5

Individual Parameter Selection Strategy for Particle Swarm Optimization

Xingjuan Cai, Zhihua Cui, Jianchao Zeng and Ying Tan
Division of System Simulation and Computer Application, Taiyuan University of Science and Technology
P.R.China

1. Brief Survey of Particle Swarm Optimization

With the industrial and scientific developments, many new optimization problems are needed to be solved. Several of them are complex multi-modal, high dimensional, non-differential problems. Therefore, some new optimization techniques have been designed, such as genetic algorithm (Holland, 1992), ant colony optimization (Dorigo & Gambardella, 1997), etc. However, due to the large linkage and correlation among different variables, these algorithms are easily trapped to a local optimum and failed to obtain the reasonable solution.

Particle swarm optimization (PSO) (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995) is a population-based, self-adaptive search optimization method motivated by the observation of simplified animal social behaviors such as fish schooling, bird flocking, etc. It is becoming very popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution (Shen et al., 2005; Eberhart & Shi, 1998; Li et al., 2005).

In a PSO system, multiple candidate solutions coexist and collaborate simultaneously. Each solution called a "particle", flies in the problem search space looking for the optimal position to land. A particle, as time passes through its quest, adjusts its position according to its own "experience" as well as the experience of neighboring particles. Tracking and memorizing the best position encountered build particle's experience. For that reason, PSO possesses a memory (i.e. every particle remembers the best position it reached during the past) . PSO system combines local search method (through self experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation.

A particle status on the search space is characterized by two factors: its position and velocity, which are updated by following equations:

$$\vec{v_j}(t+1) = w\vec{v_j}(t) + c_1 r_1(\vec{p_j}(t) - \vec{x_j}(t)) + c_2 r_2(\vec{p_g}(t) - \vec{x_j}(t)) \quad (1)$$

$$\vec{x_j}(t+1) = \vec{x_j}(t) + \vec{v_j}(t+1) \quad (2)$$

where $\vec{v_j}(t)$ and $\vec{x_j}(t)$ represent the velocity and position vectors of particle *j* at time *t*, respectively. $\vec{p_j}(t)$ means the best position vector which particle *j* had been found, as well as $\vec{p_g}(t)$ denotes the corresponding best position found by the whole swarm. Cognitive

coefficient c_1 and social coefficient c_2 are constants known as acceleration coefficients, and r_1 and r_2 are two separately generated uniformly distributed random numbers in the range [0, 1].
To keep the moving stability, a limited coefficient v_{max} is introduced to restrict the size of velocity.

$$|v_{jk}(t+1)| \le v_{max} \tag{3}$$

The first part of (1) represents the previous velocity, which provides the necessary momentum for particles to roam across the search space. The second part, known as the "cognitive" component, represents the personal thinking of each particle. The cognitive component encourages the particles to move toward their own best positions found so far. The third part is known as the "social" component, which represents the collaborative effect of the particles, in finding the global optimal solution. The social component always pulls the particles toward the global best particle found so far.
Since particle swarm optimization is a new swarm intelligent technique, many researchers focus their attentions to this new area. One famous improvement is the introduction of the inertia weight (Shi & Eberhart, 1998a), similarly with temperature schedule in the simulated annealing algorithm. Empirical results showed the linearly decreased setting of inertia weight can give a better performance, such as from 1.4 to 0 (Shi & Eberhart, 1998a), and 0.9 to 0.4 (Shi & Eberhart, 1998b, Shi & Eberhart, 1999). In 1999, Suganthan (Suganthan,1999) proposed a time-varying acceleration coefficients automation strategy in which both c_1 and c_2 are linearly decreased during the course of run. Simulation results show the fixed acceleration coefficients at 2.0 generate better solutions. Following Suganthan's method, Venter (Venter, 2002) found that the small cognitive coefficient and large social coefficient could improve the performance significantly. Further, Ratnaweera (Ratnaweera et al., 2004) investigated a time-varying acceleration coefficients. In this automation strategy, the cognitive coefficient is linearly decreased during the course of run, however, the social coefficient is linearly increased inversely.
Hybrid with Kalman filter, Monson designed a new Kalman filter particle swarm optimization algorithm (Monson & Seppi, 2004) . Similarly, Sun proposed a new quantum particle swarm optimization (Sun et al., 2004) in 2004. From the convergence point, Cui designed a global convergence algorithm — stochastic particle swarm optimization (Cui & Zeng, 2004). There are still many other modified methods, such as fast PSO (Cui et al., 2006a), predicted PSO (Cui et al.,2006b), etc. The details of these algorithms can be found in corresponding references.
The PSO algorithm has been empirically shown to perform well on many optimization problems. However, it may easily get trapped in a local optimum for high dimensional multi-modal problems. With respect to the PSO model, several papers have been written on the subject to deal with premature convergence, such as the addition of a queen particle (Mendes et al., 2004), the alternation of the neighborhood topology (Kennedy, 1999), the introduction of subpopulation and giving the particles a physical extension (Lovbjerg et al., 2001), etc. In this paper, an individual parameter selection strategy is designed to improve the performance when solving high dimensional multi-modal problems.
The rest of this chapter is organized as follows: the section 2 analyzes the disadvantages of the standard particle swarm optimization parameter selection strategies; the individual inertia weight selection strategy is designed in section 3; whereas section 4 provides the cognitive parameter selection strategy. In section 5, the individual social parameter selection strategies is designed. Finally, conclusion and future research are discussed.

2. The Disadvantages of Standard Particle Swarm Optimization

Partly due to the differences among individuals, swarm collective behaviors are complex processes. Fig.1 and Fig.2 provide an insight of the special swarm behaviors about birds flocking and fish schooling. For a group of birds or fish families, there exist many differences. Firstly, in nature, there are many internal differences among birds (or fish), such as ages, catching skills, flying experiences, and muscles' stretching, etc. Furthermore, the lying positions also provide an important influence on individuals. For example, individuals, lying in the side of the swarm, can make several choices differing from center others. Both of these differences mentioned above provide a marked contribution to the swarm complex behaviors.

Figure 1. Fish's Swimming Process

Figure 2. Birds' Flying Process

For standard particle swarm optimization, each particle maintains the same flying (or swimming) rules according to (1), (2) and (3). At each iteration, the inertia weight w, cognitive learning factor c_1 and social learning factor c_2 are the same values within the whole swarm, thus the differences among particles are omitted. Since the complex swarm behaviors can emerge the adaptation, a more precise model, incorporated with the differences, can provide a deeper insight of swarm intelligence, and the corresponding algorithm may be more effective and efficient. Inspired with this method, we propose a new algorithm in which each particle maintains personal controlled parameter selection setting.

3. Individual Inertia weight Selection Strategy

Without loss of generality, this paper consider the following problem:

$$min\ f(\vec{X}) \qquad \vec{X} \in D \subseteq R^n \tag{4}$$

From the above analysis, the new variant of PSO in this section will incorporate the personal differences into inertia weight of each particle (called PSO-IIWSS, in briefly) (Cai et al., 2008), providing a more precise model simulating the swarm behaviors. However, as a new modified PSO, PSO-IIWSS should consider two problems listed as follows:

1. How to define the characteristic differences of each particle?
2. How to use the characteristic difference to control inertia weight, so as to affect its behaviors?

3.1 How to define the characteristic differences?

If the fitness value of particle u is better than which of particle m, the probability that global optima falls into u's neighborhood is larger than that of particle m. In this manner, the particle u should pay more attentions to exploit its neighborhood. On the contrary, it may tend to explore other region with a larger probability than exploitation. Thus the information index is defined as follows:

The information index - score of particle u at time t is defined as

$$Score_u(t) = \frac{f(x_{worst}(t)) - f(x_u(t))}{f(x_{worst}(t)) - f(x_{best}(t))} \tag{5}$$

where $x_{worst}(t)$ and $x_{best}(t)$ are the worst and best particles' position vectors at time t, respectively.

3.2 How to use the characteristic differences to guild its behaviors?

Since the coefficients setting can control the particles' behaviors, the differences may be incorporated into the controlled coefficients setting to guide each particle's behavior. The allowed controlled coefficients contain inertia weight w, two accelerators c_1 and c_2. In this section, inertia weight w is selected as a controlled parameter to reflect the personal characters. Since w is dependent with each particle, we use w_u (t) representing the inertia weight of particle u at time t.

Now, let us consider the adaptive adjustment strategy of inertia weight $w_u(t)$. The following part illustrates three different adaptive adjustment strategies.

Inspired by the ranking selection mechanism of genetic algorithm (Mich ale wicz, 1992), the first adaptive adjustment of inertia weight is provided as follows:

The inertia weight $w_u(t)$ of particle u at time t is computed by

$$w_u(t) = w_{low}(t) + (w_{high}(t) - w_{low}(t)) \times (1 - Score_j(t)) \tag{6}$$

where $w_{low}(t)$ and $w_{high}(t)$ are the lower and upper bounds of the swarm at time t.
This adaptive adjustment strategy states the better particles should tend to exploit its neighbors, as well as the worse particles prefer to explore other region. This strategy implies the determination of inertia weight of each particle, may provide a large selection pressure.
Compared with ranking selection, fitness uniform selection scheme (FUSS) is a new selection strategy measuring the diversity in phenotype space. FUSS works by focusing the selection intensity on individuals which have uncommon fitness values rather than on those with highest fitness as is usually done, and the more details can be found in (Marcus, 2002). Inspired by FUSS, the adaptive adjustment strategy two aims to provide a more chance to balance exploration and exploitation capabilities.
The inertia weight $w_u(t)$ of particle u at time t is computed by

$$w_u(t) = w_{low}(t) + (w_{high}(t) - w_{low}(t)) \times (1 - Score_{rand}(t)) \tag{7}$$

where $w_{low}(t)$ and $w_{high}(t)$ are the lower and upper bounds of the swarm at time t. $Score_{rand}(t)$ is defined as follows.

$$Score_{rand}(t) = \{Score_s(t) \| r - f(x_s(t))| \leq |r - f(x_i(t))|, i = 1, ..., s-1, s+1, ..., n\} \tag{8}$$

where r is a random number sampling uniformly between $f(x_{best}(t))$ and $f(x_{worst}(t))$.
Different from ranking selection and FUSS strategies which need to order the whole swarm, tournament strategy (Blickle & Thiele, 1995) is another type of selection strategy, it only uses several particles to determine one particle's selection probability. Analogized with tournament strategy, the adaptive adjustment strategy three is designed with local competition, and defined as follows:
The inertia weight $w_u(t)$ of particle u at time t is computed by

$$w_u(t) = \begin{cases} w_{low}(t) + (w_{high}(t) - w_{low}(t)) \times (1 - Score_{r_1}(t)), & \text{if } f(x_{r_1}) < f(x_{r_2}) \\ w_{low}(t) + (w_{high}(t) - w_{low}(t)) \times (1 - Score_{r_2}(t)), & \text{otherwise} \end{cases}, \tag{9}$$

where $w_{low}(t)$ and $w_{high}(t)$ are the lower and upper bounds of the swarm at time t. $x_{r_1}(t)$ and $x_{r_1}(t)$ are two random selected particles uniformly.

3.3 The Step of PSO-IIWSS

The step of PSO-IIWSS is listed as follows.

- Step 1. Initializing each coordinate $x_{jk}(0)$ to a value drawn from the uniform random distribution on the interval $[x_{min}, x_{max}]$, for $j = 1,2, ...,s$ and $k = 1,2, ...,n$. This distributes the initial position of the particles throughout the search space. Where s is the value of the swarm, n is the value of dimension. Initializing each $v_{jk}(0)$ to a value drawn from the uniform random distribution on the interval $[-v_{max}, v_{max}]$, for all j and k. This distributes the initial velocity of the particles.
- Step 2. Computing the fitness of each particle.
- Step 3. Updating the personal historical best positions for each particle and the swarm;

- Step 4. Determining the best and worst particles at time *t*, then, calculate the score of each particle at time *t*.
- Step 5. Computing the inertia weight value of each particle according to corresponding adaptive adjustment strategy one,two and three (section 3.2, respectively) .
- Step 6. Updating the velocity and position vectors with equation (1),(2) and (3) in which the inertia w is changed with $w_j(t)$.
- Step 7. If the stop criteria is satisfied, output the best solution; otherwise, go step 2.

3.4 Simulation Results

3.4.1 Selected Benchmark Functions

In order to certify the efficiency of the PSO-IIWSS, we select five famous benchmark functions to testify the performance, and compare PSO-IIWSS with standard PSO (SPSO) and Modified PSO with time- varying accelerator coefficients (MPSO_TVAC) (Ratnaweera et al, 2004). Combined with different adaptive adjustment strategy of inertia weight one, two and three, the corresponding versions of PSO-IIWSS are called PSO-IIWSS1, PSO-IIWSS2, PSO-IIWSS3, respectively.

Sphere Modal:

$$f_1(x) = \sum_{j=1}^{n} x_j^2$$

where $|x_j| \leq 100.0$, and

$$f_1(x^*) = f_1(0,0,...,0) = 0.0$$

Schwefel Problem 2.22:

$$f_2(x) = \sum_{j=1}^{n} |x_j| + \prod_{k=1}^{n} |x_k|$$

where $|x_j| \leq 10.0$, and

$$f_2(x^*) = f_2(0,0,...,0) = 0.0$$

Schwefel Problem 2.26:

$$f_3(x) = -\sum_{j=1}^{n} (x_j \sin(\sqrt{|x_j|}))$$

where $|x_j| \leq 500.0$, and

$$f_3(x^*) = f_3(420.9687, 420.9687, ..., 420.9687) \approx -12569.5$$

Ackley Function:

$$f_4(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}) - exp(\frac{1}{n}\sum_{k=1}^{n} \cos 2\pi x_k) + 20 + e$$

where $|x_j| \leq 32.0$, and

$$f_4(x^*) = f_4(0, 0, ..., 0) = 0.0$$

Hartman Family:

$$f_5(x) = -\sum_{i=1}^{4} c_i exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2]$$

where $x_j \in$ [0.0,1.0], and a_{ij} is satisfied with the following matrix.

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

p_{ij} is satisfied with the following matrix.

$$\begin{pmatrix} 0.3687 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

c_i is satisfied with the following matrix.

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

$$f_5(x^*) = f_5(0.114, 0.556, 0.852) = -3.86$$

Sphere Model and Schwefel Problem 2.22 are unimodel functions. Schwefel Problem 2.26 and Ackley function are multi-model functions with many local minima,as well as Hartman Family with only several local minima.

3.4.2 Parameter Setting

The coefficients of SPSO,MPSO_TVAC and PSO-IIWSS are set as follows:
The inertia weight w is decreased linearly from 0.9 to 0.4 with SPSO and MPSO_TVAC, while the inertia weight lower bounds of PSO-IIWSS is set 0.4, and the upper bound of PSO-IIWSS is set linearly from 0.9 to 0.4. Two accelerator coefficients c_1 and c_2 are both set to 2.0 with SPSO and PSO-IIWSS, as well as in MPSO_TVAC, c_1 decreases from 2.5 to 0.5,while c_2

increases from 0.5 to 2.5. Total individuals are 100 except Hartman Family with 20, and v_{max} is set to the upper bound of domain.The dimensions of Sphere Model, Schwefel Problem 2.22,2.26 and Ackley Function are set to 30,while Hartman Family's is 3.Each experiment the simulation runs 30 times while each time the largest evolutionary generation is 1000 for Sphere Model, Schwefel Problem 2.22, Schwefel Problem 2.26, and Ackley Function, and due to small dimensionality, Hartman Family is set to 100.

3.4.3 Performance Analysis

Table 1 to 5 are the comparison results of five benchmark functions under the same evolution generations respectively.The average mean value and average standard deviation of each algorithm are computed with 30 runs and listed as follows.
From the Tables, PSO-IIWSSI maintains a better performance than SPSO and MPSO_TVAC with the average mean value. For unimodel functions, PSO-IIWSS3 shows preferable convergence capability than PSO-IIWSS2,while vice versa for the multi-model functions.
From Figure 1 and 2,PSO-IIWSSI and PSO-IIWSS3 can find the global optima with nearly a line track, while PSO-IIWSSI owns the fast search capability during the whole course of simulation for figure 3 and 4. PSO-IIWSS2 shows the better search performance with the increase of generations. In one word, PSO-IIWSSI owns a better performance within the convergence speed for all functions nearly.

Algorithm	Average Mean Value	Average Standard Deviation
SPSO	9.9512e-006	1.4809e-005
MPSO_TVAC	4.5945e-018	1.9379e-017
PSO-IIWSSI	1.4251e-023	1.8342e-023
PSO-IIWSS2	1.2429e-012	2.8122e-012
PSO-IIWSS3	1.3374e-019	6.0570e-019

Table 1. Simulation Results of Sphere Model

Algorithm	Average Mean Value	Average Standard Deviation
SPSO	7.7829e-005	7.5821e-005
MPSO_TVAC	3.0710e-007	1.0386e-006
PSO-IIWSSI	2.4668e-015	2.0972e-015
PSO-IIWSS2	1.9800e-009	1.5506e-009
PSO-IIWSS3	3.2359e-012	4.1253e-012

Table 2. Simulation Results of Schwefel Problem 2.22

Algorithm	Average Mean Value	Average Standard Deviation
SPSO	-6.2474e+003	9.2131e+002
MPSO_TVAC	-6.6502e+003	6.0927e+002
PSO-IIWSSI	-7.7455e+003	8.0910e+002
PSO-IIWSS2	-6.3898e+003	9.2699e+002
PSO-IIWSS3	-6.1469e+003	9.1679e+002

Table 3. Simulation Results of Schwefel Problem 2.26

Algorithm	Average Mean Value	Average Standard Deviation
SPSO	8.8178e-004	6.8799e-004
MPSO_TVAC	1.8651e-005	1.0176e-004
PSO-IIWSS1	2.9940e-011	4.7552e-011
PSO-IIWSS2	3.8672e-007	5.6462e-007
PSO-IIWSS3	3.3699e-007	5.8155e-007

Table 4. Simulation Results of Ackley Function

Algorithm	Average Mean Value	Average Standard Deviation
SPSO	-3.7507e+000	1.0095e-001
MPSO_TVAC	-3.8437e+000	2.9505e-002
PSO-IIWSS1	-3.8562e+000	1.0311e-002
PSO-IIWSS2	-3.8511e+000	1.6755e-002
PSO-IIWSS3	-3.8130e+000	5.2168e-002

Table 5. Simulation Results of Hartman Family

3.5 Individual non-linear inertia weight selection strategy (Cui et al., 2008)

3.5.1 PSO-IIWSS with Different Score Strategies (PSO-INLIWSS)

As mentioned above, the linearly decreased score strategy can not reflect the truly complicated search process of PSO. To make a deep insight of action for score, three non-linear score strategies are designed in this paper. These three strategies are unified to a power function, which is set to the following equation:

$$Score_u(t) = \{1 - [\frac{f(x_{worst}(t)) - f(x_u(t))}{f(x_{worst}(t)) - f(x_{best}(t))}]^{k_1}\}^{k_2} \tag{10}$$

where k_1 and k_2 are two integer numbers.

Figure 3 shows the trace of linear and three non-linear score strategies, respectively. In Figure 1, the value $f(x_{best}(t))$ is set 1, as well as $f(x_{worst}(t))$ is 100. When k_1 and k_2 are both set to 1, it is just the score strategy proposed in [?], which is also called strategy one in this paper. While $k_1 > 1$ and $k_2 = 1$, this non-linear score strategy is called strategy two here. And strategy three corresponds to $k_1 = 1$ and $k_2 > 1$, strategy four corresponds to $k_1 > k_2 > 1$. Description of three non-linear score strategies are listed as follows: Strategy two: the curve $k_1 = 2$ and $k_2 = 1$ in Figure 3 is an example of strategy two. It can be seen this strategy has a lower score value than strategy one. However, the increased ratio of score is not a constant value. For those particles with small fitness values, the corresponding score values are smaller than strategy one, and they pay more attention to exploit the region near the current position. However, the particles tends to make a local search is larger than strategy one due to the lower score values. Therefore, strategy two enhances the local search capability.

Strategy three: the curve $k_1 = 1$ and $k_2 = 2$ in Figure 3 is an example of strategy three. As we can see, it is a reversed curve compared with strategy two. Therefore, it enhances the global search capability.

Strategy four: the curve $k_1 = 2$ and $k_2 = 5$ in Figure 3 is an example of strategy four. The first part of this strategy is similar with strategy two, as well as the later part is similar with strategy three. Therefore, it augments both the local and global search capabilities.

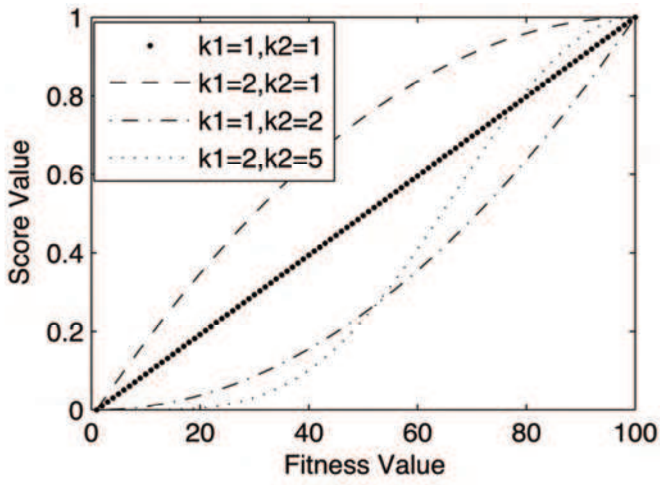


Figure 3. Illustration of Score Strategies

The step of PSO-INLIWSS with different score strategies are listed as follows.

- Step l. Initializing the position and velocity vectors of the swarm, and de termining the historical best positions of each particle and its neighbors;
- Step 2. Determining the best and worst particles at time *t* with the following definitions.

$$x_{best}(t) = \arg\min\{f(x_j(t)), j = 1, 2, ..., s\} \tag{11}$$

and

$$x_{worst}(t) = \arg\max\{f(x_j(t)), j = 1, 2, ..., s\} \tag{12}$$

- Step 3. Calculate the score of each particle at time *t* with formula (10) using different strategies.
- Step 4. Calculating the PSO-INLIWSS inertia weight according to formula (6);
- Step 5. Updating the velocity and position vectors according to formula (1), (2) and (3);
- Step 6. Determining the current personal memory (historical best position);
- Step 7. Determining the historical best position of the swarm;
- Step 8. If the stop criteria is satisfied, output the best solution; otherwise, go step 2.

3.5.2 Simulation Results

To certify the efficiency of the proposed non-linear score strategy, we select five famous benchmark functions to test the performance, and compared with standard PSO (SPSO), modified PSO with time- varying accelerator coefficients (MPSO-TVAC) (Ratnaweera et al., 2004), and comprehensive learning particle swarm optimization (CLPSO) (Liang et al., 2006). Since we adopt four different score strategies, the proposed methods are called PSO-INLIWSS1 (with strategy one, in other words, the original linearly PSO-IIWSS1), PSO-INLIWSS2 (with strategy two), PSO-INLIWSS3 (with strategy three) and PSO-INLIWSS4 (with strategy four), respectively. The details of the experimental environment and results are explained as follows.

In this paper, five typical unconstraint numerical benchmark functions are used to test. They are: Rosenbrock, Schwefel Problem 2.26, Ackley and two Penalized functions.
Rosenbrock Function:

$$f_1(x) = \sum_{j=1}^{n-1}[100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2]$$

where $|x_j| \leq 30.0$, and

$$f_1(x^*) = f_1(1, 1, ..., 1) = 0.0$$

Schwefel Problem 2.26:

$$f_2(x) = -\sum_{j=1}^{n}(x_j \sin(\sqrt{|x_j|}))$$

where $|x_j| \leq 500.0$, and

$$f_2(x^*) = f_2(420.9687, 420.9687, ..., 420.9687) \approx -12569.5$$

Ackley Function:

$$f_3(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}) - exp(\frac{1}{n}\sum_{k=1}^{n}\cos 2\pi x_k) + 20 + e$$

where $|x_j| \leq 32.0$, and

$$f_3(x^*) = f_3(0, 0, ..., 0) = 0.0$$

Penalized Function l:

$$f_4(x) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] \\ +(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$$

where $|x_j| \leq 50.0$, and

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

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

$$f_4(x^*) = f_4(1, 1, ..., 1) = 0.0$$

Penalized Function 2:

$$f_5(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]$$
$$+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$$

where $|x_j| \leq 50.0$, and

$$f_5(x^*) = f_5(1, 1, ..., 1) = 0.0$$

Generally, Rosenbrock is viewed as a unimodal function, however, in recent literatures, several numerical experiments (Shang & Qiu, 2006) have been made to show Rosenbrock is a multi-modal function with only two local optima when dimensionality between 4 to 30. Schwefel problem 2.26, Ackley, and two penalized functions are multi-model functions with many local minima.
The coefficients of SPSO, MPSO-TVAC, and PSO-INLIWSS are set as follows: inertia weight w is decreased linearly from 0.9 to 0.4 with SPSO and MPSO-TVAC, while the inertia weight lower bounds of all version of PSO-INLIWSS are both set to 0.4, and the upper bounds of PSO-INLIWSS are both set linearly decreased from 0.9 to 0.4. Two accelerator coefficients c_1 and c_2 are set to 2.0 with SPSO and PSO-INLIWSS, as well as in MPSO-TVAC, c_1 decreases from 2.5 to 0.5, while c_2 increases from 0.5 to 2.5. Total individuals are 100, and the velocity threshold v_{max} is set to the upper bound of the domain. The dimensionality is 30. In each experiment, the simulation run 30 times, while each time the largest iteration is 50 x *dimension*.

Algorithm	Mean Value	Std Value
SPSO	5.6170e+001	4.3584e+001
MPSO-TVAC	3.3589e+001	4.1940e+001
CLPSO	5.1948e+001	2.7775e+001
PSO-INLIWSS1	2.3597e+001	2.3238e+001
PSO-INLIWSS2	3.4147e+001	2.9811e+001
PSO-INLIWSS3	4.0342e+001	3.2390e+001
PSO-INLIWSS4	3.1455e+001	2.4259e+001

Table 6. The Comparison Results for Rosenbrock

Algorithm	Mean Value	Std Value
SPSO	-6.2762e+003	1.1354e+003
MPSO-TVAC	-6.7672e+003	5.7050e+002
CLPSO	-1.0843e+004	3.6105e+002
PSO-INLIWSS1	-7.7885e+003	1.1526e+003
PSO-INLIWSS2	-7.2919e+003	1.1476e+003
PSO-INLIWSS3	-9.0079e+003	7.1024e+002
PSO-INLIWSS4	-9.0064e+003	9.6881e+002

Table 7. The Comparison Results for Schwefel Problem 2.26

For Rosenbrock (see Table 6), because there is an additional local optimum near (-1, 0, 0,...,0), the performance of the MPSO-TVAC, PSO-INLIWSS1 and PSO-INLIWSS4 are better than others. We also perform several other unimodel and multi-modal functions with only few local optima, the PSO-INLIWSS1 are always the best one within these seven algorithms.
For Schwefel problem 2.26 (Table 7) and Ackley (Table 8), the performance of PSO-INLIWSS3 and PDPSO4 are nearly the same. Both of them are better than others.
However, for two penalized functions (Table 9 and 10), the performance of PSO-INLIWSS3 is not the same as the previous two, although PSO-INLIWSS4 is still stable and better than others. As we known, both of these two penalized functions has strong linkage among dimensions. This implies PSO-INLIWSS4 is more suit for multi-modal problems.
Based on the above analysis, we can draw the following two conclusions:
(l) PSO-INLIWSSl (the original version of PSO-IIWSS1) is suit for unimodel and multi-modal functions with a few local optima;
(2) PSO-INLIWSS4 is the most stable and effective among three score strategies. It is fit for multi-modal functions with many local optima especially for linkages among dimensions;

Algorithm	Mean Value	Std Value
SPSO	5.8161e-006	4.6415e-006
MPSO-TVAC	7.5381e-007	3.3711e-006
CLPSO	5.6159e-006	4.9649e-006
PSO-INLIWSS1	4.2810e-014	4.3890e-014
PSO-INLIWSS2	1.1696e-011	1.2619e-011
PSO-INLIWSS3	2.2559e-014	8.7745e-015
PSO-INLIWSS4	2.1493e-014	7.8195e-015

Table 8. The Comparison Results for Ackley

Algorithm	Mean Value	Std Value
SPSO	6.7461e-002	2.3159e-001
MPSO-TVAC	1.8891e-017	6.9756e-017
CLPSO	1.0418e-002	3.1898e-002
PSO-INLIWSS1	1.6477e-025	4.7735e-025
PSO-INLIWSS2	6.2234e-026	1.6641e-025
PSO-INLIWSS3	2.4194e-024	7.6487e-024
PSO-INLIWSS4	2.2684e-027	4.4964e-027

Table 9. The Comparison Results for Penalized Functionl

Algorithm	Mean Value	Std Value
SPSO	5.4943e-004	2.4568e-003
MPSO-TVAC	9.3610e-027	4.1753e-026
CLPSO	1.1098e-007	2.6748e-007
PSO-INLIWSS1	4.8692e-027	1.3533e-026
PSO-INLIWSS2	2.8092e-028	5.6078e-028
PSO-INLIWSS3	9.0765e-027	2.5940e-026
PSO-INLIWSS4	8.2794e-028	1.6562e-027

Table 10. The Comparison Results for Penalized Function2

4. Individual Cognitive Selection Strategy

Because each particle maintains two types of performance at time *t:* the fitness value $f(\overrightarrow{p}_j(t))$ of historical best position found by particle *j* and that of current position $f(\overrightarrow{x}_j(t))$, respectively. Similarly, two different rewards of environment are also designed associated with $f(\overrightarrow{p}_j(t))$ and $f(\overrightarrow{x}_j(t))$. For convenience, the reward based upon $f(\overrightarrow{p}_j(t))$ is called the self-learning strategy one, and the other one is called the self-learning strategy two. The details of these two strategies are explained as follows.

4.1 Self-learning Strategy One

Let us suppose $\overrightarrow{P}(t) = (\overrightarrow{p}_1(t), \overrightarrow{p}_2(t), ..., \overrightarrow{p}_n(t))$ is the historical best position vector of the swarm at time *t,* where *n* and $\overrightarrow{p_j}(t)$ denote the dimensionality and the historical best position found by particle *j* until time *t.*
The expectation limitation position of particle *j* of standard version of PSO is

$$E\{lim_{t\to\infty}\overrightarrow{x}_j(t)\} = \frac{c_1\overrightarrow{p}_j + c_2\overrightarrow{p}_g}{c_1 + c_2} \tag{13}$$

if c_1 and $\overrightarrow{p_j}(t)$ are constant values. Thus, a large c_1 makes the $E\{lim_{t\to\infty}\overrightarrow{x}_j(t)\}$ moving towards $\overrightarrow{p}_j$, and exploits near $\overrightarrow{p}_j$ with more probability, and vice versa. Combined the better $\overrightarrow{p_j}(t)$ implies the more capability of which global optima falls into, the cognitive coefficient is set as follows.

$$c_{1,j}(t) = c_{low} + (c_{high} - c_{low}) \times Reward1_j(t) \tag{14}$$

where c_{low}, and c_{high} are two predefined lower and upper bounds to control this coefficient. $Reward1_j(t)$ is defined

$$Reward1_j(t) = \begin{cases} 1 \quad , & \text{if } f_{worst} = f_{best}, \\ \frac{f_{worst} - f(\overrightarrow{p}_j(t))}{f_{worst} - f_{best}}, & \text{otherwise} . \end{cases} \tag{15}$$

where f_{worst} and f_{best} denote the worst and best values among $f(\overrightarrow{P}(t))$.

4.2 Self-learning Strategy Two

Let us suppose $\overrightarrow{X}(t) = (\overrightarrow{x}_1(t), \overrightarrow{x}_2(t), ..., \overrightarrow{x}_n(t))$ is the population at time *t*, where *n,* $\overrightarrow{x_j}(t)$ denote the dimensionality and the position of particle *j* at time *t*.
Different from strategy one, if the performance $\overrightarrow{x_j}(t)$ is better than $\overrightarrow{x}_k(t)$ (*j* and *k* are arbitrary chosen from the population), the probability of global optimal fallen near $\overrightarrow{x_j}(t)$ is larger than $\overrightarrow{x}_k(t)$, thus, particle *j* should exploit near its current position with a larger probability than particle *k.* It means $c_{1,j}(t)$ should be less than $c_{1,k}(t)$ to provide little affection of historical best position $\overrightarrow{p_j}(t)$, and the adjustment is defined as follows

$$c_{1,j}(t) = c_{low} + (c_{high} - c_{low}) \times Reward2_j(t) \tag{16}$$

where $Reward2_j(t)$ is defined as

$$Reward2_j(t) = \begin{cases} 0 \quad , & \text{if } f_{worst} = f_{best}, \\ \frac{f(\overrightarrow{x}_j(t)) - f_{best}}{f_{worst} - f_{best}}, & \text{otherwise.} \end{cases} \tag{17}$$

where f_{worst} and f_{best} denote the worst and best values among $f(\overrightarrow{X}(t))$.

4.3 Mutation Strategy

To avoid premature convergence, a mutation strategy is introduced to enhance the ability escaping from the local optima.

This mutation strategy is designed as follows. At each time, particle j is uniformly random selected within the whole swarm, as well as the dimensionality k is also uniformly random selected, then, the $v_{jk}(t)$ is changed as follows.

$$v_{jk}(t) = \begin{cases} 0.5 \times x_{max} \times r_1 \quad , & \text{if } r_2 < 0.5, \\ -0.5 \times x_{max} \times r_1 & \text{otherwise.} \end{cases} \tag{18}$$

where r_1 and r_2 are two random numbers generated with uniform distribution within 0 and 1.

4.4 The Steps of PSO-ILCSS

For convenience, we call the individual Linear Cognitive Selection Strategy(Cai X.J. et al.,2007;Cai X.J. et al.,2008) as ILCSS, and the corresponding variant is called PSO-ILCSS.

The detailed steps of PSO-ILCSS are listed as follows.

- Step l. Initializing each coordinate x_{jk} (0) and v_{jk} (0) sampling within [x_{min}, x_{max}], and [0, v_{max}], respectively, determining the historical best position by each particle and the swarm.
- Step 2. Computing the fitness of each particle.
- Step 3. For each dimension k of particle j, the personal historical best position p_{jk} (t) is updated as follows.

$$p_{jk}(t) = \begin{cases} x_{jk}(t), & \text{if } f(x_j(t)) < f(p_j(t-1)) \ , \\ p_{jk}(t-1) \quad , & \text{otherwise.} \end{cases} \tag{19}$$

- Step 4. For each dimension k of particle j, the global best position p_{gk} (t) is updated as follows.

$$p_{gk}(t) = \begin{cases} p_{jk}(t), & \text{if } f(p_j(t)) < f(p_g(t-1)) \ , \\ p_{gk}(t-1) \quad , & \text{otherwise.} \end{cases} \tag{20}$$

- Step 5. Selecting the self-learning strategy:if strategy one is selected, computing the cognitive coefficient $c_{1,j}$ (t) of each particle according to formula (14) and (15); otherwise, computing cognitive coefficient $c_{1,j}$ (t) with formula (16) and (17).
- Step 6. Updating the velocity and position vectors with equations (l)-(3).
- Step 7. Making mutation operator described in section 4.3.
- Step 8. If the criteria is satisfied, output the best solution; otherwise, goto step 2.

4.5 Simulation Results

Five famous benchmark functions are used to test the proposed algorithm's efficiency. They are Schwefel Problem 2.22,2.26, Ackley, and two different Penalized Functions, the global optima is 0 except Schwefel Problem 2.26 is -12569.5, while Schwefel Problem 2.22 is unimodel function. Schwefel Problem 2.26,Ackley function and two Penalized Functions are multi-model functions with many local minima.

In order to certify the efficiency, four different versions are used to compare: PSO-ILCSS with self-learning strategy one (PSO-ILCSSI), PSO-ILCSS with self-learning strategy two (PSO-ILCSS2),standard PSO (SPSO) and Modified PSO with time-varying accelerator coefficients (MPSO-TVAC) (Ratnaweera et al., 2004).

The coefficients of SPSO,MPSO-TVAC,PSO-ILCSS1 and PSO-ILCSS2 are set as follows: the inertia weight w is decreased linearly from 0.9 to 0.4. Two accelerator coefficients c_1 and c_2 are both set to 2.0 with SPSO, and in MPSO-TVAC, ci decreased from 2.5 to 0.5,while c_2 increased from 0.5 to 2.5. In PSO-ILCSSI and PSO-ILCSS2, the lower bounds c_{low} of c_1 set to 1.0, and the upper bound c_{high} set to linearly decreased from 2.0 to 1.0, while c_2 is set to 2.0. Total individual is 100, and the dimensionality is 30, and v_{max} is set to the upper bound of domain. In each experiment,the simulation run 30 times, while each time the largest evolutionary generation is 1000.

Table 2 is the comparison results of five benchmark functions under the same evolution generations. The average mean value and average standard deviation of each algorithm are computed with 30 runs and listed as follows.

Function	Algorithm	Average Mean Value	Average Standard Deviation
F1	SPSO	6.6044e-005	4.7092e-005
	MPSO-TVAC	3.0710e-007	1.0386e-006
	PSO-ILCSS1	2.1542e-007	3.2436e-007
	PSO-ILCSS2	9.0189e-008	1.3398e-007
F2	SPSO	-6.2474e+003	9.2131e+002
	MPSO-TVAC	-6.6502e+003	6.0927e+002
	PSO-ILCSS1	-8.1386e+003	6.2219e+002
	PSO-ILCSS2	-8.0653e+003	7.2042e+002
F3	SPSO	1.9864e-003	6.0721e-003
	MPSO-TVAC	1.8651e-005	1.0176e-004
	PSO-ILCSS1	3.8530e-008	3.8205e-008
	PSO-ILCSS2	1.3833e-008	1.0414e-008
F4	SPSO	4.3043e-002	6.6204e-002
	MPSO-TVAC	1.7278e-002	3.9295e-002
	PSO-ILCSS1	9.1694e-012	3.4561e-011
	PSO-ILCSS2	9.7771e-014	4.9192e-013
F5	SPSO	4.3662e-003	6.3953e-003
	MPSO-TVAC	3.6624e-004	2.0060e-003
	PSO-ILCSS1	9.4223e-013	3.9129e-012
	PSO-ILCSS2	4.6303e-015	1.0950e-014

Table 11. The Comparison Results of Benchmark Function

From the Table 2, PSO-ILCSSI and PSO-ILCSS2 both maintain better performances than SPSO and MPSO-TVAC no matter the average mean value or standard deviation. The dynamic performances of PSO-ILCSSI and PSO-ILCSS2 are near the same with SPSO and MPSO-TVAC in the first stage, although PSO-ILCSSI and PSO-ILCSS2 maintains quick global search capability in the last period. In one words, the performances of PSO-ILCSSI and PSO-ILCSS2 surpasses slightly than which of MPSO-TVAC and SPSO a little for unimodel functions, while for the multi-model functions, PSO-ILCSSI and PSO-ILCSS2 show preferable results.

5. Individual Social Selection Strategy

5.1 Individual Linear Social Selection Strategy (ILSSS)

Similarly with cognitive parameter, a dispersed control manner (Cai et al., 2008) is introduced, in which each particle selects its social coefficient value to decide the search direction: $\overrightarrow{p}_j$ or $\overrightarrow{p_g}$.

Since the literatures only consider the extreme value $\overrightarrow{p_g}$, however, they neglect the differences between $\overrightarrow{p}_j$ and $\overrightarrow{p_g}$. These settings lose some information maybe useful to find the global optima or escape from a local optima. Thus, we design a new index by introducing the performance differences, and the definition is provided as follows:

$$Grade_u(t) = \frac{f_{worst}(t) - f(x_u(t))}{f_{worst}(t) - f_{best}(t)} \tag{21}$$

where $f_{worst}(t)$ and $f_{best}(t)$ are the worst and best fitness values of the swarm at time t, respectively. Occasionally, the swarm converges onto one point, that means $f_{worst}(t) = f_{best}(t)$. In this case, the value $Grade_u(t)$ of arbitrary particle u is set to 1. $Grade_u(t)$ is an information index to represent the differences of particle u at time t, according to its fitness value of the current position. The better the particle is, the larger $Grade_u(t)$ is, and vice versa.

As we known, if the fitness value of particle u is better than which of particle m, the probability that global optima falls into m's neighborhood is larger than that of particle m. In this manner, the particle u should pay more attentions to exploit its neighborhood. On the contrary, it may tend to explore other region with a larger probability than exploitation. Thus, for the best solution, it should make complete local search around its historical best position, as well as for the worst solution, it should make global search around $\overrightarrow{p_g}$. Then, the dispersed social coefficient of particle j at time t is set as follows:

$$c_{2,j}(t) = c_{low} + (c_{up} - c_{low}) \times Grade_j(t) \tag{22}$$

where c_{up} and c_{low}, are two predefined numbers, and $c_{2,j}(t)$ represents the social coefficient of particle j at time t.

5.2 Individual Non-linear Social Selection Strategy(INLSSS)

As mentioned before, although the individual linear social parameter selection strategy improves the performance significantly, however, its linear manner can not meet the complex optimization tasks. Therefore, in this section, we introduce four different kinds of non-linear manner, and investigate the affection for the algorithm's performance.

Because there are fruitful results about inertia weight, therefore, an intuitive and simple method is to introduce some effective non-linear manner of inertia weight into the study of social parameter automation. Inspired by the previous literatures (Chen et al., 2006; Jiang & Etorre, 2005), four different kinds of nonlinear manner are designed.

The first non-linear social automation strategy is called parabola opening downwards strategy :

$$c_{2,j}(t) = c_{start} - (c_{start} - c_{end}) \times (\frac{t}{MAX\_ITER})^2 \times Grade_j(t) \tag{23}$$

The second non-linear social automation strategy is called parabola opening upwards strategy:

$$\begin{aligned} c_{2,j}(t) = c_{start} + (c_{start} - c_{end}) \times (\frac{t}{MAX\_ITER})^2 \\ -(c_{start} - c_{end}) \times \frac{2t}{MAX\_ITER} \times Grade_j(t) \end{aligned} \tag{24}$$

The third non-linear social automation strategy is called exponential curve strategy:

$$c_{2,j}(t) = c_{end} \cdot (\frac{c_{start}}{c_{end}})^{\frac{1}{1+10\frac{t}{MAX\_ITER}}} \times Grade_j(t) \tag{25}$$

The fourth non-linear social automation strategy is called negative-exponential strategy:

$$c_{2,j}(t) = c_{start} \times e^{-\frac{t}{MAX\_ITER}} \times Grade_j(t) \tag{26}$$

5.3 The Steps of PSO-INLSSS

The detail steps of PSO-INLSSS are listed as follows:

- Step 1. Initializing each coordinate x_j^k and v_j^k sampling within $[x_{min}, x_{max}]$ and $[-v_{max}, v_{max}]$, respectively.
- Step 2. Computing the fitness value of each particle.
- Step 3. For $k'th$ dimensional value of $j'th$ particle, the personal historical best position p_j^k is updated as follows.

$$p_j^k = \begin{cases} x_j^k, & \text{if } f(\overrightarrow{x}_j) < f(\overrightarrow{p}_j), \\ p_j^k & , \text{ otherwise.} \end{cases} \tag{27}$$

- Step 4. For $k'th$ dimensional value of $j'th$ particle, the global best position p_g^k is updated as follows.

$$p_g^k = \begin{cases} p_j^k, & \text{if } f(\overrightarrow{p}_j) < f(\overrightarrow{p}_g), \\ p_g^k & , \text{ otherwise.} \end{cases} \tag{28}$$

- Step 5. Computing the social coefficient $c_{2,j}$ value of each particle according to formula (23)- (26).
- Step 6. Updating the velocity and position vectors with equation (1)-(3) in which social coefficient c_2 is changed with $c_{2,j}$.
- Step 7. Making mutation operator described in section 4.3.
- Step 8. If the criteria is satisfied, output the best solution; otherwise, goto step 2.

5.4 Simulation Results

To testify the performance of these four proposed non-linear social parameter automation strategies, three famous benchmark functions are chosen to test the performance, and compared with standard PSO (SPSO), modified PSO with time- varying accelerator coefficients (MPSO-TVAC) (Ratnaweera et al., 2004) and individual social selection strategy (PSO-ILSSS). Since we adopt four different non-linear strategies, the proposed methods are called PSO-INLSSS-1 (with strategy one), PSO-INLSSS-2 (with strategy two), PSO-INLSSS-3 (with strategy three) and PSO-INLSSS-4 (with strategy four), respectively. The details of the experimental environment and results are explained as follows.

5.4.1 Benchmarks

In this paper, three typical unconstraint numerical benchmark functions are used to test.
Rastrigin Function:

$$f_1(x) = \sum_{j=1}^{n}[x_j^2 - 10cos(2\pi x_j) + 10]$$

where $|x_j| \leq$5.12, and

$$f_1(x^*) = f_6(0, 0, ..., 0) = 0.0$$

Ackley Function:

$$f_2(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2})$$

$$-exp(\frac{1}{n}\sum_{k=1}^{n}\cos 2\pi x_k) + 20 + e$$

where $|x_j| \leq$32.0, and

$$f_2(x^*) = f_3(0, 0, ..., 0) = 0.0$$

Penalized Function:

$$f_3(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]$$

$$+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$$

where $|x_j| \leq$50.0, and

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

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

$$f_3(x^*) = f_8(1, 1, ..., 1) = 0.0$$

5.4.2 Parameter Settings

The coefficients of SPSO, MPSO-TVAC, PSO-ILSSS and PSO-INLSSS are set as follows:

The inertia weight w is decreased linearly from 0.9 to 0.4 within SPSO, MPSO-TVAC, PSO-ILSSS and PSO-INLSSS. Accelerator coefficients c_1 and c_2 are set to 2.0 within SPSO, as well as in MPSO-TVAC, ci decreases from 2.5 to 0.5, while c_2 increases from 0.5 to 2.5. For PSO-ILSSS and PSO-INLSSS, cognitive parameter c_1 is fixed to 2.0, while social parameter c_2 is decreased, whereas the lower bounds of c_2 is set to 1.0, and the upper bounds is set from 2.0 decreased to 1.0. Total individuals are 100, and the velocity threshold v_{max} is set to the upper bound of the domain. The dimensionality is 30 and 50. In each experiment, the simulation run 30 times, while each time the largest iteration is 50 x *dimension.*

5.4.3 Performance Analysis

The comparison results of these three famous benchmarks are listed as Table 12-14, in which *Dim.* represents the dimension, *Alg.* represents the corresponding algorithm, *Mean* denotes the average mean value, while *STD* denotes the standard variance.

For Rastrigin Function (Table 12), the performances of all non-linear PSO-INLSSS algorithms are worse than PSO-ILSSS when dimension is 30, although they are better than SPSO and MPSO-TVAC. However, with the increased dimensionality, the performance of non-linear modified variant PSO-INLSSS surpasses that of PSO-ILSSS, for example, the best performance is achieved by PSO-INLSSS-3. This phenomenon implies that non-linear strategies can exactly affect the performance.

For Ackley Function (Table 13) and Penalized Function (Table 14), the performance of PSO-INLSSS-3 always wins. Based on the above analysis, we can draw the following two conclusions:

PSO-INLSSS-3 is the most stable and effective among four non-linear strategies. It is especially suit for multi-modal functions with many local optima especially.

Dim.	*Alg.*	Mean	STD
30	SPSO	1.7961e+001	4.2276e+000
	MPSO-TVAC	1.5471e+001	4.2023e+000
	PSO-ILSSS	6.4012e+000	5.0712e+000
	PSO-INLSSS-1	6.8676e+000	3.1269e+000
	PSO-INLSSS-2	8.2583e+000	2.3475e+000
	PSO-INLSSS-3	8.8688e+000	1.7600e+000
	PSO-INLSSS-4	1.0755e+001	4.2686e+000
50	SPSO	3.9958e+001	7.9258e+000
	MPSO-TVAC	3.8007e+001	7.0472e+000
	PSO-ILSSS	1.5380e+001	5.5827e+000
	PSO-INLSSS-1	1.4329e+001	4.7199e+000
	PSO-INLSSS-2	1.5623e+001	4.4020e+000
	PSO-INLSSS-3	1.3740e+001	4.3426e+000
	PSO-INLSSS-4	2.1975e+001	5.6844e+000

Table 12. Comparison Results for Rastrigin Function

6. Conclusion and Future Research

This chapter proposes a new model incorporated with the characteristic differences for each particle, and the individual selection strategy for inertia weight, cognitive learning factor and social learning factor are discussed, respectively. Simulation results show the individual selection strategy maintains a fast search speed and robust. Further research should be made on individual structure for particle swarm optimization.

Dim.	*Alg.*	Mean	STD
30	SPSO	5.8161e-006	4.6415e-006
	MPSO-TVAC	7.5381e-007	3.3711e-006
	PSO-ILSSS	4.7853e-011	9.1554e-011
	PSO-INLSSS-1	1.8094e-011	1.8533e-011
	PSO-INLSSS-2	1.1870e-011	2.0876e-011
	PSO-INLSSS-3	5.2100e-013	5.5185e-013
	PSO-INLSSS-4	3.2118e-010	2.2272e-010
50	SPSO	1.7008e-004	1.2781e-004
	MPSO-TVAC	4.4132e-002	1.9651e-001
	PSO-ILSSS	1.5870e-008	1.7852e-008
	PSO-INLSSS-1	2.3084e-008	3.6903e-008
	PSO-INLSSS-2	1.1767e-008	1.3027e-008
	PSO-INLSSS-3	4.7619e-010	1.4337e-009
	PSO-INLSSS-4	3.4499e-008	4.7674e-008

Table 13. Comparison Results for Ackley Function

Dim.	*Alg.*	Mean	STD
30	SPSO	5.4943e-004	2.45683e-003
	MPSO-TVAC	9.3610e-027	4.1753e-026
	PSO-ILSSS	5.1601e-023	1.7430e-022
	PSO-INLSSS-1	6.0108e-020	1.5299e-019
	PSO-INLSSS-2	4.5940e-021	6.2276e-021
	PSO-INLSSS-3	9.7927e-024	1.6162e-023
	PSO-INLSSS-4	1.0051e-016	1.9198e-016
50	SPSO	6.4279e-003	1.0769e-002
	MPSO-TVAC	4.9270e-002	2.0248e-001
	PSO-ILSSS	1.6229e-017	3.9301e-017
	PSO-INLSSS-1	6.2574e-015	1.3106e-014
	PSO-INLSSS-2	1.6869e-014	3.3596e-014
	PSO-INLSSS-3	6.2959e-018	5.6981e-018
	PSO-INLSSS-4	8.0886e-013	3.7972e-013

Table 14. Comparison Results for Penalized Function

7. Acknowledgement

This chapter is supported by National Natural Science Foundation of China under Grant No.60674104, and also supported by Doctoral Scientific Research Starting Foundation of Taiyuan University of Science and Technology under Grant No.20082010.

8. References

Holland, J.H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with application to biology, control, and artificial intelligence*, 2nd Edition, Cambridge, MA: MIT Press. [1]

Dorigo, M. & Gambardella, L.M. (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation,* Vol.1, No.l, 53-66. [2]

Eberhart, R.C. & Kennedy, J. (1995). A new optimizer using particle swarm theory, *Proceedings of 6th International Symposium on Micro Machine and Human Science,* pp.39-43. [3]

Kennedy, J. & Eberhart, R.C. (1995). Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks,* pp. 1942-1948. [4]

Shen, H.Y.; Peng, X.Q.; Wang, J.N. & Hu, Z.K. (2005). A mountain clustering based on improved PSO algorithm, *Lecture Notes on Computer Science 3612,* Changsha, China, pp.477-481. [5]

Eberhart, R.C. & Shi, Y. (1998). Extracting rules from fuzzy neural network by particle swarm optimization, *Proceedings of IEEE International Conference on Evolutionary Computation,* Anchorage, Alaska, USA. [6]

Li, Q.Y.; Shi, Z.P.; Shi, J. & Shi, Z.Z. (2005). Swarm intelligence clustering algorithm based on attractor, *Lecture Notes on Computer Science 3612,Changsha,* China, pp.496-504. [7]

Shi, Y. & Eberhart, R.C. (1998a). A modified particle swarm optimizer, *Proceedings of the IEEE International Conference on Evolutionary Computation,* Anchorage, Alaska, USA,pp.69-73. [8]

Shi Y. & Eberhart R.C. (1998b). Parameter selection in particle swarm optimization, *Proceedings of the 7th Annual Conference on Evolutionary Programming,* pp.591-600. [9]

Shi Y. & Eberhart R.C. (1999). Empirical study of particle swarm optimization, *Proceedings of the Congress on Evolutionary Computation,* pp. 1945-1950. [10]

Suganthan P.N. (1999). Particle swarm optimizer with neighbourhood operator, *Proceedings of the Congress on Evolutionary Computation,* pp. 1958-1962. [11]

Venter, G. (2002). Particle swarm optimization, *Proceedings of 43rd AIAA/ASME/ASCE/AHS/ASC Structure, Structures Dynamics and Materials Conference,* pp.22-25. [12]

Ratnaweera, A.; Halgamuge, S.K. & Watson, H.C. (2004). Self-organizing hierarchical particle swarm opitmizer with time-varying acceleration coefficients, *IEEE Transactions on Evolutionary Computation,* Vol.8, No.3, 240-255. [13]

Monson, C.K. & Seppi, K.D. (2004). The Kalman swarm: a new approach to particle motion in swarm optimization, *Proceedings of the Genetic and Evolutionary Computation Conference,pp.* 140-150. [14]

Sun, J. et al. (2004). Particle swarm optimization with particles having quantum behavior, *Proceedings of the IEEE Congress on Evolutionary Computation,* pp.325-331. [15]

Cui, Z.H. & Zeng, J.C. (2004). A guaranteed global convergence particle swarm optimizer, *Lecture Notes in Artificial Intelligence,* vol.3066, Sweden, pp.762-767. [16]

Cui, Z.H.; Zeng, J.C. & Sun, G.J. (2006a). A fast particle swarm optimization, *International Journal of Innovative Computing, Information and Control,* Vol.2, No.6,pp.l365-1380. [17]

Cui, Z.H.; Cai, X.J.; Zeng, J.C. & Sun, G.J. (2006b). Predicted-velocity particle swarm optimization using game-theoretic approach, *Lecture Notes in Computer Science,* vol.4115, Kunming, China, pp.145-154. [18]

Mendes, R.; Kennedy, J. & Neves, J. (2004). The fully informed particle swarm: simper, maybe better, *IEEE Transactions on Evolutionary Computation,* Vol.8, No.3, pp.204-210. [19]

Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, *Poceedings of the 1999 Congress on Evolutionary Computation,* pp.1931-1938. [20]

Løvbjerg, M.; Rasmussen, T.K. & Krink, T. (2001). Hybrid particle swarm optimiser with breeding and subpopulations, *Proceedings of the third Genetic and Evolutionary Computation Conference.* [21]

Cai, X.J.; Cui, Z.H.; Zeng, J.C. & Tan Y. (2007a). Performance-dependent adaptive particle swarm optimization, *International Journal of Innovative Computing, Information and Control,* Vol.3, No.GB, 1697-1706. [22]

Michalewicz, Z. (1992). *Genetic Algorithm+ Data Structures =Evolution Programs,* Springer-Verlag, Berlin. [23]

Marcus, H. (2002). Fitness uniform selection to preserve genetic diversity, *Proceedings of IEEE Congress on Evolutionary Computation,* pp.783-388. [24]

Blickle, T. & Thiele, L. (1995). A mathematical analysis of tournament selection, *Proceedings of the Sixth International Conference on Genetic Algorithms,* San Francisco, California, pp.9-16. [25]

Cui, Z.H; Cai, X.J. & Zeng J.C. (2008). Some Non-linear Score Strategies in *PDPSO, ICIC Express Letters* Vol.2, No.3. [26]

Liang, J.J.; Qin, A.K.; Suganthan, P.N. & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Transactions on Evolutionary Computation,* Vol.10, No.3, 281-295. [27]

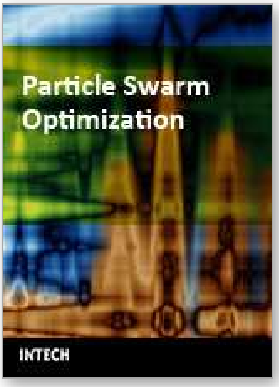
Shang, Y.W. & Qiu, Y.H. (2006). A note on the extended Rosenbrock function, *Evolutionary Computation,* Vol.14, No.l, 119-126. [28]

Cai, X.J.; Cui, Z.H.; Zeng, J.C. & Tan Y. (2007b). Self-learning Particle Swarm Optimization Based on Environmental FeedbackProceerfzngs *of the Second International Conference on Innovative Computing, Information and Control (ICICIC2007),* Japan. [29]

Cai, X.J.; Cui Z.H.; Zeng, J.C. & Tan, Y. (2008). Particle Swarm Optimization with Self-adjusting Cognitive Selection Strategy, *International Journal of Innovative Computing, Information and Control (IJICIC),* Vol.4, No.4, 943-952. [30]

Chen, G.M.; Jia, J.Y. & Han, Q. (2006). Study on the Strategy of Decreasing Inertia Weight in Particle Swarm Optimization Algorithm, *Journal of Xi'an Jiao Tong University*, Vol.40, No.l, pp.53-56. (in Chinese) [31]

Jiang, C.W. & Etorre, B. (2005). A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment, *Energy Conversion and Management*, Vol.46, No.17, pp.2689-2696. [32]



Particle Swarm Optimization
Edited by Aleksandar Lazinica

ISBN 978-953-7619-48-0
Hard cover, 476 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

Particle swarm optimization (PSO) is a population based stochastic optimization technique influenced by the social behavior of bird flocking or fish schooling.PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. This book represents the contributions of the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Xingjuan Cai, Zhihua Cui, Jianchao Zeng and Ying Tan (2009). Individual Parameter Selection Strategy for Particle Swarm Optimization, Particle Swarm Optimization, Aleksandar Lazinica (Ed.), ISBN: 978-953-7619-48-0, InTech, Available from: http://www.intechopen.com/books/particle_swarm_optimization/individual_parameter_selection_strategy_for_particle_swarm_optimization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.