ \omega_{i_{worst}} = 0.5$
20	**For** $i = 1$ to M
21	**If** $i \neq i_{worst}$
22	Generate $\omega_i (i = 1, \cdots, M)$ according to Eq. (6)
23	**For** $j = 1$ to N
24	$Gb_j = gb_j \times (1 + \lambda_j \times N(0,1))$
25	$v_{i,j}^{k+1} = \omega_i v_{i,j}^k + c_1 r_1 (pb_{i,j} - x_{i,j}^k) + c_2 r_2 (Gb_j - x_{i,j}^k)$
26	$x_{i,j}^{k+1} = x_{i,j}^k + v_{i,j}^{k+1}$
27	**End For**
28	**End If**
29	**End For**
30	**End For**
31	**End**

4. Experimental Results and Analysis

Twenty problem instances are chosen to evaluate the optimization performance of SGPSO, and they are expressed as follows:

The first is Sphere function, defined as:

$$\min\ f_1 = \sum_{i=1}^{N} x_i^2 \tag{11}$$

where global optimum $x^* = (0,0,\cdots,0)$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

The second is Rosenbrock function, defined as:

$$\min\ f_2 = \sum_{i=1}^{N-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{12}$$

where global optimum $x^* = (1,1,\cdots,1)$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

The third is Generalized Rastrigrin function, defined as:

$$\min\ f_3 = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{13}$$

where global optimum $x^* = (0,0,\cdots,0)$ and $f(x^*) = 0$ for $-10 \le x_i \le 10$.

The fourth is Generalized Griewank function, defined as:

$$\min\ f_4 = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{14}$$

where global optimum $x^* = (0,0,\cdots,0)$ and $f(x^*) = 0$ for $-600 \le x_i \le 600$.

The fifth is Ackley's function, defined as:

$$\min\ f_5 = 20 + e - 20\exp(-0.2\sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}}) - \exp(\frac{\sum_{i=1}^{N} \cos(2\pi x_i)}{N}) \tag{15}$$

where global optimum $x^* = (0,0,\cdots,0)$ and $f(x^*) = 0$ for $-32 \le x_i \le 32$.

The sixth is Schwefel's Problem 2.22, defined as:

$$\min\ f_6 = \sum_{i=1}^{N} |x_i| + \prod_{i=1}^{N} |x_i| \tag{16}$$

where global optimum $x^* = (0,0,\cdots,0)$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

The seventh is Schwefel's problem 2.26, defined as:

$$\min\ f_7 = 418.9829N - \sum_{i=1}^{N} (x_i \sin(\sqrt{|x_i|})) \tag{17}$$

where global optimum $x^* = (420.9687, 420.9687, \cdots, 420.9687)$ and $f(x^*) = 0$ for $-500 \le x_i \le 500$.

The eighth is Rotated hyper-ellipsoid function, defined as:

$$\min\ f_8 = \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2 \tag{18}$$

where global optimum $x^* = (0, 0, \cdots, 0)$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

The ninth [17] is Shifted Sphere Function, defined as:

$$\min\ f_9 = \sum_{i=1}^{N} z_i^2 - 450 \tag{19}$$

where $z = x - o$; $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum; global optimum $x^* = o$ and $f(x^*) = -450$ for $-100 \le x_i \le 100$.

The tenth [17] is Shifted Schwefel's Problem 1.2, defined as:

$$\min\ f_{10} = \sum_{i=1}^{N} (\sum_{j=1}^{i} z_j)^2 - 450 \tag{20}$$

where $z = x - o$; $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum; global optimum $x^* = o$ and $f(x^*) = -450$ for $-100 \le x_i \le 100$.

The eleventh [17] is Shifted Rotated High Conditioned Elliptic Function, defined as:

$$\min\ f_{11} = \sum_{i=1}^{N} (10^6)^{\frac{i-1}{N-1}} z_i^2 - 450 \tag{21}$$

where $z = (x - o) \times M$, and $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum, M is orthogonal matrix; global optimum $x^* = o$ and $f(x^*) = -450$ for $-100 \le x_i \le 100$.

The twelfth [17] is Shifted Schwefel's Problem 1.2 with Noise in Fitness, defined as:

$$\min\ f_{12} = \sum_{i=1}^{N} (\sum_{j=1}^{i} z_j)^2 (1 + 0.4 * | N(0,1) |) - 450 \tag{22}$$

where $z = x - o$; $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum; global optimum $x^* = o$ and $f(x^*) = -450$ for $-100 \le x_i \le 100$.

The thirteenth [17] is Shifted Rosenbrock's Function, defined as:

$$\min\ f_{13} = \sum_{i=1}^{N-1} (100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2) + 390 \tag{23}$$

where $z = x - o + 1$; $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum; global optimum $x^* = o$ and $f(x^*) = 390$ for $-100 \le x_i \le 100$.

The fourteenth [17] is Shifted Rotated Griewank's Function without Bounds, defined as:

$$\min\ f_{14} = \sum_{i=1}^{N} \frac{z_i^2}{4000} - \prod_{i=1}^{N} \cos(\frac{z_i}{\sqrt{i}}) + 1 - 180 \tag{24}$$

where $z = (x-o) \times M$, and $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum, M is linear transformation matrix; global optimum $x^* = o$ and $f(x^*) = -180$.

The fifteenth [17] is Shifted Rotated Ackley's Function with Global Optimum on Bounds, defined as:

$$\min\ f_{15} = 20 + e - 20\exp(-0.2\sqrt{\frac{\sum_{i=1}^{N} z_i^2}{N}}) - \exp(\frac{\sum_{i=1}^{N} \cos(2\pi z_i)}{N}) - 140 \tag{25}$$

where $z = (x-o) \times M$, and $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum, specially, $o_{2j-1} = -32$, and o_{2j} are randomly distributed in the search range, for $j = 1,2,\cdots,\lfloor N/2 \rfloor$. M is linear transformation matrix; global optimum $x^* = o$ and $f(x^*) = -140$ for $-32 \le x_i \le 32$.

The sixteenth [17] is as follows:

$$\min \quad f_{16} = \sum_{i=1}^{N} i x_i^4 + rand[0,1), \quad -1.28 \le x_i \le 1.28 \tag{26}$$

The seventeenth [17] is as follows:

$$\begin{aligned} \min \quad f_{17} &= \frac{\pi}{N}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{N-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} \\ &+ \sum_{i=1}^{N} u(x_i, 10, 100, 4), \quad -50 \le x_i \le 50 \end{aligned} \tag{27}$$

where $y_i = 1 + (x_i + 1)/4$ and

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

The eighteenth [17] is Shifted Rotated Rastrigin's Function, defined as:

$$\min \quad f_{18} = \sum_{i=1}^{N} (z_i^2 - 10\cos(2\pi z_i) + 10) - 330 \tag{28}$$

where $z = (x-o) \times M$, and $o = [o_1, o_2, \cdots, o_N]$ is the shifted global optimum, M is linear transformation matrix whose condition number is equal to 2; global optimum $x^* = o$ and $f(x^*) = -330$ for $-5 \le x_i \le 5$.

The nineteenth [17] is as follows:

$$\min \quad f_{19} = \max_i \{|x_i|\} \tag{29}$$

Global optimum $f(x^*) = 0$ for $-100 \le x_i \le 100$.

The twentieth [17] is as follows:

$$\min \quad f_{20} = \sum_{i=1}^{N} \lfloor x_i + 0.5 \rfloor^2 \tag{30}$$

Global optimum $f(x^*) = 0$ for $-100 \le x_i \le 100$.

Besides SGPSO, the other three PSO algorithms are also considered to solve the above twenty unconstrained optimization problems, and they are PSO, BBPSO and CPSO respectively. Moreover, their parameters setting are as follows: For PSO, $M = 40$, $c_1 = 2$, $c_2 = 2$, $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$; For BBPSO, $M = 40$; For CPSO, $M = 40$, $c_1 = 2$, $c_2 = 2$, $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$; For the SGPSO, $M = 40$, $c_1 = 2$, $c_2 = 2$, $\omega'_m = 0.5$, $\omega'_{std} = 0.01$, $\gamma = 0.1$; $\omega_m = 0.5$ (Initial value); $\omega_{std} = 0.01$. When solving the problems with dimension size $N = 10$, the maximal generation numbers of four algorithms are set at 1000. When N reaches 30 and 50, the maximal generation numbers are set at 2000 and 3000, respectively. 50 independent runs are carried out for each function, and the means and standard deviations of the best function values averaged over 50 runs on $f_1 - f_{20}$ are reported in Tables 2-4.

Table 2. Comparison of PSO, BBPSO, CPSO and SGPSO on f_1-f_{20} (N=10) with Means and (Standard Deviations)

Fun/Alg	PSO	BBPSO	CPSO	SGPSO
f_1	3.9767e-024	**7.3584e-078**	9.2250e-025	1.0796e-055
	(1.4272e-023)	**(3.2083e-077)**	(2.8015e-024)	(4.9226e-055)
f_2	2.0091e+005	2.0073e+004	8.3870e+000	**1.7152e+000**
	(4.0361e+005)	(1.4141e+005)	(1.8026e+001)	**(8.5387e-001)**
f_3	4.0628e+000	6.0893e+000	**2.8880e+000**	5.3529e+000
	(1.8924e+000)	(3.1503e+000)	**(1.4251e+000)**	(2.8561e+000)
f_4	8.2221e-002	7.1092e-002	7.8627e-002	**3.1385e-002**
	(4.2924e-002)	(4.0363e-002)	(4.2238e-002)	**(3.7308e-002)**
f_5	5.8488e-013	2.3103e-002	1.8002e-013	**2.6645e-015**
	(1.3679e-012)	(1.6336e-001)	(2.2656e-013)	**(0)**
f_6	2.2489e-010	8.0000e+000	2.0934e-012	**3.6395e-027**
	(8.4989e-010)	(2.7405e+001)	(6.5170e-012)	**(2.3998e-026)**
f_7	8.6457e+002	8.6241e+002	9.6688e+002	**7.4162e+002**
	(2.8862e+002)	**(2.4715e+002)**	(2.7934e+002)	(2.7184e+002)
f_8	1.0000e+002	5.8021e-017	3.1944e-007	**2.0765e-018**
	(7.0711e+002)	(1.7721e-016)	(6.9786e-007)	**(9.7312e-018)**
f_9	2.1537e+002	-4.0733e+002	-4.3095e+002	**-4.5000e+002**
	(6.6839e+002)	(1.1287e+002)	(9.3319e+001)	**(2.3838e-005)**

f_{10}	4.1714e+001	-3.7848e+002	-4.5000e+002	-4.5000e+002
	(7.6308e+002)	(1.0987e+002)	**(1.1156e-006)**	(4.8229e-004)
f_{11}	4.7875e+006	5.5553e+005	6.1126e+005	**2.0221e+005**
	(8.3436e+006)	(7.9943e+005)	(1.1219e+006)	**(1.6163e+005)**
f_{12}	2.6694e+002	-3.5593e+002	-4.4703e+002	**-4.5000e+002**
	(1.4074e+003)	(1.1797e+002)	(2.0820e+001)	**(9.6124e-004)**
f_{13}	4.6198e+007	1.0554e+006	3.6792e+005	**4.5720e+002**
	(8.2800e+007)	(3.7345e+006)	(2.2141e+006)	**(1.1523e+002)**
f_{14}	-1.5281e+002	-1.7874e+002	-1.7854e+002	**-1.7960e+002**
	(4.2899e+001)	(2.8899e+000)	(7.5434e+000)	**(2.0418e-001)**
f_{15}	-1.1964e+002	-1.1960e+002	-1.1964e+002	**-1.1976e+002**
	(8.7840e-002)	(9.0308e-002)	(1.0922e-001)	(9.7263e-002)
f_{16}	2.8169e-003	2.0782e-003	**1.1950e-003**	1.3717e-003
	(1.4028e-003)	(1.3533e-003)	**(5.5860e-004)**	(1.0277e-003)
f_{17}	2.6870e-025	1.2440e-002	**8.2851e-026**	3.1996e-010
	(8.8512e-025)	(6.1563e-002)	**(5.4088e-025)**	(9.0108e-011)
f_{18}	-2.8213e+002	-2.8239e+002	**-2.9395e+002**	-2.8972e+002
	(1.2433e+001)	(1.4028e+001)	(1.0521e+001)	**(8.8972e+000)**
f_{19}	7.2209e-007	1.3175e-017	1.8726e-007	**7.6387e-019**
	(7.6876e-007)	(3.6790e-017)	(1.9345e-007)	**(1.9532e-018)**
f_{20}	0	0	0	0
	(0)	(0)	(0)	(0)

Table 3. Comparison of PSO, BBPSO, CPSO and SGPSO on f_1-f_{20} (N=30) with Means and (Standard Deviations)

Fun/Alg	PSO	BBPSO	CPSO	SGPSO
f_1	2.0000e+002	3.6735e-032	3.3931e-013	**2.1556e-039**
	(1.4142e+003)	(1.1056e-031)	(3.6301e-013)	**(5.3124e-039)**
f_2	3.4221e+005	2.4130e+005	5.5583e+001	**2.2426e+001**
	(4.7696e+005)	(4.3071e+005)	(4.6954e+001)	**(7.6117e-001)**
f_3	4.7376e+001	6.4314e+001	**2.2901e+001**	2.9272e+001
	(3.3081e+001)	(2.1129e+001)	**(7.0480e+000)**	(9.8580e+000)
f_4	1.4464e-002	1.3913e-002	1.0981e-002	**3.7929e-003**
	(1.4674e-002)	(1.4171e-002)	(1.6654e-002)	**(6.7869e-003)**
f_5	3.6666e+000	2.6259e+000	8.5236e-008	**1.3110e-014**
	(6.6098e+000)	(6.4074e+000)	(6.0112e-008)	**(2.9949e-015)**
f_6	2.0934e+002	3.0800e+002	1.2369e+002	**2.6949e-002**
	(1.1941e+002)	(1.6640e+002)	(8.7003e+001)	**(1.9035e-001)**
f_7	3.5889e+003	3.6509e+003	3.5526e+003	**2.9329e+003**
	(8.0820e+002)	**(5.6864e+002)**	(8.3683e+002)	(5.9178e+002)
f_8	1.9309e+004	1.0019e+004	7.4800e+001	**3.3604e-001**
	(9.0728e+003)	(6.4836e+003)	(4.3733e+001)	**(5.3258e-001)**
f_9	1.2880e+004	1.4979e+003	-2.8638e+002	**-4.5000e+002**
	(6.0434e+003)	(2.6033e+003)	(4.6232e+002)	**(2.0266e-003)**
f_{10}	2.4781e+004	9.7679e+003	8.7314e+000	**-4.0008e+002**
	(1.2831e+004)	(9.6687e+003)	(2.1008e+002)	**(3.3295e+001)**
f_{11}	8.0832e+007	1.2707e+007	1.3486e+007	**3.6928e+006**

	(6.4269e+007)	(1.0862e+007)	(1.5666e+007)	**(1.9080e+006)**
f_{12}	2.9099e+004	1.7757e+004	**5.3872e+003**	6.2185e+003
	(1.5754e+004)	(1.2569e+004)	**(2.7846e+003)**	(2.8808e+003)
f_{13}	3.6893e+009	3.0740e+008	2.5015e+006	**7.4083e+002**
	(3.8323e+009)	(3.8005e+008)	(1.7684e+007)	**(3.8425e+002)**
f_{14}	4.5175e+002	-9.9367e+001	-3.9010e+001	**-1.7887e+002**
	(3.0511e+002)	(7.6268e+001)	(2.1765e+002)	**(2.5969e-002)**
f_{15}	-1.1905e+002	-1.1900e+002	-1.1904e+002	**-1.1927e+002**
	(6.6267e-002)	**(5.1846e-002)**	(6.1850e-002)	(1.1950e-001)
f_{16}	1.9103e+000	3.3664e-001	**6.1608e-003**	7.2195e-003
	(3.3100e+000)	(1.5932e+000)	(2.7364e-003)	**(2.6493e-003)**
f_{17}	3.5298e-002	1.2657e-001	6.2201e-003	**1.1659e-009**
	(8.5333e-002)	(1.9812e-001)	(2.4870e-002)	**(3.4961e-010)**
f_{18}	-7.9498e+001	-1.4323e+002	**-2.1489e+002**	-1.8563e+002
	(5.3823e+001)	**(4.9723e+001)**	(5.7773e+001)	(5.2737e+001)
f_{19}	6.6866e+000	4.8027e+000	4.0579e-001	**1.9257e-002**
	(2.2078e+000)	(2.8513e+000)	(2.1571e-001)	**(3.4406e-002)**
f_{20}	2.0002e+002	3.0000e-001	0	0
	(1.4142e+003)	(5.4398e-001)	(0)	(0)

Table 4. Comparison of PSO, BBPSO, CPSO and SGPSO on f_1-f_{20} (N=50) with Means and (Standard Deviations)

Fun/Alg	PSO	BBPSO	CPSO	SGPSO
f_1	3.2000e+003	4.0000e+002	2.0272e-009	**8.6547e-032**
	(5.8693e+003)	(1.9795e+003)	(4.9491e-009)	**(2.2575e-031)**
f_2	2.4355e+005	2.4093e+005	9.3744e+001	**4.7108e+001**
	(4.2949e+005)	(4.3094e+005)	(5.0670e+001)	**(1.7420e+001)**
f_3	1.6629e+002	1.8074e+002	**5.2208e+001**	5.2992e+001
	(9.9552e+001)	(5.6418e+001)	**(1.1546e+001)**	(1.4930e+001)
f_4	3.4341e+001	3.6142e+000	5.2183e-003	**3.9940e-003**
	(5.7363e+001)	(1.7840e+001)	(8.1436e-003)	**(6.8461e-003)**
f_5	8.6563e+000	1.1508e+001	7.1814e-006	**3.1157e-014**
	(7.5701e+000)	(8.9167e+000)	(1.7544e-005)	**(8.1043e-015)**
f_6	6.8818e+002	9.0400e+002	3.7425e+002	**7.1950e+000**
	(1.9752e+002)	(2.3901e+002)	(1.2473e+002)	**(3.5725e+001)**
f_7	7.7034e+003	6.5054e+003	6.3017e+003	**5.9653e+003**
	(1.0704e+003)	**(7.5629e+002)**	(1.2716e+003)	(8.1347e+002)
f_8	6.4145e+004	3.4487e+004	1.1567e+003	**4.0282e+001**
	(1.3678e+004)	(1.3519e+004)	(6.4634e+002)	**(4.6241e+001)**
f_9	3.7884e+004	8.8996e+003	-3.5700e+002	**-4.4997e+002**
	(1.0910e+004)	(5.6767e+003)	(6.5762e+002)	**(1.1625e-002)**
f_{10}	8.0455e+004	4.1396e+004	1.0010e+004	**1.4853e+003**
	(2.6939e+004)	(1.6961e+004)	(2.9793e+003)	**(9.5868e+002)**
f_{11}	5.5597e+008	4.3972e+007	1.0574e+008	**9.3144e+006**
	(3.4627e+008)	(2.0362e+007)	(1.5170e+008)	**(2.6953e+006)**
f_{12}	1.1680e+005	8.1957e+004	4.7071e+004	**3.6564e+004**
	(3.0770e+004)	(2.3154e+004)	(1.7547e+004)	**(7.7249e+003)**

f_{13}	1.5720e+010	2.9663e+009	**7.1238e+002**	7.7036e+002
	(1.0053e+010)	(2.4723e+009)	(3.0154e+002)	**(2.3249e+002)**
f_{14}	1.4096e+003	1.7753e+002	**-1.7881e+002**	-1.7744e+002
	(5.2519e+002)	(2.7799e+002)	**(3.9245e+000)**	(6.7628e+000)
f_{15}	-1.1886e+002	-1.1883e+002	-1.1886e+002	**-1.1909e+002**
	(3.9681e-002)	(4.1169e-002)	(5.0763e-002)	(9.8683e-002)
f_{16}	1.7218e+001	3.2648e+000	**1.2618e-002**	1.7311e-002
	(1.7443e+001)	(5.5030e+000)	(4.6801e-003)	**(4.6127e-003)**
f_{17}	5.0259e-001	9.2364e-002	8.7082e-003	**2.4881e-003**
	(5.7496e-001)	(2.6485e-001)	(2.1802e-002)	**(1.2313e-002)**
f_{18}	1.8870e+002	1.3620e+001	**-1.8031e+002**	-6.9370e+001
	(1.0671e+002)	**(5.8516e+001)**	(8.8354e+001)	(6.5918e+001)
f_{19}	2.6216e+001	8.9194e+001	2.5711e+000	**2.2981e+000**
	(2.8933e+000)	(5.8912e+000)	**(5.4951e-001)**	(9.2149e-001)
f_{20}	4.0005e+003	4.0206e+002	6.0000e-002	**0**
	(6.9986e+003)	(1.9796e+003)	(2.3990e-001)	**(0)**

Where Fun/Alg denotes Function/Algorithm. Table 2 presents the means and standard deviations of the 50 runs of the four algorithms on the twenty test functions with $N = 10$. The bold font indicates the best results for each case. According to Table 2, SGPSO is superior to the other three algorithms on functions f_2, f_4, f_5, f_6, f_8, f_9, f_{11}, f_{12}, f_{13}, f_{14} and f_{19}, and the means and standard deviations obtained by SGPSO are the best. For functions f_3, f_{16}, f_{17}, CPSO can obtain better results than the other three algorithms. In the case of function f_1, BBPSO can produce the best mean and standard deviation. Regarding function f_{20}, the results obtained by all the four algorithms are the same.

Table 3 reports the optimization results of the 50 runs of the four algorithms on the twenty test functions with $N = 30$. From Table 3, we notice that SGPSO shows clearly a better performance over the other three algorithms for f_1, f_2, f_4, f_5, f_6, f_8, f_9, f_{10}, f_{11}, f_{13}, f_{14}, f_{17} and f_{19}. With respect to functions f_3 and f_{12}, CPSO can generate better solutions than the other three algorithms. In the case of function f_{20}, SGPSO and CPSO can obtain the same means and standard deviations that are better than those obtained by PSO and BBPSO.

Table 4 lists the means and standard deviations of the 50 runs of the four algorithms on the twenty test functions with $N = 50$. It is clear from Table 4 that the results of SGPSO are better than those of the other three algorithms for f_1, f_2, f_4, f_5, f_6, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{17} and f_{20}. The means and standard deviations from CPSO are better than those from the other three algorithms for f_3 and f_{14}.

Figures. 2-21 depict the average optimization curves of four PSO algorithms for functions $f_1 - f_{20}$ with dimension size $N = 50$.

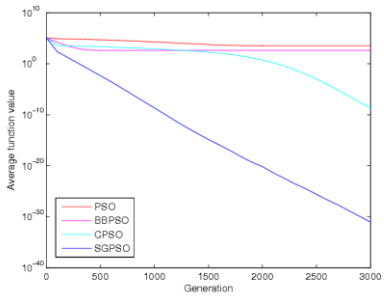

Figure 2. AOCs obtained by Four PSO Algorithms for f_1

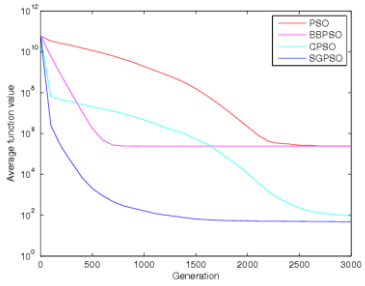

Figure 3. AOCs obtained by Four PSO Algorithms for f_2

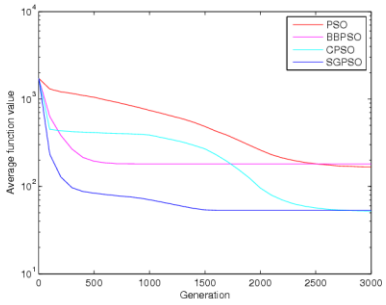

Figure 4. AOCs obtained by Four PSO Algorithms for f_3

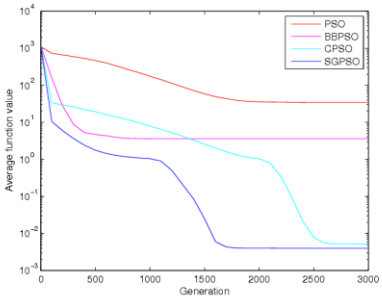

Figure 5. AOCs obtained by Four PSO Algorithms for f_4

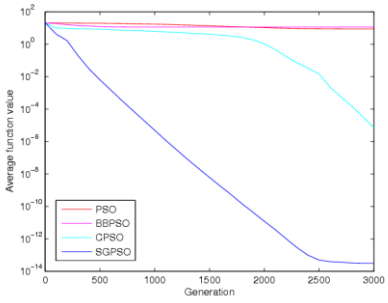

Figure 6. AOCs obtained by Four PSO Algorithms for f_5

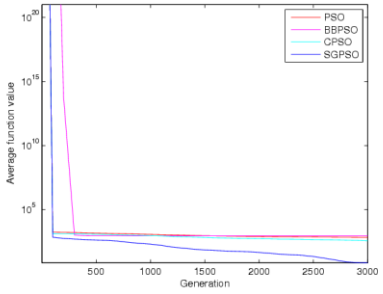

Figure 7. AOCs obtained by Four PSO Algorithms for f_6

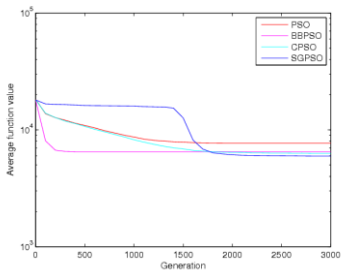

Figure 8. AOCs obtained by Four PSO Algorithms for f_7

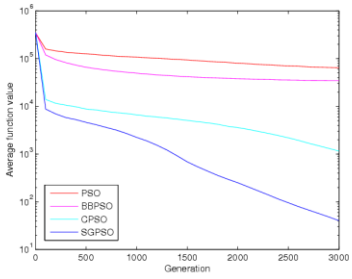

Figure 9. AOCs obtained by Four PSO Algorithms for f_8

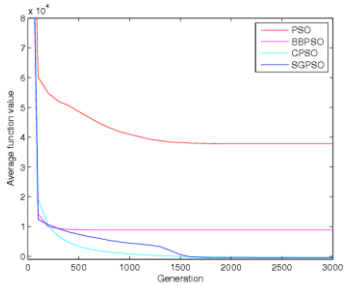

Figure 10. AOCs obtained by Four PSO Algorithms for f_9

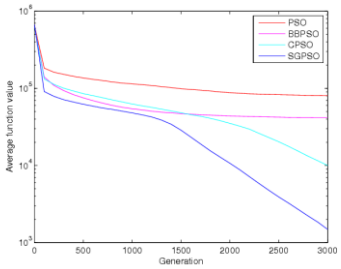

Figure 11. AOCs obtained by Four PSO Algorithms for f_{10}

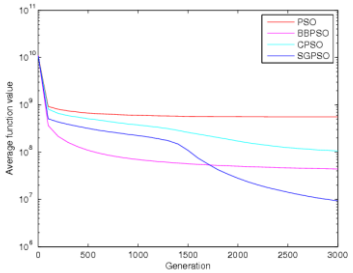

Figure 12. AOCs obtained by Four PSO Algorithms for f_{11}

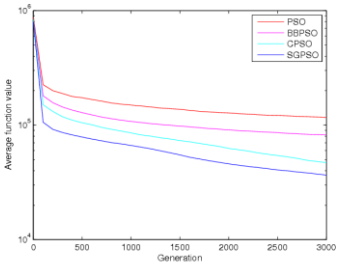

Figure 13. AOCs obtained by Four PSO Algorithms for f_{12}

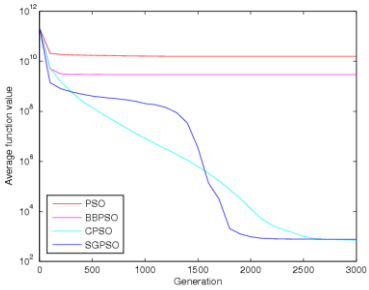

Figure 14. AOCs obtained by Four PSO Algorithms for f_{13}

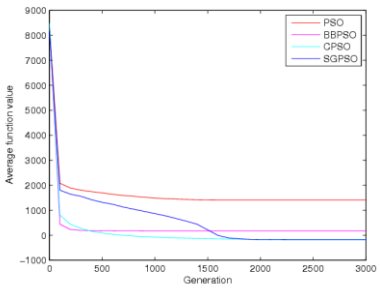

Figure 15. AOCs obtained by Four PSO Algorithms for f_{14}

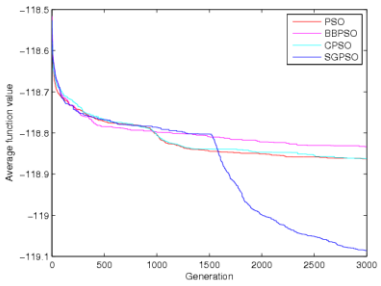

Figure 16. AOCs obtained by Four PSO Algorithms for f_{15}

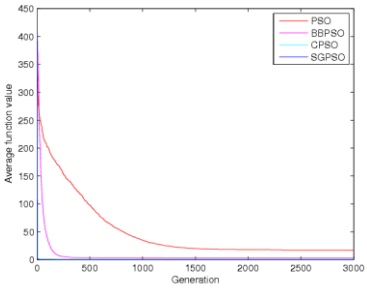

Figure 17. AOCs obtained by Four PSO Algorithms for f_{16}

ISSN: 2005-4254 IJSIP

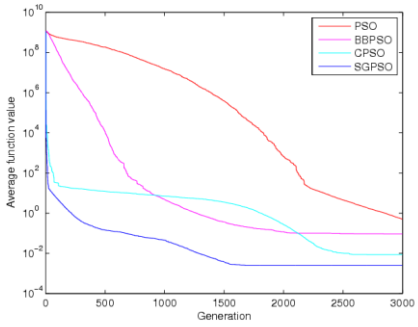

Figure 18. AOCs obtained by Four PSO Algorithms for f_{17}

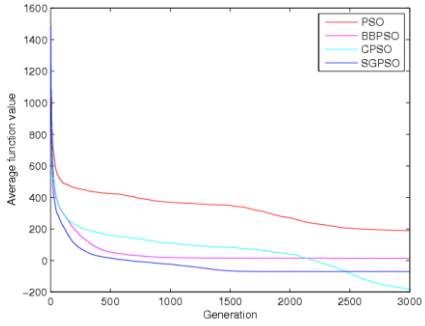

Figure 19. AOCs obtained by Four PSO Algorithms for f_{18}

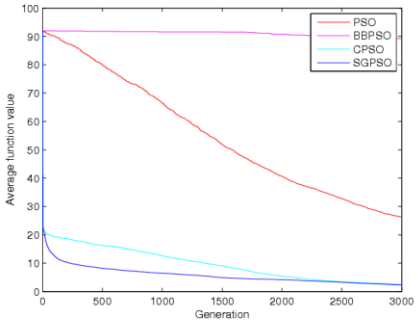

Figure 20. AOCs obtained by Four PSO Algorithms for f_{19}

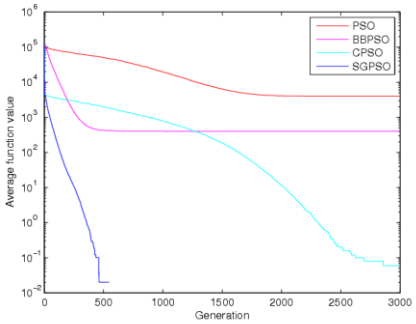

Figure 21. AOCs obtained by Four PSO Algorithms for f_{20}

5. Conclusion and Discussion

This paper introduces a self-adaptive global particle swarm optimization (SGPSO) algorithm. There three modifications in the SGPSO model: first, each particle yields inertia weight by using Gaussian distribution whose mean comes from the inertia weights of the successful particles with a large probability. This adaptive adjusting strategy enables SGPSO to fully develop the solution space. Second, the original velocity updating and position updating are excluded from the updating of the worst particle, and the averaging of the other particles is adopted, which is useful to improve the quality of the population. Finally, a global disturbance is introduced to get rid of the premature convergence of SGPSO. More specifically, a large disturbance happens in the early evolution, and a small disturbance happens in the late evolution. Therefore, the proposed algorithm has the capacity of controlling the global searching and local searching. A set of benchmark functions are chosen to investigate the performance of SGPSO and the other three algorithms. Experimental results indicate that SGPSO are better than the other three PSO versions on finding the best solutions of most unconstrained optimization problems.

ISSN: 2005-4254 IJSIP

Acknowledgements

This work was supported by Science Fundamental Research Project of Jiangsu Normal University under Grant 9212812101.

References

[1] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", Proc.IEEE International Conference on Neural Networks, Perth, WA, (**1995**), pp.1942-1948.
[2] R. Storn, "Differential evolution design of an IIR-filter", IEEE International Conference on Evolutionary Computation, Nagoya, (**1996**), pp. 268-273.
[3] J. H. Holland, "Adaptive in natural and artificial systems", Ann Arbor, Univ. Mich. Press, MI, (**1975**).
[4] Z. W. Geem, J. H. Kim and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search", Simulation, vol. 76, no. 2, (**2001**), pp. 60-68.
[5] D. X. Zou, L. Q. Gao and J. H. Wu, *et al.* "Novel global harmony search algorithm for unconstrained problems", Neurocomputing, vol. 73, nos. 16-18, (**2010**), pp. 3308-3318.
[6] D. X. Zou, L. Q. Gao and J. H. Wu, *et al.* "A novel global harmony search algorithm for reliability problems", Computers & Industrial Engineering, vol. 58, no. 2, (**2010**), pp. 307-316.
[7] P. F. Wu, L. Q. Gao and D. X. Zou, *et al.* "An improved particle swarm optimization algorithm for reliability problems", ISA Transactions, vol. 50, no. 1, (**2011**), pp. 71-81.
[8] T. Navalertporn and N. V. Afzulpurkar, "Optimization of tile manufacturing process using particle swarm optimization", Swarm and Evolutionary Computation, vol. 1, no. 2, (**2011**), pp. 97-109.
[9] B. V. Babu, P. G. Chakole and J. H. Syed Mubeen, "Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor", Chemical Engineering Science, vol. 60, no. 17, (**2005**), pp. 4822-4837.
[10] H. A. Mosalman Yazdi and N. H. Ramli Sulong, "Optimization of Off-Centre bracing system using Genetic Algorithm", Journal of Constructional Steel Research, vol. 67 ,no. 10, (**2011**), pp. 1435-1441.
[11] L. Zhao, F. Qian and Y. P. Yang, *et al.* "Automatically extracting T-S fuzzy models using cooperative random learning particle swarm optimization", Applied Soft Computing, vol. 10, no. 3, (**2010**), pp. 938-944.
[12] Y. H. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization", Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, (**1999**), pp. 945-1950.
[13] J. Kennedy, "Bare bones particle swarms", Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, (**2003**), pp. 80-87.
[14] Y. X. Zhao, W. Zu and H. T. Zeng, "A modified particle swarm optimization via particle visual modeling analysis", Computers & Mathematics with Applications, vol. 57, nos. 11-12, (**2009**), pp. 2022-2029.
[15] G. Ma, W. Zhou and X. L. Chang, "A novel particle swarm optimization algorithm based on particle migration", Applied Mathematics and Computation, vol. 218, no. 11, (**2012**), pp. 6620-6626.
[16] Y. Liu, Z. Qin, Z.W. Shi and J. Lu, "Center particle swarm optimization", Neurocomputing, vol. 70, nos. 4-6, (**2007**), pp. 672-679.
[17] P. N. Suganthan, N. Hansen and J. J. Liang, *et al.* "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization", Technical Report, Nanyang Technological University, Singapore, May 2005 (KanGAL Report# 2005005, IIT Kanpur, India).

Author

Dexuan Zou, he was born in China, in April 1982. He received the B.S. degree in Electronic Information Science and Technology from Liaoning University, Shenyang, P.R.China in 2005. He received the M. S. degree in Signal and Information Processing from Northeastern University, Shenyang, P.R.China in 2008. He received the Ph.D. degree in Control Theory and Control Engineering from Northeastern University, Shenyang, P.R.China in 2011. He is recently a lecturer in Jiangsu Normal University. His research interests include optimization and its applications.

