INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 11(6):804-818, December 2016.

# The Particle Swarm Optimization Algorithm with Adaptive Chaos Perturbation

L. Mengxia, L. Ruiquan, D. Yong

**Li Mengxia**
1. School of Information and Mathematics, Yangtze University
Jingzhou Hubei 434023, China
2. Petroleum Engineering College, Yangtze University
Wuhan Hubei 430100, China
3. The Branch of Key Laboratory of CNPC for Oil and Gas Production, Yangtze University
Wuhan Hubei 430100, China
4. Key Laboratory of Exploration Technologies for Oil and Gas Resources, Yangtze University
Wuhan Hubei 430100, China
limengxia81@126.com

**Liao Ruiquan**
1. Petroleum Engineering College, Yangtze University
Wuhan Hubei 430100, China
2. The Branch of Key Laboratory of CNPC for Oil and Gas Production, Yangtze University
Wuhan Hubei 430100, China
3. Key Laboratory of Exploration Technologies for Oil and Gas Resources, Yangtze University
Wuhan Hubei 430100, China
liaoruiquan@263.net

**Dong Yong***
1. School of Information and Mathematics,Yangtze University
Jingzhou Hubei 434023, China
2. The Branch of Key Laboratory of CNPC for Oil and Gas Production, Yangtze University
Wuhan Hubei 430100, China
3. Key Laboratory of Exploration Technologies for Oil and Gas Resources, Yangtze University
Wuhan Hubei 430100, China
*Corresponding author: dongyong80@126.com

**Abstract:** Aiming at the two characteristics of premature convergence of particle swarm optimization that the particle velocity approaches 0 and particle swarm congregate, this paper learns from the annealing function of the simulated annealing algorithm and adaptively and dynamically adjusts inertia weights according to the velocity information of particles to avoid approaching 0 untimely. This paper uses the good uniformity of Anderson chaotic mapping and performs chaos perturbation to part of particles based on the information of variance of the population's fitness to avoid the untimely aggregation of particle swarm. The numerical simulations of five test functions are performed and the results are compared with several swarm intelligence heuristic algorithms. The results shows that the modified algorithm can keep the population diversity well in the middle stage of the iterative process and it can improve the mean best of the algorithm and the success rate of search.
**Keywords:** Particle Swarm Optimization, inertia weight,population diversity, expected velocity, chaos perturbation.

## 1 Introduction

Particle swarm optimization (PSO) is an evolutionary algorithms (EAs). It stems from the simulation of group behavior for birds swarm's foraging [1, 2]. The parameters and structures of

PSO are very simple and are apt to be realized. It has parallelism essentially. It has no requirement to the properties of objective functions. It has good adaptation and gains wide concern. It is applied in many fields such as decision feedback equalizer [3, 4], parameter identification [5, 6], power dispatch [7] and mechanical control [8,9], etc.

The particle swarm of the standard PSO is apt to get into local best position and occurs premature convergence phenomenon. The premature convergence has two characteristics that the velocity of particle swarm approach 0 and the particles aggregate in a small region.

In order to avoid early approaching to 0 for the velocity of particle swarm, the inertia weights of standard PSO are modified from fixed values to variation values in the references [10-15], and the performance of PSO is improved in some extent.

The reference [2] proposed a method that the inertia weight decreases within the increasing of iterations. It emphasizes the global search in the earlier stage of evolution and intensifies the local exploitation in the later stage of evolution. But the effect on the complex problem is not obvious. The reference [10] use the fuzzy controller to adjust the inertia weight, but it is complex to build the fuzzy rule. The reference [11] proposed the method to set the inertia weight randomly and obtained a greater effect on pursuing the optimization. The reference [12] used the chaotic mapping to change the inertia weight and the inertia weight changes according to the Logistic chaotic mapping. The reference [13, 14] used different exponential forms to build the setting method of the inertia weight which deceases according to the exponential rule. The reference [15] uniformly adjusted the inertia weight for the all particles according to the mean velocity of the particle swarm, and the numerical results showed that it can keep the population diversity in the middle stage of evolution.

For PSO algorithm, the particle,s position is different. The global best particle and the personal best particle are in the leading position. In the process of evolution, the other particles are affected by the two particles and are close to them. The global best particle and the personal best particle are considered as on group which is called the dominant group. The other particles are taken as another group which is called the ordinary group. Obviously, it is appropriate to deal with the dominant group and the ordinary group respectively.

This paper uses the smaller inertia weight to develop the local development capabilities for the particles in the dominant group. For the particles in the ordinary group, the method in reference [15] that adjusts the inertia weight based on the mean velocity of swarm is modified and the inertia weight corresponding to each particle is adjusted respectively according to the expected values of velocity of particle swarm to avoid the swarm velocity early approaching 0.

For the second characteristics of the premature convergence, the reference [16] considered that the variance of the population's fitness can reflect the concentration degree of population. If the value of is less than some specified threshold, the particle swarm aggregate and the premature convergence occurs.

$$\sigma^2 = \sum_{k=1}^{n} (\frac{f_k - f_m}{f})^2 \tag{1}$$

$$f = max \left\{ 1, \max_k \{|f_k - f_{avg}|\} \right\} \tag{2}$$

In Eq. 1 and Eq. 2, $f_k$denotes the fitness value of the $k$-th particle.$f_m$denotes the mean value of the variance of the population's fitness. $f$denotes the normalized parameter.

The reference [16] didn't point out how to determine the appropriate threshold value. To analyze a large number of numerical simulations, the results showed that the variances $\sigma^2$keep stabilization and basically equal in the subsequent several evolutionary when the premature convergence occurs. Hence, this paper compares the corresponding variances$\sigma^2$between two

adjacent iterations. If the change of $\sigma^2$value is very small, it considers that the premature convergence occurs and needs to set the position of particle swarm to advance the population diversity. One common method is to use the Logistic chaotic mapping performing perturbation to part of particles.

It is easy to verify that the distribution of the chaotic sequence generated by the Logistic chaotic mapping has the characteristic that is big at both ends and small in the middle and the uniformity of the chaotic sequence is poor. But the chaotic sequence generated by the Anderson chaotic mapping has better uniformity. So it is appropriate to use the Anderson chaotic mapping to perform the perturbation to part of particles.

Based on the above analysis, this paper firstly considers the expected value of population velocity, and changes the inertia weights respectively corresponding to the particles according to the category of particles. And then it performs chaos perturbation to part of particles based on the difference of the variances of the population's fitness between two adjacent iterations. Finally, the numerical simulations are performed and the results show that the new algorithm has higher performance.

## 2 Standard PSO algorithm

PSO algorithm is simulating the foraging process of the bird flock. Particle denotes the individual. It has position and velocity, but has no volume and mass. Multiple particles constitute the particle swarm which denote the bird flock. The objective function value corresponding to the particle's position is called the fitness of the particle which is used to evaluate the good points and the bad points of the particle. The best position which the individual particle went through is called the personal previous best position. The personal previous best position of the $k$th particle is denoted by $p_k$.The previous best position of the particle swarm is called the swarm best position which is denoted by $p_g$.The evolution of the particle is realized through tracing the personal previous best position and swarm best position.

The evolution of standard PSO algorithm is divided into the evolution of the velocity and the evolution of the position. The evolution equations are shown in Eq. 3 and Eq. 4 [1, 2] as below.

$$v_k^{t+1} = \omega \times v_k^t + c_1 \times r_1 \times (p_k - x_k^t) + c_2 \times r_2 \times (p_g - x_k^t) \tag{3}$$

$$x_k^{t+1} = x_k^t + v_k^{t+1} \tag{4}$$

In Eq. 3 and Eq. 4, $v_k^t$denotes the velocity of the $k$th particle in the $t$th evolution. $x_k^t$denotes the position of the $k$th particle in the $t$th evolution. $r_1$ and $r_2$ denote the independent random numbers on the interval [0, 1]. $c_1$ and $c_2$ denote the learning factors which generally equal 2. $\omega$ denotes the inertia weight which takes fixed value between 0.1 and 0.9.

The value ranges of the particle's velocity and position are usually restricted. If the maximum of the position is determined by $x_{max}$, the maximum of the velocity is $v_{max} = k \times x_{max}$ for $0.1 \leq k \leq 1.0$ [16].

## 3 Modification of the inertia weight

The mean velocity $v_a^t$ of the particle swarm in the $k$th evolution is denoted by Eq. 5.

$$v_a^t = (\sum_{i=1}^{n} \sum_{k=1}^{m} \left| v_{k,i}^{(t)} \right|)/(n \times m) \tag{5}$$

$n$ is the dimension of the problem which is the dimension of the search space. $m$ is the total number of the particles. $v_{k,i}^{(t)}$ is the velocity of the $k$th particle in the $i$th dimensionality as the particle experiences the $t$th evolution. $v_a^{(t)}$ denotes the evolution amplitude of the particle swarm or the evolution step. The larger the value of $v_a^{(t)}$ is, the larger the evolution step is. It means that the search capability of the particle swarm is great. If the value of $v_a^{(t)}$ is small, it means that the search of particle swarm focuses on the local exploitation. In the earlier stage of evolution, the global search capability is expected to be stronger. It means that the value of $v_a^{(t)}$ is expected to be larger. In the later stage of the evolution, the capability of local exploitation is expected to be intensified to increase the capability of finding the maximal solution by the algorithm, and the value of $v_a^{(t)}$ is expected to be smaller.

This paper firstly introduce the expected value of the mean value of particle swarm in the evolution which is shown in Eq. 6 as follow.

$$v_e^t = v_0 \times exp(-(\lambda_1 \times t/T_{max})^{\lambda_2}) \tag{6}$$

In Eq. 6, $v_0$ denotes the mean velocity of the initial particle swarm. $\lambda_1$ and $\lambda_2$ are adjustable parameters which is to control the change rule of the expected value $v_e^t$ of the particle swarm shown in Figure 1. $T_{max}$ denotes the maximal evolution generations.

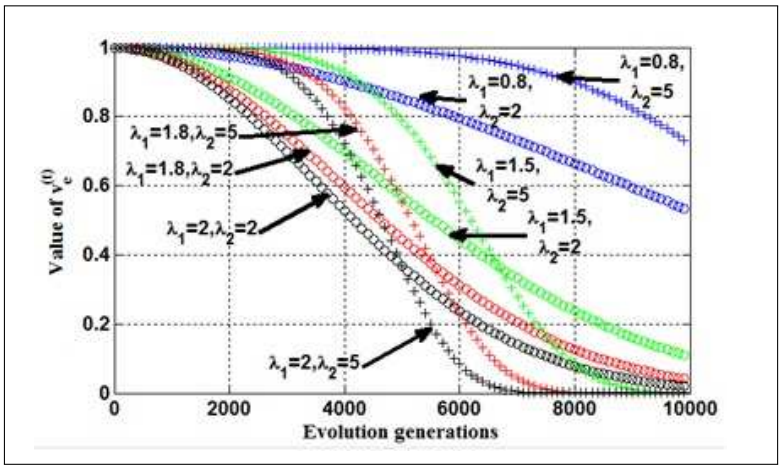


Figure 1: The change rule of $v_e^t$ under different parameters

From Figure 1, it can be seen that the value of $v_e^t$ as $\lambda_1 = 1.8$ and $\lambda_2 = 5$ is suitable to be the expected mean velocity of particle swarm. In this case, in the earlier stage of evolution, the expected value of mean velocity of particle swarm is larger and it helps to keep the global search capability. In the last stage of evolution, the expected value of mean velocity approaches 0 and it helps to improve the local exploitation capability.

$\omega_0$ denotes the initial inertia weight. $v_e^t$ denotes the actual velocity of the $k$th particle. $\omega_k^t$ denotes the inertia weight of the $k$th particle in the $t$th evolution. $\omega_k^{t+1}$ denotes the inertia weight of the $k$th particle in the $(t+1)$th evolution. $\omega_k^{t+1}$ is determined by the following rule.

(1) If the particle is the better particle, $\omega_k^{t+1}$=$\omega_{min}$;

(2) If the particle is the trivial particle, then, define

$$v_{k,a}^{(t)} = (\sum_{i=1}^{n} \left| v_{k,i}^{(t)} \right|)/n \tag{7}$$

there are three cases to be considered as below.

(2.1) If $v_{k,a}^{(t)} > v_e^{(t)}$,then, $\omega_k^{(t+1)} = \omega_k^{(t)} \times p_1$.

(2.2) If $v_{k,a}^{(t)} < v_e^{(t)}$,then, $\omega_k^{(t+1)} = \omega_k^{(t)} \times p_2$.

(2.3) if $v_{k,a}^{(t)} = v_e^{(t)}$,then, $\omega_k^{(t+1)} = \omega_k^{(t)}$.

In the above process, $p_1$ and $p_2$ are constants with $0 < p_1 < 1$ and $p_2 > 1$. It also can take $p_1 = 1/p_2$. If $\omega_k^{(t+1)} > \omega_{max}$, $p_2$ is taken to be 1.05 according to the reference [8] or is determined by the experiment. Then taking $\omega_k^{(t+1)} = \omega_{max}$. $\omega_{max}$ and $\omega_{min}$ denote the upper limit and lower limit of the inertia weight, respectively.

## 4 Anderson chaotic mapping

This paper uses the rule that the difference of $\sigma^2$ between the adjacent evolution process is less than some given value $se$ to recognize the premature convergence, for example, taking $se = 10^{-3}$. If there exists premature convergence, then it needs to perform the perturbation for the current particle to improve the population diversity.

Generally, the researchers use the Logistic chaotic mapping to perform perturbation [18-21], but the sequence generated by the Logistic chaotic mapping has bad uniformity [22]. The Logistic chaotic mapping is shown in Eq. 8 as below.

$$y_{n+1} = \mu \times y_n \times (1 - y_n) \tag{8}$$

When $\mu = 4$, Eq. 8 will generate the chaotic sequence whose value is taken between 0 and 1.

This paper introduce Anderson chaotic mapping to generate chaotic sequence which has good uniformity [23]. Anderson chaotic mapping is shown in Eq. 9 as below.

$$y_i = (Ln(x_i + 1/2) + Ln(2)))/ln(3) \tag{9}$$

where,

$$x_{n+1} = \begin{cases} \frac{3}{2} \times x_n + 1/4 & 0 \leqslant x_n < \frac{1}{2} \\ \frac{1}{2} \times x_n - 1/4 & \frac{1}{2} \leqslant x_n < 1 \end{cases} \quad (n = 1, 2, ...) \tag{10}$$

The sequence $y_i$ is uniform distribution on $(0, 1)$.

In order to compare the uniformity of the two chaotic mapping, the initial value is taken arbitrarily. And then iterate 1000 times according to Eq. 8 and Eq. 9. The interval [0, 1] is divided into 10 subintervals with equal width. The frequency that the elements of the chaotic sequence locate at each subinterval is considered and shown in Figure 2.

If the other initials are taken except that the value of 0.25 and 0.75 cannot be taken because of the occurring of fixed points for Logistic chaotic mapping. The obtained frequency graph is similar to Figure 2. This means that Anderson chaotic mapping has stable uniformity.

## 5 Modified PSO algorithm

The objective function is expressed in Eq. 11.

$$minf(x) = f(x_1, x_2, ..., x_n) \tag{11}$$

Where, $n$ is the dimension of the problem.

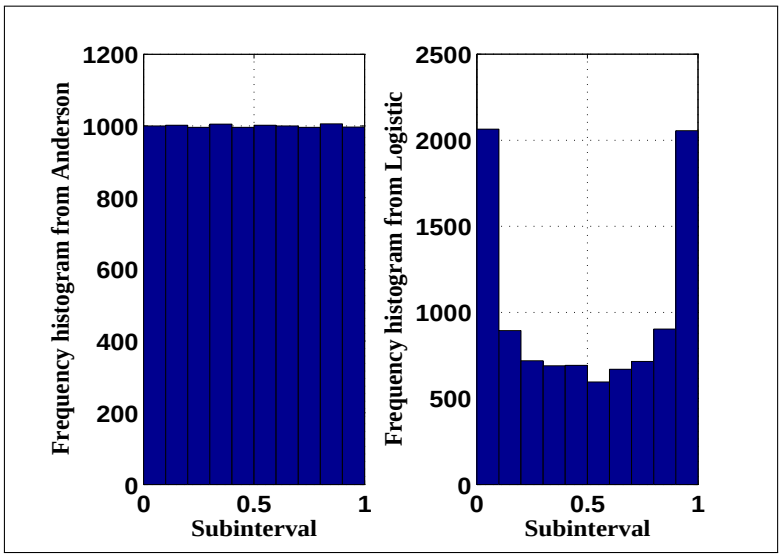


Figure 2: Comparison of the uniformity between two chaotic mappings

### 5.1 Initial particle swarm generated by the chaotic mapping

Firstly, one $m$-dimensional vector is generated randomly and its element is between 0 and 1. Secondly, each element is taken as initial value. And using Anderson chaotic mapping iterates $n-1$ times to generate $m$s $n$-dimensional vector which are denoted by $y_1, y_2, ..., y_m$. Finally, the value range of element of $y_i$ for $i = 1, 2, ..., m$ is transformed to the search space from the interval $[0, 1]$.

Suppose $y \in [a, b]$,then $x \in [c, d]$. And the relation between $x$ and $y$ is shown in Eq. 12.

$$x = c + \frac{y-a}{b-a}(d-c) \tag{12}$$

The initialization of the velocity can be finished analogously.

### 5.2 Modification for evolution process

In Figure.3, after finishing the $t$th evolution process, the variance $\sigma^2(t)$ is obtained by calculating the variance of the population's fitness. And then it is compared with the variance $\sigma^2(t-1)$ in the previous evolution process. If $\left|\sigma^2(t) - \sigma^2(t-1)\right| \leqslant se$, the particles are ordered from small to large according to the fitness value. The particles which account for 61.8% of total particle number after ordering are performed chaotic perturbation. The number of foregoing particles is denoted by $S$. That is, the positions of $S$s particles are initialized again.

### 5.3 Modified PSO algorithm

Step 1 Give the initial value of the inertia weight $\omega_0 = \omega_{max} = 0.95$,$\omega_{min} = 0.05$ learning factors $c_1 = c_2 = 2$, the number of particles $m = 40$, the maximal evolution generations $T_{max} = 10000$, the maximal evolution generations $n$, the control value for starting the chaotic perturbation $se = 10^{-3}$, the search space $[x_{min}, x_{max}]$ and the upper limit of velocity $v_{max}$.

step 2 Randomly generate a $m$-dimensional particle on $[0, 1]$.Use the Anderson mapping shown in Eq. 9 and Eq. 10 to get $m$s $n$-dimensional particles. The $m$s $n$-dimensional particles which are transformed into the search space based on Eq. 12 are denoted by $x_i$ for $i = 1, 2, ..., m$.

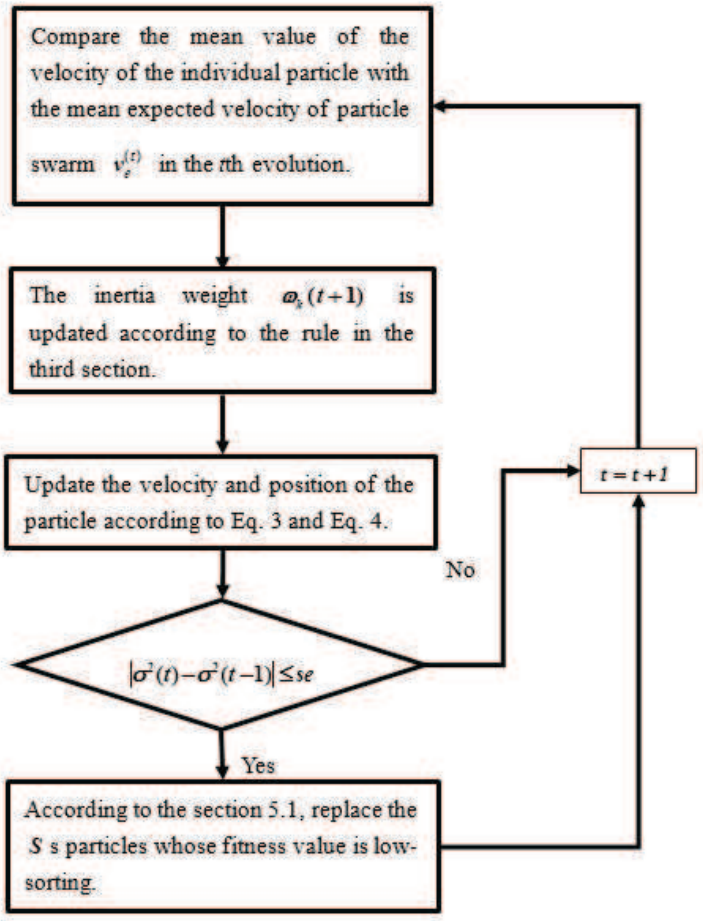


Figure 3: The flow chart of chaos perturbation

Similarly, generate the particle's velocity. Take $v_{max} = x_{max}$. Let the evolution generations be 0, and then turn to step 3.

step 3 Calculate the fitness of each particle and the variance of the population's fitness. Determine the swarm best position $p_g$, the personal previous best position $p_i$ for $i = 1, 2, ..., m$. Turn to step 4.

Step 4 Adjust $\omega$ according to the contents in section 3. Evolve the particle's velocity and position according to Eq. 3 and Eq. 4. And then add one to the evolution generations. Update $p_g$ and $p_i$.Calculate the variance of current population's fitness. Compare the calculating variance with the variance of fitness value before evolution. If the difference is less than $se$, turn to step 5. Otherwise, turn to step 6.

Step 5 Determine the number of particles which need to be performed perturbation, $S$. Use Anderson mapping to generate $S$s particles to replace the $S$s particles whose fitness value order are back. Then turn to step 6.

Step 6 If the evolution generations is less than $T_{max}$, turn to step 4. Otherwise, turn to step 7.

Step 7 Output the results: $p_g$ and $f(p_g)$.

## 6 Numerical simulation

In order to test the performance of the PSO algorithm with chaos perturbation proposed in this paper, this paper selects five benchmark function shown in Table 1.

Table 1: Benchmark functions and part parameters

| Function | Dimensionality | Search space | Optimal value | Convergence criteria |
|---|---|---|---|---|
| Sphere Function | 30 | $[-100, 100]^n$ | 0 | 0.01 |
| Rosenbrock's Function | 30 | $[-30, 30]^n$ | 0 | 0.01 |
| Rastrigin's Function | 30 | $[-5.12, 5.12]^n$ | 0 | 0.01 |
| Griewanks's Function | 30 | $[-600, 600]^n$ | 0 | 0.01 |
| Schwefel Function | 30 | $[-100, 100]^2$ | 0 | 0.01 |

Table 1 gives some parameters of the algorithm, such as dimensionality, search space, theoretical optimum and convergence criteria. The theoretical optimums of these benchmark functions are all zeros. Because the theoretical optimum is difficult to obtained, so this paper uses the convergence criteria to determine the astringency of the algorithm. The convergence criteria equals to 0.01.

Aiming at the above benchmark functions, this paper compares the optimal performance of the following algorithms with each other: the algorithm ACPSO proposed in this paper, the standard PSO algorithm denoted by SPSO, the algorithm in the reference [12] denoted by Ref. 12, the algorithm in the reference [15] denoted by Ref. 15, and the algorithm in the reference [20] denoted by Ref. 20. The algorithm ACPSO proposed in this paper tests the number of two kinds of particles. The SPSO algorithm is shown in the reference [2]. The algorithm Ref. 12 generates the random number of the velocity equation for the SPSO algorithm by using Logistic chaos. The algorithm Ref. 15 uniformly adjusts the inertia weight according to the average velocity of the all particles. The algorithm Ref. 20 is the firefly algorithm. For Ref. 20, it takes the number

of particles 40, the evolution generations 1000, initial attractiveness 0.728, the initial absorption coefficient 0.345 and the randomization parameter 0.25. Using the parameters shown in Table 2, each algorithm is run 20 times respectively.

Table 2: The parameters for each algorithm

| Algorithm | Number of particles | Evolution generations | Initial $\omega$ | $p_2$ | $se$ | Chaos update rate | $r_1$ | $r_2$ |
|---|---|---|---|---|---|---|---|---|
| ACPSO | 40 | 1000 | 0.95 | 1.05 | 1 | 61.8% | / | / |
| ACPSO | 400 | 1000 | 0.95 | 1.05 | 1 | 61.8% | / | / |
| SPSO | 40 | 1000 | 0.7298 | / | / | / | / | / |
| Ref. 12 | 40 | 1000 | 0.7298 | / | / | / | 0.2 | 0.8 |
| Ref. 15 | 40 | 1000 | 3 | 1.05 | / | / | / | / |

The performance of each algorithm is compared by adapting the following criterions. (1) The convergence rate $I_r$. It is the ratio of running time as achieving the convergence criteria to the total running time 20.

(2) Optimal value $f_{best}$. It is the minimum value of the results in 20 times run.

(3) The mean optimal value denoted by $f_{av}$. It is defined by the arithmetic mean value of the optimal fitness in 20 times run.

(4) Mean squared deviation denoted by *std*. It is the mean squared deviation for some algorithm in 20 times run.

(5) Elapsed time. It is the consuming time for each run.

(6) The schematic diagram of mean convergence. Firstly the evolution generations and the corresponding optimal value are extracted. And then take the arithmetic mean value for the optimal value in 20 times run for each evolution generation. Finally, draw the graph by taking the evolution generations and the mean optimal value of each evolution generation to be the abscissa and ordinate respectively.

(7) The schematic diagram of optimal convergence. Aiming at 20 times run, take the running process corresponding the optimal result. And draw the graph by taking the evolution generations and the optimal value of each evolution generation to be the abscissa and ordinate, respectively.

It gives the result respectively for each benchmark function from Table 3 to Table 7 as follow.

Table 3: The result of Sphere Function

| Algorithm | $I_r$ /% | $f_{best}$ | $f_{av}$ | $std$ | Elapsed time /$s$ |
|---|---|---|---|---|---|
| ACPSO | 0 | 0.7204 | 2.68 | 1.11 | 1.66 |
| ACPSO | 1 | $7.83e-13$ | $6.60e-12$ | 4.77 | 15.23 |
| SPSO | 0 | 2164.84 | 6705.48 | 2772.00 | 0.97 |
| Ref. 12 | 0 | 22237.05 | 55486.04 | 14306.06 | 0.97 |
| Ref. 15 | 0 | 41.8138 | 438.52 | 448.86 | 1.14 |
| Ref. 20 | 1 | 0.00174 | 0.00268 | 0.000495 | 5.12 |

The benchmark functions adapted in this paper are all high-dimensional functions of 30-dimension. From Figure 4 to Figure 13, it can be seen that the ACPSO algorithm proposed in this paper has better search capability, especially in the earlier stage of the evolution. From Table 3 to 7, it can be seen that the ACPSO algorithm shows better stability. On the whole,

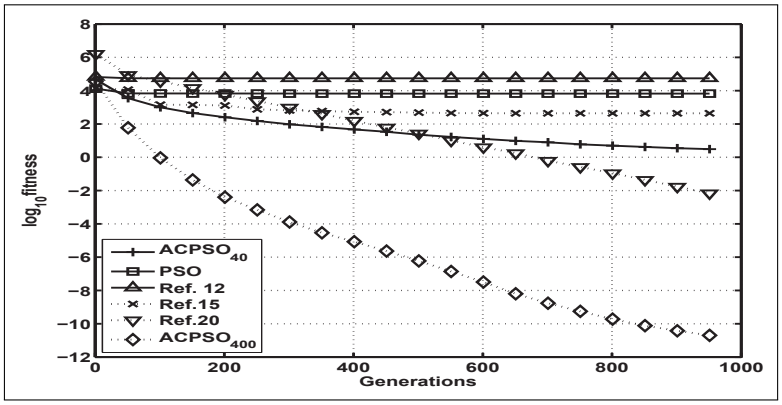


Figure 4: Average evolutionary process for Sphere function

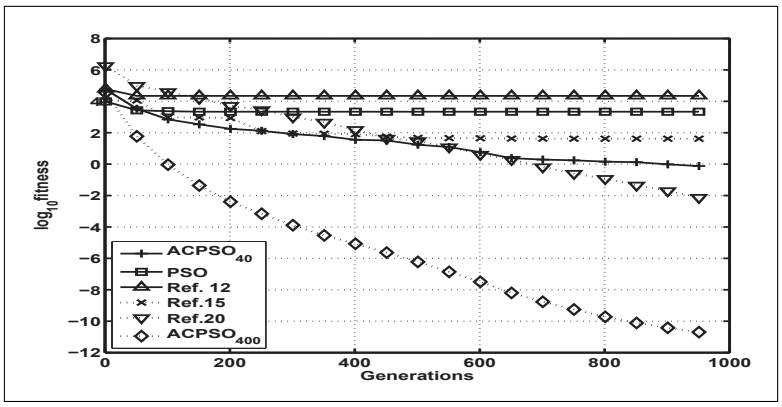


Figure 5: Best evolutionary process for Sphere function

Table 4: The result of Rosenbrock's Function

| Algorithm | $I_r/\%$ | $f_{best}$ | $f_{av}$ | $std$ | Elapsed time $s$ |
|---|---|---|---|---|---|
| ACPSO | 0 | 38.10 | 174.31 | 128.14 | 1.64 |
| ACPSO | 1 | 16.90 | 23.19 | 10.99 | 14.74 |
| SPSO | 0 | $6.65e+5$ | $3.79e+6$ | $3.16e+6$ | 0.90 |
| Ref. 12 | 0 | $1.40e+8$ | $2.26e+8$ | $4.57e+7$ | 1.03 |
| Ref. 15 | 0 | 138.44 | $3.75e+3$ | $4.86e+3$ | 1.05 |
| Ref. 20 | 1 | 28.07 | $1.76e+5$ | $7.00e+5$ | 5.27 |

Table 5: The result of Rastrigin's Function

| Algorithm | $I_r/\%$ | $f_{best}$ | $f_{av}$ | $std$ | Elapsed time $s$ |
|---|---|---|---|---|---|
| ACPSO | 0 | 44.32 | 105.88 | 39.94 | 1.63 |
| ACPSO | 0 | 29.85 | 80.35 | 29.34 | 15.05 |
| SPSO | 0 | 205.15 | 262.46 | 39.06 | 0.93 |
| Ref. 12 | 0 | 304.99 | 392.91 | 41.42 | 0.94 |
| Ref. 15 | 0 | 29.01 | 62.14 | 24.09 | 1.09 |
| Ref. 20 | 0 | 17.39 | 34.47 | 9.39 | 4.96 |

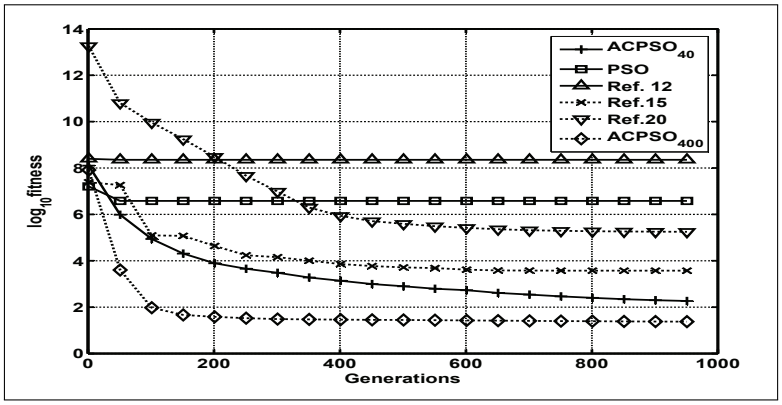


Figure 6: Average evolutionary process for Rosenbrock's function

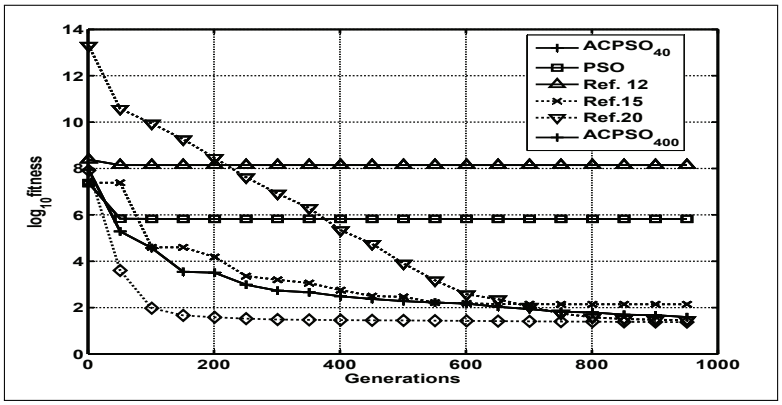


Figure 7: Best evolutionary process for Rosenbrock's function

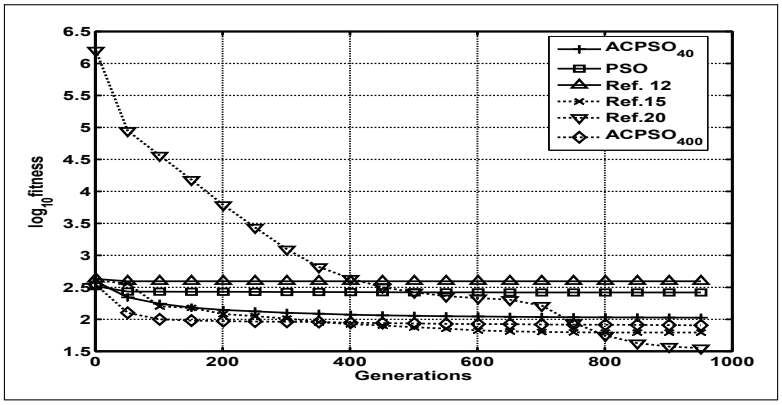


Figure 8: Average evolutionary process for Rastrigin's function

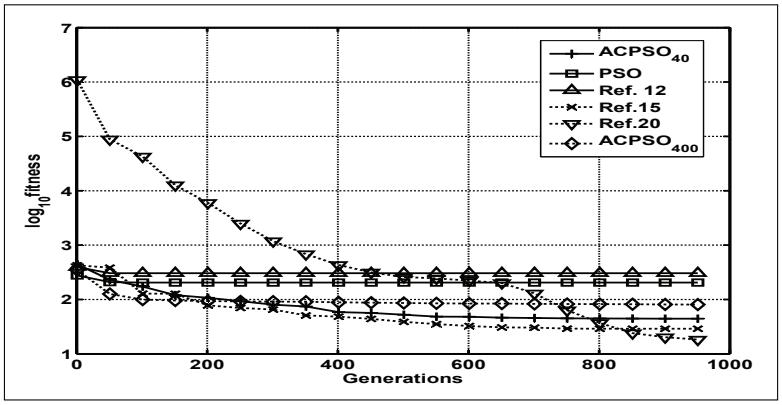


Figure 9: Best evolutionary process for Rastrigin's function

Table 6: The result of Griewanks's Function

| Algorithm | $I_r/\%$ | $f_{best}$ | $f_{av}$ | $std$ | Elapsed time $s$ |
|---|---|---|---|---|---|
| ACPSO | 0 | 0.7952 | 0.9675 | 0.0068 | 1.87 |
| ACPSO | 35 | $5.66e-10$ | 0.0214 | 0.0169 | 19.52 |
| SPSO | 0 | 19.303 | 51.182 | 18.643 | 1.10 |
| Ref. 12 | 0 | 278.85 | 527.87 | 116.75 | 1.11 |
| Ref. 15 | 0 | 1.2985 | 6.2802 | 5.9002 | 1.36 |
| Ref. 20 | 85 | $1.04e-4$ | 0.0041 | 0.0082 | 6.21 |

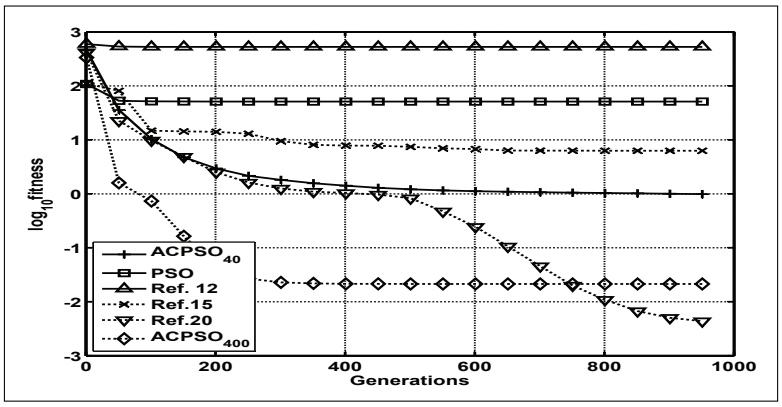


Figure 10: Average evolutionary process for Griewanks's function

Table 7: The result of Griewanks's Function

| Algorithm | $I_r/\%$ | $f_{best}$ | $f_{av}$ | $std$ | Elapsed time $s$ |
|---|---|---|---|---|---|
| ACPSO | 0 | 1.0486 | 2.0469 | 0.6447 | 1.84 |
| ACPSO | 1 | 0.00038 | 0.00038 | $1.91e-08$ | 17.08 |
| SPSO | 0 | 207.8326 | 453.7501 | 130.34 | 1.10 |
| Ref. 12 | 0 | 2014.7359 | 3771.971 | 852.53 | 1.11 |
| Ref. 15 | 0 | 12.2179 | 44.2053 | 32.499 | 1.27 |
| Ref. 20 | 0 | 4621.4715 | 6014.141 | 767.825 | 5.38 |

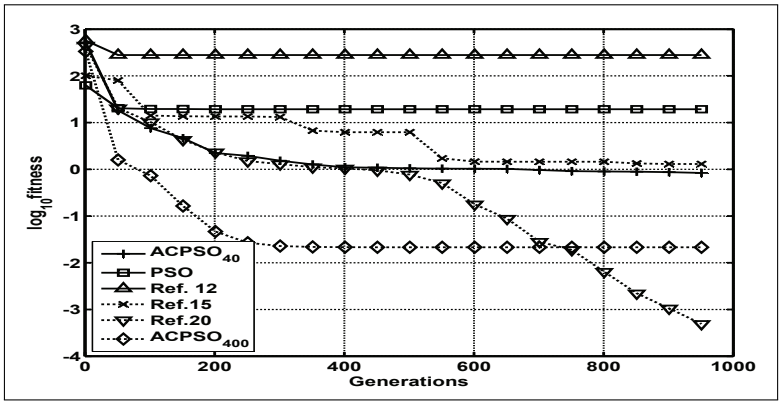


Figure 11: Best evolutionary process for Griewanks's function

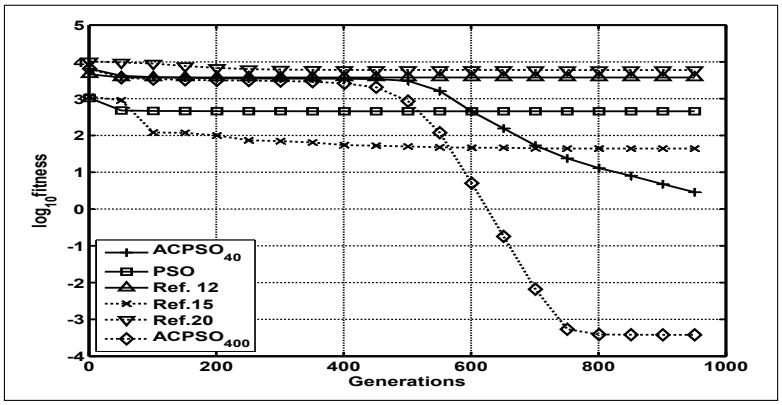


Figure 12: Average evolutionary process for Schwefel function

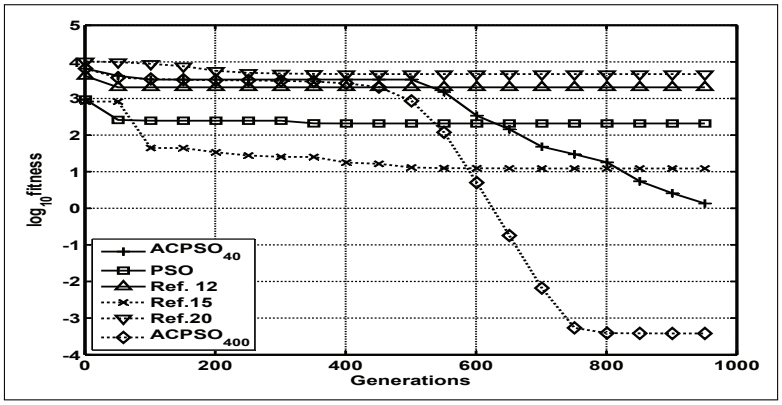


Figure 13: Best evolutionary process for Schwefel function

the search capability of the algorithm proposed in this paper is superior to the standard PSO algorithm and is better than the algorithm Ref. 20 for part benchmark functions.

## Conclusions

(1) The algorithm proposed in this paper introduces the Anderson chaotic mapping to realize the initial of the particle swarm and the perturbation of part particles. It also introduces the concept of the expected mean velocity of particle swarm to realize the adjustment of the inertia weight of particles.

(2) The results of Benchmark functions show that the algorithm proposed in this paper reflects preferable search ability.

(3) The analyses of the mean optimal fitness and evolution generations show that the algorithm of this paper keeps the population diversity well in the middle stage of the evolution and it has stronger local search ability in the later stage of evolution.

### Acknowledgment

The authors will thank people in the Branch of Key Laboratory of CNPC for Oil and Gas Production and Key Laboratory of Exploration Technologies for Oil and Gas Resources for their great help. This paper is supported by Educational Commission of Hubei Province of China (B2015449) and National Natural Science Foundation of China (61572084 and 51504038).

## Bibliography

[1] Kennedy J, Eberhart R,(1995); Particle swarm optimization. *IEEE Int. Conf. on Neural Networks*. Piscataway, NJ. IEEE Service Center, 1942-1948.

[2] Shi Y, Eberhart R C,(1998); A modified particle swarm optimizer. *IEEE Int. Conf. on Evolutionary Computation*. Piscataway, NJ. IEEE Service Center, 69-73.

[3] Naveed Iqbal, Azzedine Zerguine,Naofal Al-Dhahir,(2014); Decision Feedback Equalization using Particle Swarm Optimization. *Signal Processing*, 108:1-12, DOI: 10.1016/j.sigpro.2014.07.030

[4] Manish Mandloi, Vimal Bhatia,(2016); A low-complexity hybrid algorithm based on particle swarm and ant colony optimization for large-MIMO detection. *Signal Processing*, 50:66-74. DOI: 10.1016/j.eswa.2015.12.008

[5] Md Ashiqur Rahmana,Sohel Anwara,Afshin Izadian,(2016); Electrochemical model parameter identification of a lithium-ion battery using particle swarm optimization method.*Journal of Power Sources*.307, 86-97. DOI: 10.1016/j.jpowsour.2015.12.083

[6] Razieh Sheikhpour,Mehdi Agha Sarrama,Robab Sheikhpour,(2016); Particle swarm optimization for bandwidth determination and feature selection of kernel density estimation based classifiers in diagnosis of breast cancer.*Applied Soft Computing*, 40113-131. DOI: 10.1016/j.asoc.2015.10.005

[7] Yu-Shan Cheng,Man-Tsai Chuang,Yi-Hua Liu,Shun-Chung Wang,Zong-Zhen Yang,(2016); A particle swarm optimization based power dispatch algorithm with roulette wheel re-distribution mechanism for equality constraint.*Renewable Energ*, 88: 58-72, DOI: 10.1016/j.renene.2015.11.023.

[8] Migdat Hodzic, Li-Chou Tai (2016); Grey Predictor reference model for assisting particle swarm optimization for wind turbine contro.*Renewable Energ*, 86: 251-256, DOI: 10.1016/j.renene.2015.08.001.

[9] Tao Lin, Peng Wu, Fengmei Gao, Yi Yu, Linhong Wang (2016); Study on SVM temperature compensation of liquid ammonia volumetric flowmeter based on variable weight PSO, *International Journal of Heat and Technology* , 33(2):151-156, DOI: 10.18280/ijht.330224.

[10] Shi Y. and Eberhart, R.C. (2001); Fuzzy adaptive particle swarm optimization, *IEEE Int. Conf. on Evolutionary Computation*.Seoul, Korea, 101-106.

[11] Zhang L., Yu H., Hu S. (2003); A new approach to improve particle swarm optimization, *Genetic and Evolutionary Computation Conference 2003* .Chicago, IL, USA, 134-142.

[12] Jiang C. W., Etorre B. (2005); A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization, *Mathematics and Computers in Simulation*, 68:57-65.

[13] Chen G.M., Huang X.B.,Jia J Y, Min Z.F.(2006); Natural exponential inertia weight strategy in particle swarm optimization. *World Congress on Intelligent Control & Automation*, 1: 3672-3675.

[14] Jiao B., Lian Z.G.,Gu X.S.(2008); A dynamic inertia weight particle swarm optimization algorithm.*Chaos Solitons & Fractals*, 37(3): 698-705.

[15] Zhang Dingxue, Liao Ruiquan,(2009); Adaptive particle swarm optimization algorithm based on population velocity.*Control and Decision*, 24(8): 1257-1265, DOI:10.13195/j.cd.2009.08.139.zhangdx.025

[16] Lv Zhensu, Hou Zhirong (2004); Particle Swarm Optimization with Adaptive Mutation.*Acta Electronica sinica*, 32(3):416-420.

[17] Zeng Jianchao, Cui Zhihua (2012); Nature Inspired Computation, *Beijing, National Defense Industry Press*, 252-256.

[18] Kazem A., Sharifi E., Hussain F.K., Saberi M., Hussain O.K. (2013); Support vector regression with chaos-based firefly algorithm for stock market price forecasting, *Applied Soft Computing*, 13: 947-958. DOI: http://dx.doi.org/10.1016/j.asoc.2012.09.024.

[19] LIU Huaying, LIN Yue (2006); A Hybrid Particle Swarm Optimization Based on Chaos Strategy to Handle Local Convergence, *Computer Engineering and Applications*, 42(13):77-79.

[20] Yang X.S.(2011); Chaos-enhanced firefly algorithm with automatic parameter tuning, *International Journal of Swarm Intelligence Research*, 2: 1-11.

[21] Lu Y., Liu X.(2011); A new population migration algorithm based on the chaos theory, *IEEE 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, 147-150, DOI: 10.1109/IPTC.2011.44

[22] LV Jinhu, LU Junan, Chen Shihua,(2002); Chaotic time series analysis and its application, *Wuhan: Wuhan University press*, 87-89.

[23] R. Anderson (1996); Industrial Cryptography, *IEEE REV.*, 118-120.