Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2012, Article ID 791373, 21 pages
doi:10.1155/2012/791373

Research Article

A Novel PSO Model Based on Simulating Human Social Communication Behavior

Yanmin Liu[1,2] and Ben Niu[3,4,5]

[1] *School of Economics and Management, Tongji University, Shanghai 200092, China*
[2] *School of Mathematics and Computer Science, Zunyi Normal College, Zunyi 563002, China*
[3] *College of Management, Shenzhen University, Shenzhen 518060, China*
[4] *Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, China*
[5] *E-Business Technology Institute, The University of Hong Kong, Hong Kong*

Correspondence should be addressed to Ben Niu, drniuben@gmail.com

Received 11 May 2012; Revised 22 June 2012; Accepted 25 June 2012

Academic Editor: Vimal Singh

In order to solve the complicated multimodal problems, this paper presents a variant of particle swarm optimizer (PSO) based on the simulation of the human social communication behavior (HSCPSO). In HSCPSO, each particle initially joins a default number of social circles (SC) that consist of some particles, and its learning exemplars include three parts, namely, its own best experience, the experience of the best performing particle in all SCs, and the experiences of the particles of all SCs it is a member of. The learning strategy takes full advantage of the excellent information of each particle to improve the diversity of the swarm to discourage premature convergence. To weight the effects of the particles on the SCs, the worst performing particles will join more SCs to learn from other particles and the best performing particles will leave SCs to reduce their strong influence on other members. Additionally, to insure the effectiveness of solving multimodal problems, the novel parallel hybrid mutation is proposed to improve the particle's ability to escape from the local optima. Experiments were conducted on a set of classical benchmark functions, and the results demonstrate the good performance of HSCPSO in escaping from the local optima and solving the complex multimodal problems compared with the other PSO variants.

1. Introduction

Particle swarm optimization (PSO), originally introduced by Kennedy and Eberhart [1], has proven to be a powerful competitor to other evolutionary algorithms (e.g., genetic algorithms) [2]. In PSO, these individuals, instead of being manipulated by the evolution operator such as crossover and mutation, are "evolved" by the cooperation and competition

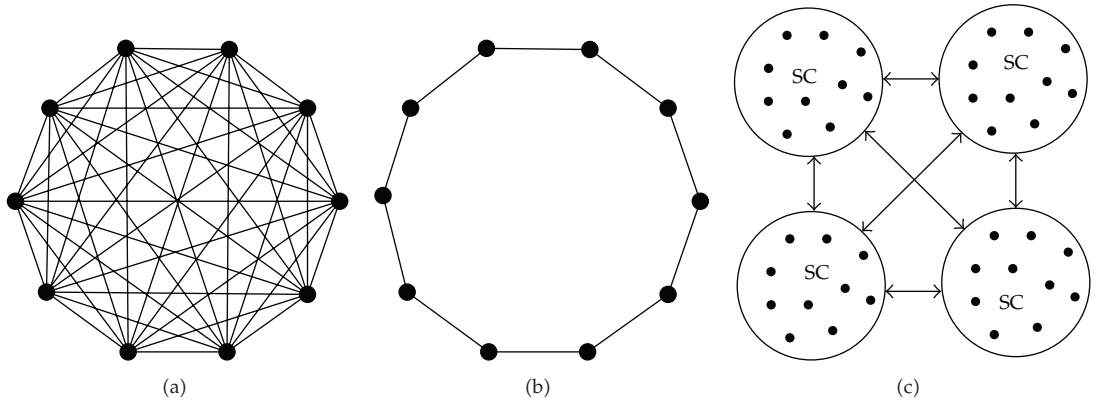


Figure 1: (a) Global neighborhood; (b) local neighborhood; (c) the proposed algorithm neighborhood.

among the individuals through generations. Each individual in the swarm is called a particle (a point) with a velocity that is dynamically adjusted in the search process according to its own flying experience and the best experience of the swarm.

When solving the unconstraint optimization problem, PSO has empirically turned out to perform well on many optimization problems. However, when it comes to solving complex multimodal problems, PSO may easily get trapped in a local optimum. In order to overcome this defect and improve PSO performance, some researchers proposed several methods [3–20]. In this paper, we present an improved PSO based on human social communication. This strategy ensures the swarm's diversity against the premature convergence, especially when solving the complex multimodal problems.

This paper is organized as follows. Section 2 presents an overview of the original PSO, as well as a discussion of the previous attempts to improve the PSO performance. Section 3 proposed an improved PSO based on simulation of human communication. Section 4 gives the test functions, the experimental setting, and results. Finally, some conclusions and the future works are discussed in Section 5.

2. Particle Swarm Optimization

2.1. The Original PSO (OPSO)

The original PSO algorithm (OPSO) was inspired by the search behavior of the biological organisms, where each particle moves through the search space by a combination of the best position found so far by itself and its neighbors. In the PSO domain, generally there are two main neighborhood topologies, namely, the global and the local neighborhood that are shown in Figures 1(a) and 1(b), respectively.

The two neighborhood topologies derive two classical PSO variants, namely, the global PSO (GPSO) and the local PSO (LPSO) in which the behaviors of the particles are described by (2.1) and (2.2), respectively. In HSCPSO, the neighborhood topology is somewhat similar to four clusters [4], but not the same as one, as can be seen in Figure 1(c). Here, the black dots represent the particles in each social circle (SC), the circles represent the social circle, and the arrows represent the relationship between SCs. Note that in HSCPSO each particle's neighborhood is the set of the particles of all SCs that it is a member of (see Section 3.1).

Consider the following:

$$\begin{aligned} \vec{v_i}(t) &= w \cdot \vec{v_i}(t-1) + \varphi_1 \cdot r_1\left(\vec{p_i} - \vec{x_i}(t-1)\right) + \varphi_2 \cdot r_2\left(\vec{p_g} - \vec{x_i}(t-1)\right), \\ \vec{x_i}(t) &= \vec{x_i}(t-1) + \vec{v_i}(t), \end{aligned} \tag{2.1}$$

$$\begin{aligned} \vec{v_i}(t) &= w \cdot \vec{v_i}(t-1) + \varphi_1 \cdot r_1\left(\vec{p_i} - \vec{x_i}(t-1)\right) + \varphi_2 \cdot r_2\left(\overrightarrow{p_{\text{neighbor}_i}} - \vec{x_i}(t-1)\right), \\ \vec{x_i}(t) &= \vec{x_i}(t-1) + \vec{v_i}(t), \end{aligned} \tag{2.2}$$

where $i = 1, \ldots, ps$, ps is the population size; t is the iteration counter; n is the dimension of the search space; φ_1 and φ_2 denote the acceleration coefficients; r_1 and r_2 are random vectors with the components uniformly distributed in the range of $[0,1]$; $\vec{x_i}(t) = (v_i^1, v_i^2, \ldots, v_i^n)$ represents the position of the ith particle at iteration t; $\vec{v_i}(t) = (v_i^1, v_i^2, \ldots, v_i^n)$ represents the velocity of the ith particle; $\vec{p_i}$ is the best position yielding the best fitness value for the ith particle; $\vec{p_g}$ is the best position discovered by the whole population; $\overrightarrow{p_{\text{neighbor}_i}}$ is the best position achieved within the neighborhood of current particle i. Note, in this paper the original PSO represent two PSOs, one is PSO with inertia weight and ring topology, and the other is PSO with inertia weight and global topology.

2.2. Related Works

Since the introduction of PSO, PSO has attracted a great deal of attention. Many researchers have worked on improving its performance in various ways and derived many interesting variants. In [3], Clerc and Kennedy indicated that a constriction factor χ may help to the convergence of PSO. The velocity and position of the ith particle are updated as follows:

$$\begin{aligned} \vec{v_i}(t) &= \chi \cdot \left\{\vec{v_i}(t-1) + \varphi_1 \cdot r_1\left(\vec{p_i} - \vec{x_i}(t-1)\right) + \varphi_2 \cdot r_2\left(\vec{p_g} - \vec{x_i}(t-1)\right)\right\}, \\ \vec{x_i}(t) &= \vec{x_i}(t-1) + \vec{v_i}(t), \end{aligned} \tag{2.3}$$

where $\chi = 2/|2 - \phi - \sqrt{\phi^2 - 4\phi}|$ and $\phi = \phi_1 + \phi_2$, $\phi > 4$.

Kennedy and Mendes [4] claimed that PSO with a small neighborhood might perform better on the complex problems, while the one with a large neighborhood would perform better on the simple problems. In [5], the author proposed a quantum-behaved particle swarm optimization (QPSO) to improve PSO performance. Some researchers [6–9] also combined some techniques to improve PSO performance (e.g., evolutionary operators). However, these improvements are based on a static neighborhood network, which greatly decreases the swarm diversity due to that the particle only learns from the fixed neighborhood. Hence, Suganthan [10] and Hu and Eberhart [11] proposed a dynamic neighborhood topology which transforms gradually from acting like the local neighborhood in the early stage of the search to behaving more like the global neighborhood in the late stage of the search. Liang et al. [12] proposed an improved PSO called CLPSO, which used a novel learning strategy where all particles' historical best information is used to update a particle's velocity. In [13], the author proposed the fitness-distance-ratio-based PSO (FDR-PSO) where FDR-PSO selects another particle $\vec{p_i}$ which is supposed to be a higher fitness value and the nearest to the particle being updated. Mohais et al. [14] proposed a randomly generated

directed graph to define neighborhood which is generated by using two methods, namely, the random edge migration method and the neighborhood restructuring method. Janson and Middendorf [15] arranged the particles in a hierarchy structure, where the best performing particles ascend the tree to influence more particles, replacing relatively worse performing particles which descend the tree. In [16], clubs-based particle swarm optimization (C-PSO) algorithm was proposed, whose membership is dynamically changed to avoid premature convergence and improve performance. In [17], Mendes and Kennedy introduced a fully informed PSO where instead of using the *pbest* (*pbest* (personal best): personal best position of a given particle, so far) and *gbest* (*gbest* (global best): position of the best particle of the entire swarm) positions in the standard algorithm, all the neighbors of the particle are used to update the velocity. The influence of each particle on its neighbors is weighted based on its fitness value and the neighborhood size. In [18], the authors proposed a cooperative particle swarm optimizer (CPSO-H). Although CPSO-H uses one-dimensional (1D) swarms to search each dimension separately, the results of these searches are integrated by a global swarm to significantly improve the performance of the original PSO on multimodal problems. In recent works [19, 20], the authors proposed the dynamic neighborhood topology where the whole population is divided into a number of sub-swarms. These subswarms are regrouped frequently by using various regrouping schedules and information exchange among all particles in the whole swarm.

The main differences of our approach with respect to the other proposals existing in the above literatures are as follows.

(i) In HSCPSO, we create the social circle (SC) for the particles analogous to our communities where people can communicate and study to widen the knowledge to each other.

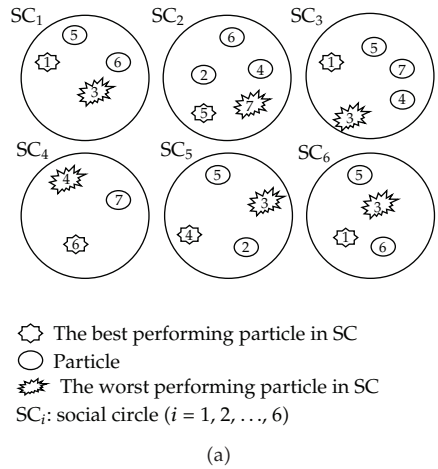
(ii) The learning exemplars of each particle include three parts, namely, its own best experience, the experience of the best performing particle in all SCs, and the experiences of the particles of all SCs it is a member of. This strategy greatly ensures the swarm diversity against the premature convergence.

(iii) The parallel hybrid mutation is used to improve the particle's ability to escape from the local optima.

3. PSO Based on Simulation of Human Communication Behavior in the Society

3.1. Updating Strategy of Particle Velocity

Based on the simulation of the human social communication behavior, each particle can join more than one SC, and each SC can accommodate any number of particles. Vacant SCs also are allowed. Firstly, each particle randomly joins a predefined number of SCs, which is called the default social circle level (DSC). At each run, the worst performing particles are more socialized through joining more SCs to improve their knowledge, while the best performing particles are less socialized through leaving an SC to reduce their strong influences on other members, which leads to the fact that DSC is dynamically adjusted in terms of the particle's performance. During this cycle of leaving and joining SCs, the particles that no longer show the extreme performance in its neighborhood will gradually return to DSC. The speed of regaining DSC will decide the algorithm performance, so the check is made every n iteration (here, n is called the gap iteration for adjusting DSC) to find the particles that have SC level



(a)

```
Begin
 For i = 1:ps//ps-swarm size
  If (x_i is the best in its neighbors_i) and
    (|membership_i|>the minimum of allowed SC number)
    Randomly leave SC
  EndIf
  If (x_i is the worst in its neighbors_i) and
    (|membership_i|< maximum of allowed SC number)
    Randomly join SC
  EndIf
  If (the gap iteration for adjusting DSC is satisfied)
    and (|membership_i|≠DSC)
    Update SC of each particle
  EndIf
 EndFor
End
```

(b)

Figure 2: The snapshot of the SCs at iteration t.

above or below DSC, and then take them back to DSC if they do not show the extreme performance. Thus, the gap iteration for adjusting DSC needs a suitable value to ensure the performance efficiency of HSCPSO.

In order to control when a particle joins or leaves an SC, we designed a mechanism to control the process. If a particle continues to show the worst performance compared with the particles in the SCs that it is a member of, then it will join more SCs one after the other until the number of SCs reaches the maximum of allowed SC number defined by the user, while the one that continues to show the superior performance in the SCs that it is a member of will leave SCs one after another until the number of SC reaches the minimum of allowed SC number. The methods to determine DSC, the gap iteration for adjusting DSC, the maximum of allowed SC number, and the minimum of allowed SC number will be discussed in the following.

Figure 2(a) shows a snapshot of the SCs during an execution of HSCPSO at iteration t. In this example, the swarm consists of 7 particles, and there are 6 SCs available for them to join. The minimum of allowed SC number, DSC, and maximum of allowed SC number are 2, 3 and 5, respectively. Particle 1 will leave social circle 1 (SC_1), SC_3, or SC_6 because it is the best performing particle in its neighborhood. Particle 3 will join SC_2 or SC_4, because it is the worst performing particle in its neighborhood. Particle 4 will leave SC_2, SC_3, SC_4, or SC_5 to go back to DSC, while Particle 2 will join SC_1, SC_3, SC_4 or SC_6 to return to DSC. Figure 2(b) gives the pseudocode of SCs during an execution of HSCPSO, where neighbor_i is the set of the neighbors of particle i (note that in HSCPSO each particle's neighborhood is the set of the particles of all SCs that it is a member of), and $|\text{membership}_i|$ is the set of the SCs that particle i is a member of.

The thought of HSCPSO is somewhat similar to C-PSO [16], but not the same as one. In our algorithm, when updating the particle's velocity, a particle does not learn from all dimensions of the best performing particle in its neighbors, but learns from the corresponding dimension of the best performing particles of the SCs that it is a member of. In order to compare with the difference of the two strategies, the experiment was conducted as follows: HSCPSO and C-PSO are run 20 times on Sphere, Rosenbrock, Ackley, Griewanks,

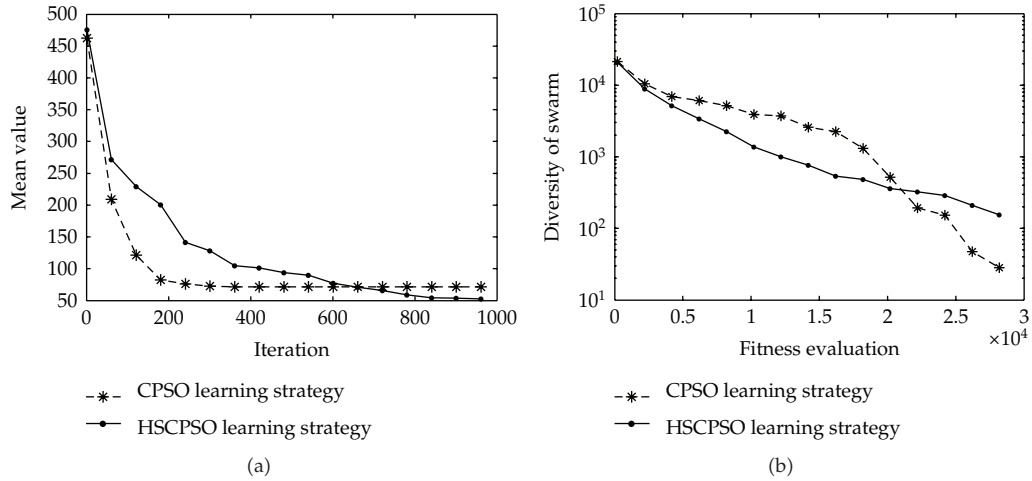


Figure 3: Comparison of the different learning strategies.

and Rastrigin functions, respectively; the iteration of each run is set as 1000. The mean values of the results are plotted in Figure 3(a).

As all test functions are the minimization problems, the smaller the mean value is, the better the performance works. From Figure 3, we can observe that the learning strategy of C-PSO suffers the premature convergence, while the strategy of HSCPSO not only ensures the diversity of the swarm, but also improves the swarm ability to escape from the local optima.

In the real world, everyone has the ability to make himself be one member in any SC to widen the visual field. In a similar way, we hypothesize that each particle in the swarm has the same ability as human in the society. Based on our previous work [20], the following updating equation of the velocity and position is employed in HSCPSO:

$$\begin{aligned}\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + \varphi_1 \cdot r_1\left(\vec{p}_i - \vec{x}_i(t-1)\right) + \varphi_2 \cdot r_2\left(\overrightarrow{p_{bin_{(i)}}} - \vec{x}_i(t-1)\right) \\ + \varphi_3 \cdot r_3\left(\vec{p}_g - \vec{x}_i(t-1)\right), \\ \vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t),\end{aligned} \tag{3.1}$$

where $\vec{p}_i = (p_i^1, p_i^2, \ldots, p_i^n)$ is the best position for the ith particle; $\vec{p}_g$ denotes the experience of the best performing particle in all SCs; $\overrightarrow{p_{bin_{(i)}}}$ is called the comprehensive strategy in which the particles' historical best information of the SCs that particle i is a member of is used to update a particle's velocity. Here, φ_1=0.972, φ_2=0.972, φ_2=0.972, and w = 0.729. In the comprehensive strategy, the pseudocode of the learning exemplar choice is shownin Algorithm 1.

3.2. Parallel Hybrid Mutation

In [4], the author has concluded that PSO converges rapidly in the first search process and then gradually slows down or stagnates. The phenomenon is caused by the loss of diversity in the swarm. In order to conquer the default, some researchers [7–9] applied the evolutionary

```
% SC_i denotes the set of the particles of all SCs that particle i is a member of
% fit(p) represents the corresponding fitness value of an array p
(1) Function comprehensive_learning ()
(2)     For each particle (i = 1 : ps)
(3)         For each dimension (k = 1 : n)%n = particle dimension
(4)             flag(i,k) = 0
(5)             For each particle (j = 1 : ps) %ps-swarm size
(6)                 If (j is the member in SC_i)&&(|fit(p_i) − fit(p_j)|/|p_ik − x_ik| > flag(i,k))
(7)                     p(i,k) = |fit(p_i) − fit(p_j)|/|p_ik − x_ik| %p_i denotes
                        any particle's pbest
(8)                     x_index = j% the particle ID
(9)                 End If
(10)            Next j
(11)            p_bin(i,k) = x_index% the k dimension exemplar of the particle i
(12)        Next k
(13)        Output
(14)    Next i
(15) End function
```

Algorithm 1

```
(1) For i = 1 : ps
(2)     If ceil(mc_i + rand − 1)==1
(3)         If rand ⩽ p_u
(4)             pos(i,d) = (1 + rand) × pos(i,d)
(5)         Else
(6)             pos(i,d) = Gaussian(σ) × pos(i,d)
(7)         End If
(8)     End If
(9) End
```

Algorithm 2

operator to improve the diversity of the swarm. In this paper, we proposed a parallel hybrid mutation which combines the uniform distribution mutation and the gaussian distribution mutation. The former prompts a global search in a large range, while the latter searches in a small range with the high precision. The process of mutation is given as follows.

(i) Give the following expression to set the mutation capability (mc_i) value for each particle,

$$\mathrm{mc}_i = 0.05 + 0.45\frac{(\exp(5(i-1)/ps - 1) - 1)}{\exp(5) - 1}. \tag{3.2}$$

Figure 4(b) shows an the example of mc assigned for 40 particles. Each particle has a mutation ability ranging from 0.05 to 0.5.

(ii) Choose the mode of the mutation, as follows in Algorithm 2.

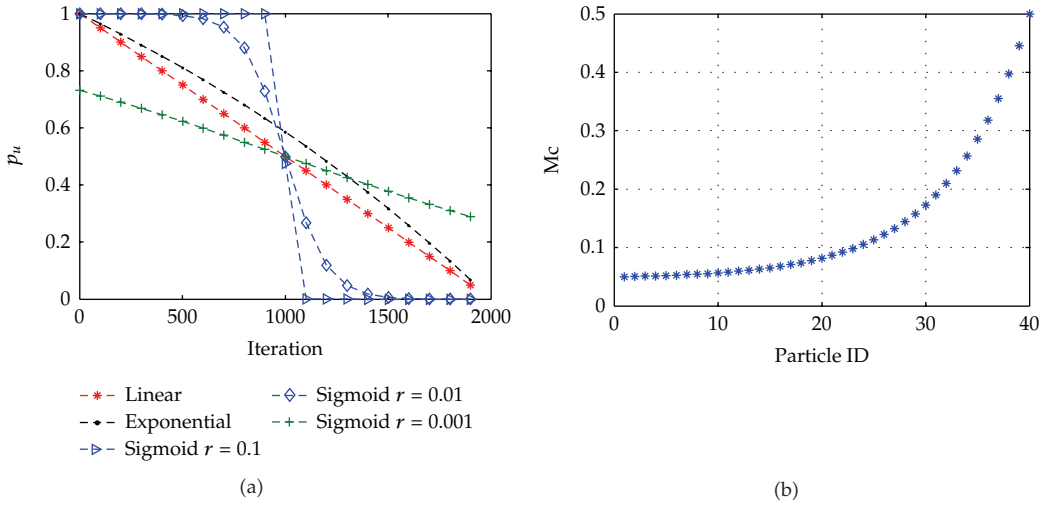


Figure 4: Mutation capability (Mc) and mutation factors.

Here, p_u is called the mutation factor that denotes the ratio of the uniform distribution mutation and correspondingly, $1 - p_u$ is the ratio of the gaussian distribution mutation. The Gaussian(σ) returns a random number drawn from a Gaussian distribution with a standard deviation σ. ceil(p) rounds the elements of p to the nearest integers greater than or equal to p. Here, three main mutation factors (Linear, Exponential, and Sigmoid defined in (3.3), (3.4) and (3.5)) are adopted, and their shapes with maximum generation 2000 are shown in Figure 4(a):

$$p_u(t) = 1 - \frac{t}{\text{gen}}, \tag{3.3}$$

$$p_u(t) = 1 - \left(\exp\left(\frac{t \log 2}{\text{gen}}\right) - 1\right), \tag{3.4}$$

$$f(r,t) = \frac{1}{1 + \exp(-rt)}, \tag{3.5}$$

$$p_u(t) = 1 - f\left(t - \frac{\text{gen}}{2}, r\right), \tag{3.6}$$

where t denotes the current generation; gen is the maximum generation.

In order to test which mutation factor is appropriate to our algorithm, the following experiment is conducted. On Sphere, Rosenbrock, Griewanks, Ackley, Rastrigin_noncont, and Rastrigin, HSCPSO with the different mutation factor is run 30 times on each function, and the iteration is set as 2000. The experimental results are standardized to the same scale with (3.7). The results are presented in Table 1, where p_u represents the type of mutation factor; X_{end} is the standardized value after 2000 iterations; *Linear* represents the linear mutation factor; *Exponent* denotes the exponential mutation factor; *Sigm* is Sigmoid mutation factor; *No* denotes no mutation factor. We can observe that HSCPSO with linear mutation factor achieves the best result, thus the linear mutation factor is adopted in HSCPSO.

Table 1: Standardized value of different mutation factors.

p_u	X_{end}	p_u	X_{end}	p_u	X_{end}
Linear	$6.7e-06$	Sigm0.1	$3.8e-05$	Sigm0.001	$5.9e-04$
Exponent	$8.9e-04$	Sigm0.01	$7.2e-04$	No	$6.8e-03$

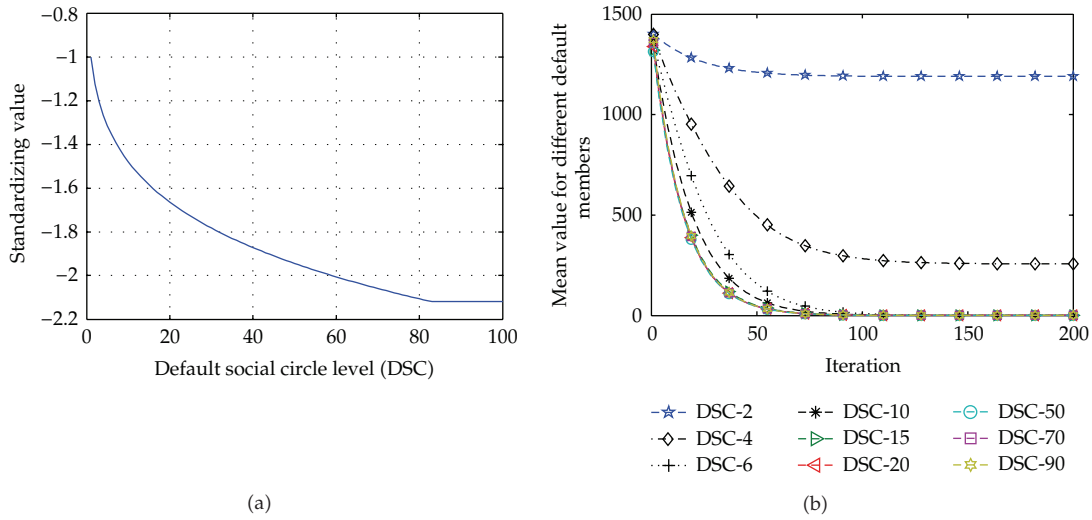


Figure 5: Effect of the default social circles on the algorithm.

3.3. *HSCPSO's Parameter Settings and Analysis of the Swarm's Diversity*

In this section, we will discuss four parameters, namely, default social circle level (DSC), the gap iteration (n) for adjusting DSC, and minimum and maximum of allowed SC number.

3.3.1. *Default Social Circle Level (DSC)*

In order to explore the different DSC effects on HSCPSO, five ten-dimensional test functions are used (Sphere, Rosenbrock, Ackley, Griewanks, and Rastrigin functions). When dealing with experimental data, it is impossible to combine the raw results from the PSOs with the different DSC for the different functions, as they are scaled differently. Thus, Mendes's method [17] is used to deal with the raw results which are standardized to the same scale as follows:

$$X = \frac{x_{ij} - \mu_{ij}}{\sigma_{ij}}, \tag{3.7}$$

where x_{ij}, μ_{ij}, and σ_{ij} denote the trial results, mean, and standard deviation of the ith test function in the jth algorithm, respectively. Note that the parameter j represents the algorithms with the different DSC.

By trial and error, we found that the different DSC yielded the different results as can be seen in Figure 5(a). As all test functions are the minimization problems, the smaller

Table 2: Result of t-tests for the paired comparison of the default social circles.

$\alpha = 0.05$	DSC-15 DSC-2	DSC-15 DSC-4	DSC-15 DSC-6	DSC-15 DSC-10	DSC-15 DSC-20	DSC-15 DSC-50	DSC-15 DSC-70	DSC-15 DSC-90
P-value	0.00314	0.00012	0.00143	0.05912	0.00149	0.00235	0.00438	0.00135
Result	1	1	1	0	1	1	1	1

the standardizing value is, the better the performance works (i.e., the larger the DSC is, the better the performance goes). However, given a second thought, we found the phenomenon is not logical because in the real-life people have no enough energy to take part in all social activities. Therefore, another experiment (the experiment setting and the method of the data processing are the same as one in Figure 5(a)) is conducted to test the impact of DSC. Figure 5(b) gives the simulation results, and we found that the smaller the DSC is, the slower the convergence speed is, and vice versa. The slower convergence rate, rather than the faster convergence speed, is obviously beneficial to the ability to escape from a local optimum [4]. Based on the above analysis, the parameter DSC is set as 15 in our algorithm. Furthermore, in order to ensure the effectiveness of the DSC choice statistically, we adopted the nonparametric Wilcoxon rank sum tests to determine whether the difference is significant or not. The criterion is the following:

$$\text{result} = \begin{cases} 1 & \text{if } p < \alpha, \\ 0 & \text{if } p \geq \alpha. \end{cases} \tag{3.8}$$

If p is less than a, two group numbers are statistically different. If p is equal or greater than a, it means that two group numbers are not statistically different. From Table 2, we can observe that the performance of DSC-15 is statistically different from that of the other DSC values except for DSC-10.

3.3.2. The Gap Iteration (n) for Adjusting DSC

This section mainly discusses the effect of the gap iteration (n) for adjusting DSC on the algorithm. Six different thirty-dimensional test functions (Sphere, Rosenbrock, Griewanks, Ackley, Rastrigin_noncont, and Rastrigin) are used to investigate the impact of this parameter. HSCPSO is run 20 times on each function (the iteration of each run is 1000), and the results are also standardized to the same scale using (3.7). The mean values of the results are plotted in Figure 6(a). It clearly indicates that the gap iteration n can influence the results. When n is 25, that is about 1/40 of the total iteration, a faster convergence speed and a better result are obtained on all test functions. Furthermore, the standardizing diversity in Figure 6(b) also supports the conclusion.

3.3.3. Minimum and Maximum of Allowed SC Number

At each run, the number of the SCs that a particle is a member of is dynamically changed according to its own performance, which will influence the HSCPSO performance. Additionally, in PSO, as the first search process for the global best position requires the exploration of the possible solutions, the LPSO algorithm is chosen in this stage. While the

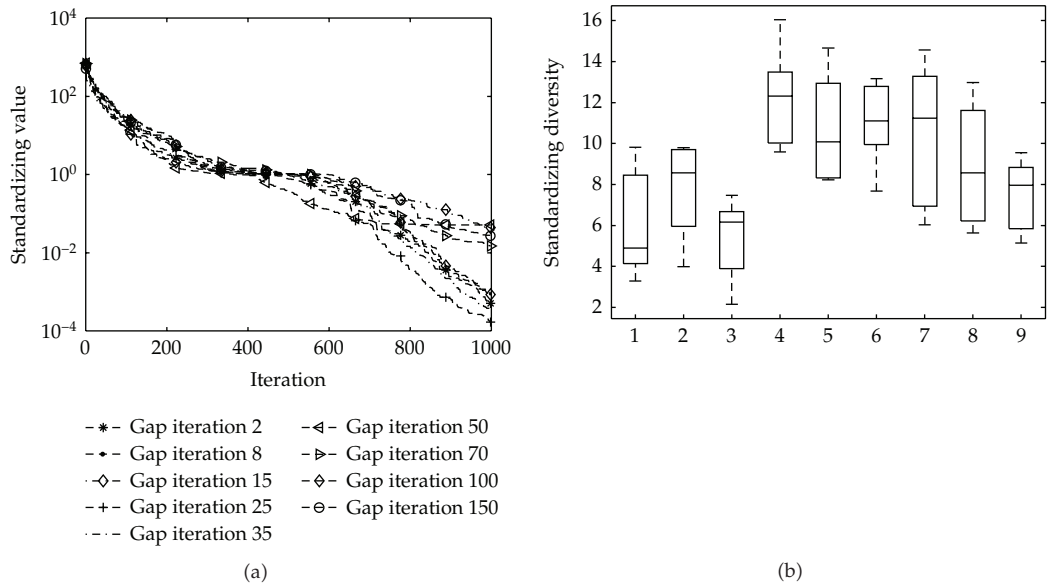


Figure 6: Standardizing mean values and diversity with different gap iterations (n).

later search process requires the exploitation of the best found possible solutions by the early search process, the GPSO algorithm is chosen to meet this requirement. Thus, given the above characteristics, in our algorithm, we made the minimum and maximum of allowed SC number dynamically change with the iteration elapsed and empirically proposed (3.9) to set them, respectively:

$$f_{\min} = 4 + \text{floor}\left(\frac{t}{100}\right) \quad t \in |\text{Maxgen}|,$$
$$f_{\max} = 30 - \text{floor}\left(\frac{t}{100}\right) \quad t \in |\text{Maxgen}|, \tag{3.9}$$

where the floor operation is round towards minus infinity; t is the iteration counter; $|\text{Maxgen}|$ is the total number of the iteration. Figure 7(a) represents the minimum and maximum of allowed SC number with the iteration elapsed, respectively, and we can observe that the dynamic minimum and maximum of allowed SC number (dynamic max-min) improves the swarm diversity compared with the fixed max-min. Note that the measure method of the diversity is presented in [20].

3.3.4. Search Bounds Limitation

There are the bounds on the variables' ranges in many practical problems, and all test functions used in this paper have the bounds. Thus, in order to make the particles fly within the search range, some researchers use the equation $x_i(t) = \min[x_{\max}, \max(x_{\min}, x_i)]$ to restrict a particle on the border. Here, a different method, but similar to the above-mentioned one, is presented; that is, only when a particle is within the search range, we calculate its fitness

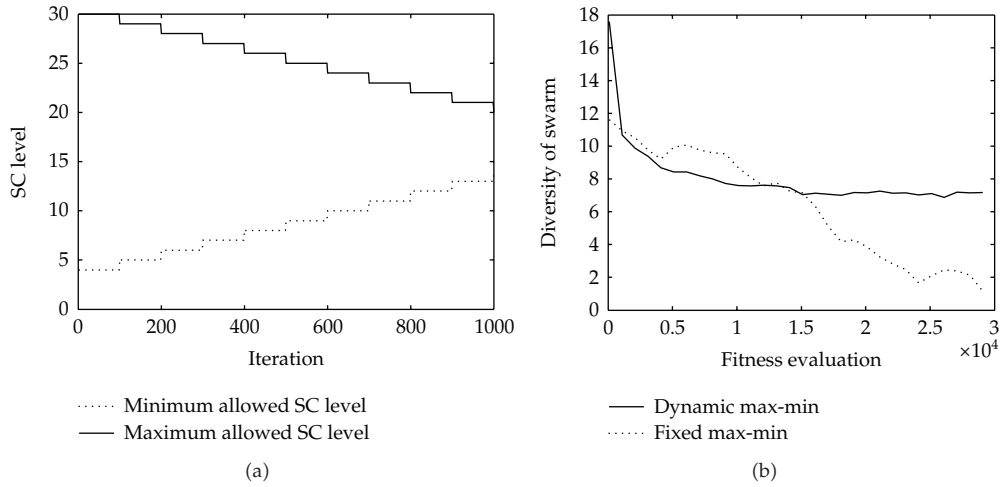


Figure 7: The allowed SC number and the effect on the swarm diversity.

value and update its $\overrightarrow{p_{bin_{(i)}}}$, $\overrightarrow{p_g}$, and $\overrightarrow{p_i}$. As all learning exemplars are within the search range, a particle will eventually return to the search range.

3.3.5. Analysis of Swarm's Diversity

In order to compare the particle diversity of OPSO with that of HSCPSO, we omit the previous velocity $w \cdot \overrightarrow{v_i}(t-1)$ and make φ_1 and φ_2 equal to one. Thus, the following velocity updating equation is employed in the experiment:

$$\begin{aligned} \overrightarrow{v_i}(t) &\longrightarrow r_1\left(\overrightarrow{p_i} - \overrightarrow{x_i}(t-1)\right) + r_2\left(\overrightarrow{p_g} - \overrightarrow{x_i}(t-1)\right), \\ \overrightarrow{v_i}(t) &\longrightarrow r_1\left(\overrightarrow{p_i} - \overrightarrow{x_i}(t-1)\right) + r_2\left(\overrightarrow{p_{bin_{(i)}}} - \overrightarrow{x_i}(t-1)\right). \end{aligned} \tag{3.10}$$

In terms of (3.10), we run HSCPSO and OPSO on Rosenbrock and Rastrigin functions to analyze the number of the best particle (NBP) at each iteration. Figure 8 gives the scatter plot about the index of the best performing particles associated with the numbers of the iteration. For example, a dot (20, 4000) represents that the index 20's particle has the best global fitness in the iteration number 4000. The velocity updating strategy of HSCPSO has more NBP than OPSO, which shows that the strategy of HSCPSO increases the swarm's diversity compared with OPSO [16]. The pseudocode of the HSCPSO is shown in Algorithm 3.

4. Experiment Settings and Results

4.1. Test Functions and Parameter Settings of PSO Variants

To compare with the performance of all algorithms, Sphere, Rosenbrock, Acley, Griewanks, Rastrigin, and Rastrigin_noncont functions are selected to test the convergence characteristics,

(1) **Begin**
(2) Initialize particles' position, velocity, $pbest_i$, $gbest$ and the related parameters
(3) Initialize default social circle of which the particle i is a member
(4) $V_{\max} = 0.25\,(X_{\max} - X_{\min})$
(5) Initialize p_u
(6) **while** (fitcount < Max_FES) && (k < iteration)
(7) **For each particle**
(8) Locate the best particle position of the SCs that particle i is a member of
(9) Computer $\overline{p_{\text{bin}_{(i)}}}$ in terms of (3.1).
(10) **For each dimension**
(11) Updating the particles' velocity and position in terms of (3.1)
(12) **If** fitness (x_i) < fitness $(pbest_i)$
(13) $pbest_i = x_i,\ pbestval_i = \text{fitness}(x_i)$
(14) **End If**
(15) **If** ceil(mc_i+ rand − 1)==1
(16) **If** rand ≤ p_u
(17) $x(i,d) = (1 + \text{rand}) \times \text{pos}(i,d)$ **Else** $x(i,d) = \text{Gaussian}(\sigma) \times \text{pos}(i,d)$
(18) **End If**
(19) **End If**
(20) **Next dimension**
(21) **Next particle**
(22) **Update the member of SC in terms of Figure 2(b)**
(23) **If** (mod(k, n)) && ($|\text{SC}_i| \Leftarrow$ DSC)
(24) Update particle's DSC level.
(25) **End If**
(26) Update each particle neighbor.
(27) Index = find $(pbestval_i)$//find-a function which finds a particle's position
(28) **If** fitness $(gbest)$ > min$(pbestval_i)$
(29) $gbestval = pbestval_i,\ gbest = x_{\text{index}}$
(30) **End If**
(31) **End While**
(32) **End**

Algorithm 3

and Ackley, Rastrigin, Rastrigin-noncont, and Rosenbrock functions are rotated with Salomon's algorithm [21] to increase the optimization difficulty and test the algorithm robustness (here, the predefined threshold is 0.05, 50, 2, and 3 resp.). Table 3 represents the properties of these functions. Note that in Table 3, the values listed in the search space column are used to specify the range of the initial random particles' position; the x^* denotes the global optimum, and the $f(x^*)$ is the corresponding fitness value.

In order to make these different algorithms comparable, the population size is set at 30 for all PSOs, and each test function is run 30 times. At each run, the maximum fitness evaluation (FEs) is set at 3×10^4 for the unrotated test functions and 6×10^4 for the rotated. The comparative algorithms and their parameters settings are listed as below.

(i) Local version PSO with constriction factor (CF-LPSO) [3],

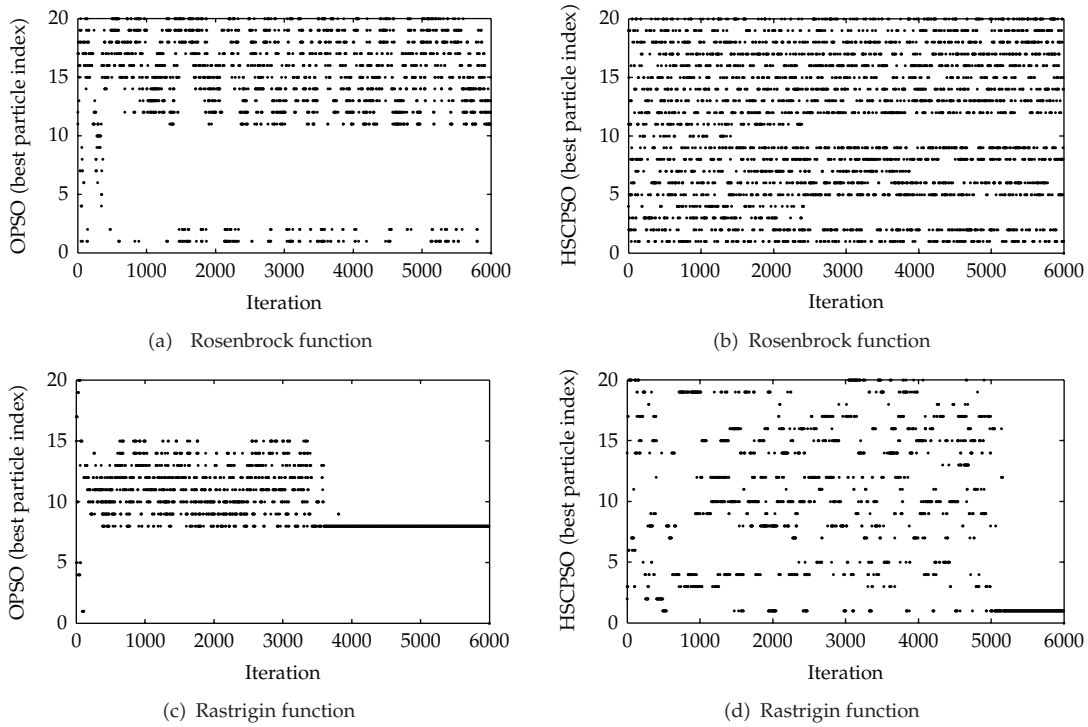
(ii) Global version PSO with constriction factor (CF-GPSO) [3],

(iii) Fitness-distance-ratio-based PSO (FDR-PSO) [13],

(iv) Fully informed particle swarm optimization (FIPS) [17],

Table 3: Dimension, search space, and global optimum of the test functions.

		Test functions and formulas	n	Global optimum (x^*)	$f(x^*)$	search Space
Unrotated function	Sphere (f_1)	$f(x) = \sum_{i=1}^{n} {x_i}^2$	30	$[0,0,\ldots,0]$	0	$[-100,100]^{30}$
	Rosenbrock (f_2)	$f(x) = \sum_{i=1}^{n-1} (100({x_i}^2 - {x_{i+1}}^2)^2 + (x_i - 1)^2)$	30	$[1,1,\ldots,1]$	0	$[-2.048,2.048]^{30}$
	Ackley (f_3)	$f(x) = -20 \cdot \exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} {x_i}^2}) - \exp((1/n)\sum \cos(2\pi x_i)) + 20 + e$	30	$[0,0,\ldots,0]$	0	$[-32.768,32.768]^{30}$
	Griewanks (f_4)	$f(x) = \sum_{i=1}^{n} \left({x_i}^2/4000\right) - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	30	$[0,0,\ldots,0]$	0	$[-600,600]^{30}$
	Rastrigin (f_5)	$f(x) = \sum_{i}^{n} (x_i - 10\cos(2\pi x_i) + 10)$	30	$[0,0,\ldots,0]$	0	$[-5.12,5.12]^{30}$
	Rastrigin-noncont (f_6)	$f(x) = \sum_{i=1}^{n} ({z_i}^2 - 10\cos(2\pi z_i) + 10)$	30	$[0,0,\ldots,0]$	0	$[-5.12,5.12]^{30}$
Rotated function		Ackley (f_7)	30	$[0,0,\ldots,0]$	0	$[-32.768,32.768]^{30}$
		Rastrigin (f_8)	30	$[0,0,\ldots,0]$	0	$[-5.12,5.12]^{30}$
		Rastrigin-noncont (f_9)	30	$[0,0,\ldots,0]$	0	$[-5.12,5.12]^{30}$
		Rosenbrock (f_{10})	30	$[0,0,\ldots,0]$	0	$[-2.048,2.048]^{30}$



(a) Rosenbrock function (b) Rosenbrock function

(c) Rastrigin function (d) Rastrigin function

Figure 8: The best particle in the swarm with iteration elapsed.

(v) Cooperative particle swarm optimization (CPSO-H_k) [18],

(vi) Comprehensive learning particle swarm optimizer (CLPSO) [12].

The fully informed PSO (FIPS) with a U-ring topology that achieved the highest success rate was used. CPSO-H_k is a cooperative PSO model combined with the standard PSO, in which $k = 6$ is used in this paper. $\varphi_1 = 1.492$, $\varphi_2 = 1.492$, and $w = 0.729$ are used in all PSOs except HSCPSO.

4.2. Fixed Iteration Results and Analysis

Tables 4 and 5 present the means and 95% confidence interval after 3×10^4 and 6×10^4 function evaluations. The best results among seven PSO algorithms are presented in bold. In addition, to determine whether the result obtained by HSCPSO is statistically different from the results of the other six PSO variants, the Wilcoxon rank sum tests are conducted between the HSCPSO result and the best result achieved by the other five PSO variants on each test problem, and the test results are presented in the last row of Tables 4 and 5. Note that an h value of one implies that the performance of the two algorithms is statistically different with 95% certainty, whereas h value of zero indicates that the performance is not statistically different. Figures 9 and 10 present the convergence characteristics in terms of the best fitness value of the median run of each algorithm for each test function.

Table 4: Means after 3×10^4 function evaluations for the unrotated test functions.

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6
CF-LPSO	4.1167$e-045$	2.5761$e+001$	3.5339$e-004$	1.2321$e-002$	4.2783$e+001$	7.0021$e+000$
	$\pm$ 2.340$e-045$	$\pm$ 3.1156$e+001$	$\pm$ 6.1254$e-004$	$\pm$ 6.2135$e-002$	$\pm$ 3.5631$e+001$	$\pm$ 1.2031$e+000$
CF-GPSO	**7.1619e − 072**	2.2596$e+001$	1.3404$e+000$	7.7928$e-001$	4.5768$e+001$	5.0004$e+000$
	$\pm$ **1.342e − 072**	$\pm$ 4.7652$e+001$	$\pm$ 7.8543$e+000$	$\pm$ 3.7452$e-001$	$\pm$ 6.3256$e+001$	$\pm$ 2.0128$e+000$
FDR-PSO	4.2434$e-021$	2.5387$e+001$	3.6685$e-008$	5.1549$e-002$	4.7758$e+001$	3.0011$e+000$
	$\pm$ 1.096$e-021$	$\pm$ 2.0341$e+001$	$\pm$ 5.3412$e-008$	$\pm$ 7.1321$e-002$	$\pm$ 3.6134$e+001$	$\pm$ 3.2863$e+000$
FIPS	2.6302$e-013$	2.9948$e+001$	1.8719$e+000$	1.0885$e+000$	1.4509$e+002$	6.9801$e+000$
	$\pm$ 2.431$e-013$	$\pm$ 2.0987$e+001$	$\pm$ 6.9432$e+000$	$\pm$ 4.8312$e+000$	$\pm$ 2.6753$e+002$	$\pm$ 2.1109$e+000$
CPSO	3.6684$e-006$	6.3901$e+000$	3.6273$e-002$	8.0047$e-002$	3.3829$e+001$	3.0242$e+000$
	$\pm$ 3.1233$e-006$	$\pm$ 1.0245$e+000$	$\pm$ 2.8965$e-002$	$\pm$ 3.8901$e-002$	$\pm$ 2.7953$e+001$	$\pm$ 1.9023$e+000$
CLPSO	8.2953$e-013$	7.2600$e+001$	1.5160$e+000$	1.1316$e-002$	1.2702$e+001$	8.1642$e-003$
	$\pm$ 2.6712$e-013$	$\pm$ 2.1456$e+001$	$\pm$ 1.1463$e+000$	$\pm$ 1.0932$e-002$	$\pm$ 1.6954$e+001$	$\pm$ 3.9206$e-003$
HSCPSO	2.3335$e-012$	**2.4682e − 001**	**1.3323e − 014**	**7.3275e − 015**	**7.2479e − 003**	**1.3488e − 005**
	$\pm$ 3.9123$e-012$	$\pm$ **1.6324e − 001**	$\pm$ **7.9854e − 014**	$\pm$ **3.0451e − 015**	$\pm$ **3.4521e − 003**	$\pm$ **1.0028e − 005**
Result	0	1	1	1	1	1

Table 5: Means after 6×10^4 function evaluations for the rotated test functions.

Algorithm	f_7	f_8	f_9	f_{10}
CF-LPSO	9.0528$e+000$ $\pm$	9.3130$e-001$ $\pm$	4.5769$e+001$ $\pm$	3.7000$e+001$ $\pm$
	1.2314$e+000$	2.1345$e-001$	2.7804$e+001$	2.1456$e+001$
CF-GPSO	3.4452$e+000$ $\pm$	3.3445$e+000$ $\pm$	1.1243$e+002$ $\pm$	8.7012$e+001$ $\pm$
	2.1423$e+000$	6.4321$e+000$	2.0041$e+002$	2.0098$e+001$
FDR-PSO	4.4461$e+000$ $\pm$	2.4083$e+000$ $\pm$	8.0591$e+001$ $\pm$	6.1231$e+001$ $\pm$
	3.5623$e+000$	3.7854$e+000$	3.0971$e+001$	3.0981$e+001$
FIPS	1.1432$e+001$ $\pm$	2.2201$e+000$ $\pm$	1.2506$e+002$ $\pm$	1.3650$e+002$ $\pm$
	4.6785$e+001$	9.1232$e+000$	6.3210$e+002$	2.0981$e+002$
CPSO	1.4112$e+001$ $\pm$	1.7780$e+000$ $\pm$	1.0149$e+002$ $\pm$	9.400$e+001$ $\pm$
	3.5621$e+001$	1.4563$e+000$	2.0012$e+002$	4.0561$e+001$
CLPSO	**5.2190e − 001** $\pm$	**4.6109e − 004** $\pm$	4.9697$e+001$ $\pm$	5.4287$e+001$ $\pm$
	3.4512e − 001	**7.5431e − 004**	5.7821$e+001$	1.0975$e+001$
HSCPSO	3.3121$e+000$ $\pm$	1.1219$e-003$ $\pm$	**3.5824e + 001** $\pm$	**1.7903e + 001** $\pm$
	2.1457$e+000$	3.8712$e-003$	**6.1023e + 001**	**1.3294e + 001**
Result	0	0	1	1

From the above experimental results, we can observe that Sphere function is easily optimized by CF-GPSO and CF-LPSO, while the other five algorithms show the slower convergence speed. Rosenbrock function has a narrow valley from the perceived local optima to the global optimum. In the unrotated case, HSCPSO may avoid the premature convergence. Note that in the rotated case, there is little effect on all algorithms.

There are many local minima positioned in a regular grid on the multimodal Ackley. In the unrotated case, HSCPSO takes the lead, and the FDR-PSO performs better than the other five PSO variants. In the rotated case, the FDR-PSO algorithm is trapped in local optima early on, whereas CLPSO and HSCPSO belong to the performance leaders. As a result of the

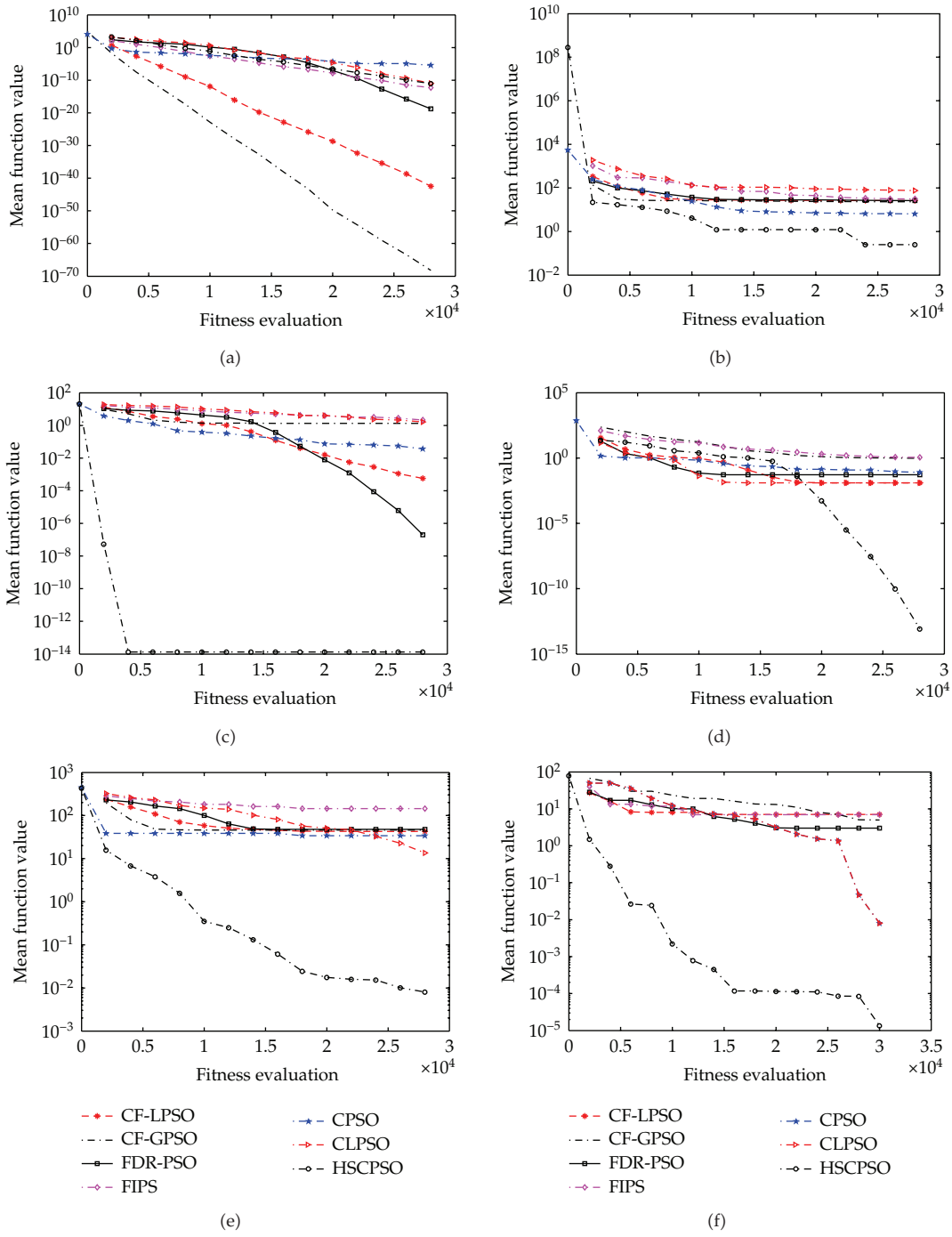


Figure 9: Convergence characteristics of the unrotated test functions. (a) Sphere; (b) Rosenbrock; (c) Ackley; (d) Griekwanks; (e) Rastrigin; (f) Rastrigin-noncont.

Table 6: Robustness analysis after 3×10^4 function evaluations for the unrotated test functions.

Algorithm	f_2		f_3		f_5		f_6	
	FES	Success rate (%)	FES	Success rate (%)	FES	Success rate (%)	FES	Success rate (%)
CF-LPSO	4240	100	21081	100	13145	31	13982	42
CF-GPSO	2196	100	20080	70	5400	20	10273	41
FDR-PSO	3803	100	19761	100	13674	100	9023	69
FIPS	12398	100	19832	90	13654	94	8419	74
CPSO	4635	100	19325	100	919	100	24294	85
CLPSO	4240	100	19034	95	21603	100	25832	100
HSCPSO	299	100	1023	100	784	100	16397	100

Table 7: Robustness analysis after 6×10^4 function evaluations for the rotated test functions.

Algorithm	f_7		f_8		f_9		f_{10}	
	FES	Success rate (%)	FES	Success rate (%)	FES	Success rate (%)	FES	Success rate (%)
CF-LPSO	4270	20	2670	26	12137	22	4325	100
CF-GPSO	3331	19	3200	20	6203	26	4200	100
FDR-PSO	15525	45	1980	90	23985	42	13709	100
FIPS	10642	90	3100	95	38124	45	14000	100
CPSO	2474	90	3560	75	24284	39	12800	100
CLPSO	16481	100	2578	100	32193	95	25682	100
HSCPSO	1631	100	1800	100	1328	100	10000	100

comprehensive learning strategy in CLPSO, it can discourage premature convergence. At the same time, this learning strategy of HSCPSO can also ensure the diversity of the swarm to be preserved to discourage the premature convergence.

Rastrigin function exhibits a pattern similar to that observed with Ackley function. In the unrotated case, HSCPSO performs very well, but its performance rapidly deteriorates when the search space is rotated. In the rotated case, CLPSO takes the lead. Note that CLPSO is still able to improve the performance in the rotated case.

On the unrotated Rastrigin-noncont function, HSCPSO presents an excellent performance compared with other algorithms, and after 2.6×10^4 function evaluation CLPSO has the strongest ability to escape from local optima. In the unrotated case, all algorithms quickly get trapped in a local minimum besides CLPSO and HSCPSO. These two algorithms can avoid the premature convergence and escape from local minima.

Altogether, according to Wilcoxon rank sum tests, we can observe that the HSCPSO algorithm can perform better than the other six algorithms on functions f_2, f_3, f_4, f_5, f_6, f_9, and f_{10}. Although HSCPSO has no best performance in f_7 and f_8, it shows almost the same convergence character as one of CLPSO.

4.3. Robustness Analysis

Tables 6 and 7 show the results of the robustness analysis. Here, the "robustness" is used to test the stability of the algorithm optimization ability in different conditions (the rotated and

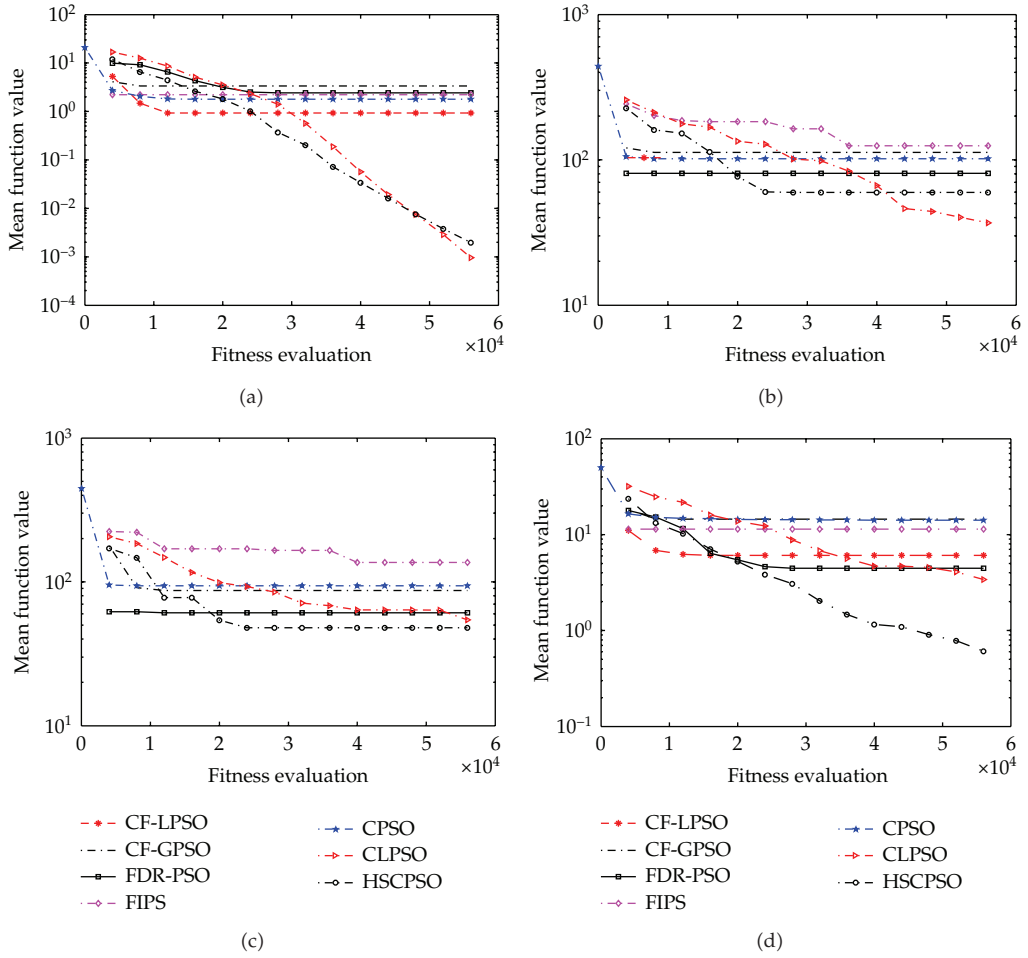


Figure 10: Convergence characteristics of the rotated test functions. (a) Ackley; (b) Rastrigin; (c) Rastrigin_noncont; (d) Rosenbrock.

unrotated test functions) by a certain criterion. In HSCPSO, the criterion is that the algorithm succeeded in the ability of reaching a specified threshold. A robust algorithm is the one that manages to reach the threshold consistently whether in the rotated or the unrotated. The "Success rate" column lists the rate of algorithm reaching threshold in 60 times run. The "FES" column means the number of the function evaluations when reaching the threshold, and only the dates of the successful runs are used to compute "Success rate."

As can be seen in Tables 6 and 7, none of all PSOs had any difficulty in reaching the threshold on the Rosenbrock function during 30 runs in any case. CF-GPSO has some difficulties in the unrotated and the rotated Ackley function, but CF-LPSO reaches the threshold on the unrotated Ackley. FDR-PSO and CPSO failed on the rotated Ackley function, but in the unrotated case, they consistently reached the threshold. The FIPS completely failed in both the unrotated and the rotated cases. It is interesting to note that CLPSO reached the threshold during all the runs on the rotated Ackley function. However, in the unrotated case, it did not get a perfect success rate. Only HSCPSO consistently reached the threshold in both the unrotated and rotated cases.

The Rastrigin-noncont function is hard to be solved for the majority algorithms, as can be seen in Tables 6 and 7. CLPSO and HSCPSO consistently reached the threshold in the unrotated case, while in the rotated case, only HSCPSO could achieve a perfect success rate.

On Rastrigin function, HSCPSO and CLPSO consistently reached the threshold on both the unrotated and the rotated cases. CPSO and FDR-PSO reached the threshold in the unrotated case, but they failed in the rotated case. CF-LPSO, FIPS, and CF-GPSO algorithms had some difficulties in both the unrotated and the rotated cases.

Altogether, on Rosenbrock, Ackley, Rastrigin, and Rastrigin-noncont functions, HSCPSO shows the stability of the optimization ability in the different conditions. CLPSO, FDR, CPSO, and FIPS consistently reached the threshold on most of test functions, and they were slightly less robust. CF-LPSO and CF-GPSO seemed to be unreliable on the multimodal benchmark functions.

5. Conclusions and Future Works

This paper proposed an improved PSO based on the simulation of the human social behavior (HSCPSO for short) where each particle can adjust its learning strategy in terms of the current condition self-adaptively. From the convergence character and the robustness analysis, we can conclude that HSCPSO significantly improves the ability to solve the complicated multimodal problems compared with the other PSO versions. In addition, from Wilcoxon rank sum tests, these results achieved by HSCPSO are statistically different from the second best result. Although the HSCPSO algorithm outperformed the other PSO variants on most of the test functions evaluated in this paper, it can be regarded as an effective improvement in the PSO domain.

In the future, we will focus on (i) constructing the model for solving the relevant parameters of PSO, (ii) testing the proposed algorithm effectiveness with more multimodal test problems and several composition functions that are more difficult to be optimized, (iii) applying the proposed algorithm to some practices to verify its effectiveness, and (iv) in the future this proposed algorithm will be tested on CEC 2005 problems.

Acknowledgments

This work is supported by The National Natural Science Foundation of China (Grants nos. 71001072, 71271140, 71210107016), China Postdoctoral Science Foundation (Grant nos. 20100480705, 2012T50584), the Science and Technology Fund of Guizhou Province ([2012] 234000), Zunyi Normal College Research Funded Project (2012 BSJJ19), Shanghai Postdoctoral Fund Project (12R21416000), Science and Technology Project of Shenzhen (Grant no. JC201005280492A), and the Natural Science Foundation of Guangdong Province (Grant no. 9451806001002294).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[2] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th Annual Conference on Evolutionary Programming*, San Diego, Calif, USA, 1998.

[3] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[4] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Congress on Evolution Computation*, pp. 1671–1676, Honolulu, Hawaii, USA, 2002.

[5] J. Sun, J. Liu, and W. Xu, "Using quantum-behaved particle swarm optimization algorithm to solve non-linear programming problems," *International Journal of Computer Mathematics*, vol. 84, no. 2, pp. 261–272, 2007.

[6] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98)*, pp. 84–89, Anchorage, Alaska, USA, May 1998.

[7] M. Lovbjerg and T. Krink, "Hybrid particle swarm optimizer with breeding and subpopulations," in *Proceeding of the Genetic Evolution Computation Conference*, pp. 469–476, 2001.

[8] A. Stacey and M. Jancic, "Particle swarm optimization with mutation," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1425–1430, Canberra, Australia, 2003.

[9] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'06)*, pp. 1044–1051, July 2006.

[10] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1958–1962, Washington, DC, USA, 1999.

[11] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1677–1681, Honolulu, Hawaii, USA, 2002.

[12] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[13] T. Peram and K. Veeramachaneni, "Fitness-distance-ratio based particle swarm optimization," in *Proceeding of the IEEE Swarm Intelligence Symposium*, pp. 174–181, April 2003.

[14] A. S. Mohais, R. Mendes, C. Ward, and C. Posthoff, "Neighborhood re-structuring in particle swarm optimization," in *AI 2005: Advances in Artificial Intelligence*, vol. 3809 of *Lecture Notes in Computer Science*, pp. 776–785, Springer, Berlin, Germany, 2005.

[15] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 6, pp. 1272–1282, 2005.

[16] W. Elshamy, H. M. Emara, and A. Bahgat, "Clubs-based particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'07)*, pp. 289–296, Honolulu, Hawaii, USA, April 2007.

[17] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[18] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[19] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08)*, pp. 3845–3852, Hong Kong, China, June 2008.

[20] Y. M. Liu, Q. Z. Zhao, C. L. Sui, and Z. Z. Shao, "Particle swarm optimizer based on dynamic neighborhood topology and mutation operator," *Control and Decision*, vol. 25, no. 7, pp. 968–974, 2010.

[21] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.