Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 8250480, 14 pages
https://doi.org/10.1155/2018/8250480

Research Article

A New Chaotic Starling Particle Swarm Optimization Algorithm for Clustering Problems

Lin Wang[1], **Xiyu Liu**[1], **Minghe Sun**[2], **Jianhua Qu**[1] and **Yanmeng Wei**[1]

[1]*College of Management Science and Engineering, Shandong Normal University, Jinan 250014, China*
[2]*College of Business, The University of Texas at San Antonio, San Antonio, TX, USA*

Correspondence should be addressed to Xiyu Liu; sdxyliu@163.com

Received 3 May 2018; Accepted 2 August 2018; Published 19 August 2018

Academic Editor: AMA Neves

A new method using collective responses of starling birds is developed to enhance the global search performance of standard particle swarm optimization (PSO). The method is named chaotic starling particle swarm optimization (CSPSO). In CSPSO, the inertia weight is adjusted using a nonlinear decreasing approach and the acceleration coefficients are adjusted using a chaotic logistic mapping strategy to avoid prematurity of the search process. A dynamic disturbance term (DDT) is used in velocity updating to enhance convergence of the algorithm. A local search method inspired by the behavior of starling birds utilizing the information of the nearest neighbors is used to determine a new collective position and a new collective velocity for selected particles. Two particle selection methods, Euclidean distance and fitness function, are adopted to ensure the overall convergence of the search process. Experimental results on benchmark function optimization and classic clustering problems verified the effectiveness of this proposed CSPSO algorithm.

1. Introduction

The particle swarm optimization (PSO) algorithm is a global optimization method based on intelligent search strategy in population inspired by the behavior of birds flocking. As a swarm intelligence algorithm, each particle flies in the search space, called the solution space, with a certain velocity and updates its velocity and position through a linear combination of individual and global best positions in history. Compared with other evolutionary algorithms, PSO uses individual and global experiences of the particles and has well-balanced mechanism between its exploitation and exploration abilities [1]. Therefore, it has been successfully applied to many difficult optimization problems [2].

PSO provides good exploitation and exploration performance for solving optimization problems. The global experience guides the direction of the particle population, and individual experience gives a more precise direction in the search space. In this way, the particle population will move gradually closer to the global optimum. Hence, it has short computing time and is easy to implement. However, like most swarm intelligence algorithms, this algorithm can be easily trapped into local optima in later generations or iterations, and the search process may premature. Many works have been done to improve the standard or traditional PSO algorithm [3–6].

Data clustering has been an important technology in data analysis. The purpose of data clustering is to discover valuable informative patterns and implicit information [7]. As an unsupervised classification technique, data clustering classifies similar data into the same groups or clusters using the characteristics of the data without any prior knowledge about the groups or clusters. Therefore, data clustering can extract potential patterns or knowledge from the dataset and can help in obtaining and understanding the valuable information in the data [8]. Because data clustering problems are NP-hard, traditional methods are sometimes ineffective in solving these problems [9]. Therefore, a lot of work has been done to solve this problem by adopting PSO, and some recent researches and works are summarized below.

Netjinda et al. [10] presented a new PSO procedure, called starling PSO, inspired by the collective responses of starlings. Dor et al. [11] proposed a dynamic topology DCluster algorithm based on two topologies using a four-cluster approach and a fitness function to solve the underlying

```
for i = 1 to M
    v_ij(t+1) = w * v_ij(t) + c1 * r1 * (p_ij(t) − x_ij(t)) + c2 * r2 * (p_gj(t) − x_ij(t));
    x_ij(t+1) = x_ij(t) + v_ij(t+1);
    if Particle(i).Fitness<Particle(i).Best.Fitness
        Particle(i).Best.Position=Particle(i).Position;
        Particle(i).Best.Fitness= Particle(i).Fitness;
        if Particle(i).Best.Fitness<Best Particle.Fitness
            Best Particle.Position=Particle(i).Best.Position;
            Best Particle.Fitness=Particle(i).Best.Fitness;
            End if
    End if
End for
```

Algorithm 1: The Standard PSO Algorithm.

problem. Ali [12] presented a new variant of the position updating rule of PSO, in which each particle is treated as a bidimensional vector. Armano et al. [13] proposed a multiobjective clustering PSO algorithm with two objective functions defined to integrate data connectivity and cohesion. Niu et al. [14] proposed a population-based clustering algorithm which combines different PSO and k-means algorithms to help particles escape from local optima.

Exploring the topological neighborhood of the k-nearest neighbors and employing pattern search are considered to be useful tools to improve the performance of PSO [15]. Cormark [16] proposed a PSO procedure using Renyi entropy clustering, which contains two steps, initialization and particle removal. Bharti and Singh [17] proposed a binary PSO procedure with an opposition-based learning mechanism using chaotic mapping, dynamic inertia weight, and a mutation operator. Song et al. [18] added an environment factor to the velocity adjustment in PSO to enhance the robust behavior of the particles. Liu et al. [19] proposed a modified coevolutionary multiswarm PSO procedure based on new velocity updating and similarity detection to solve multiobjective clustering problems.

Although PSO has been widely used in many fields and has shown a great potential in solving optimization problems, it still has some limitations. For example, it is easily trapped into local optima and has a low convergence speed. So far, no effective methods have been developed to balance local and global searching abilities [20]. Therefore, more works are needed to enhance the performance of PSO [21]. On the other hand, data clustering, as one of the most popular data mining techniques in discovering potential information and knowledge from data, needs effective methods to obtain better clustering results. In this study, a chaotic starling particle swarm optimization (CSPSO) algorithm is proposed to obtain better clustering results by improving the PSO performance.

CSPSO has three major parts. A chaotic mapping, rather than a random generation, method is introduced to generate the acceleration parameters. A dynamic disturbance term (DDT) is added in velocity updating to avoid trapping into local optima. In order to improve the search ability, a local search strategy based on the behavior of starling birds is used, and the information of neighbors is collected to guide the direction of the particle.

The rest of this paper is organized as follows. The traditional PSO and the clustering problem are described in Section 2. The CSPSO is developed and described in detail in Section 3. In Section 4, simulation experiments are conducted and comparisons with existing methods are performed to analyze the effectiveness of CSPSO. Section 5 gives conclusions and future research directions.

2. Preliminary

In this section, the basic concepts of PSO and data clustering problems, related to the development of the proposed algorithm, are described in some detail.

2.1. The Standard PSO. PSO is one of the swarm intelligence algorithms inspired by social behavior of bird flocking [26]. In PSO, the population size of the particles is denoted by M, and the dimension of the search space is denoted by D. Each particle i has a position vector $x_i = \{x_{i1}, x_{i2}, \cdots x_{iD}\}$, a velocity vector $v_i = \{v_{i1}, v_{i2}, \cdots v_{iD}\}$, and an individual best position in history $p_i = \{p_{i1}, p_{i2}, \cdots p_{iD}\}$. The best position found by all particles in the swarm, called the global best, is represented by $p_g = \{p_{g1}, p_{g2}, \cdots p_{gD}\}$. At iteration t, the new updated position and velocity of particle i are determined by (1) and (2), respectively, in the following:

$$v_{ij}(t+1) = w * v_{ij}(t) + c_1 * r_1 * \left(p_{ij}(t) - x_{ij}(t)\right) + c_2 * r_2 * \left(p_{gj}(t) - x_{ij}(t)\right), \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad (2)$$

where w is the inertia weight, c_1 and c_2 are the acceleration coefficients controlling the step size, and r_1 and r_2 are two independently generated random numbers uniformly distributed between 0 and 1. Two termination criteria are used. One is when a preset maximal number of iterations is reached and the other is when a tolerance level has been achieved. The standard PSO algorithm is described in Algorithm 1.

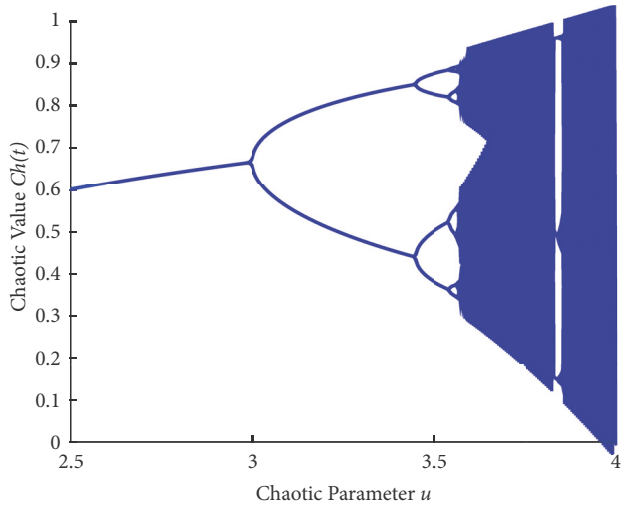


Figure 1: Bifurcation diagram for the logistic map.

2.2. Chaotic Mapping. Chaotic maps are mapping methods used to generate random numbers with features like ergodicity and nonlinear and random similarity [27]. There are many kinds of chaotic maps, such as tent map, Tchebychev map, and logistic map. Specifically, the logistic map developed by May [28] can explore in the vicinity of a solution by oscillating in the region. One chaotic variant based on logistic mapping is given by (3) in the following [29]:

$$Ch(t+1) = u * Ch(t) * (1 - Ch(t)), \quad Ch(t) \in (0,1),\ u \in (0,+\infty), \tag{3}$$

where $Ch(t)$ represents the value of the chaotic number at time t, u is the control parameter, and $Ch(0) \notin \{0, 0.25, 0.5, 0.75, 1.0\}$. In (3), the parameter u controls the behavior of the chaotic variant Ch. The values of $Ch(t)$ for varying values of u are shown in Figure 1.

2.3. Data Clustering Problems. Let $X = \{X_q\}$, for $q = 1, 2, \ldots, N$, be a dataset containing N data points or instances. Data point q is represented by $X_q = \{X_{q1}, X_{q2}, \ldots, X_{qd}\}$ with d representing the dimension of the data. The purpose of a data clustering problem is to find a partition of the dataset by optimizing a fitness function. A partition is represented by $C = \{C_k\}$, for $k = 1, 2, \ldots, K$, where K is the number of clusters [30]. A partition must satisfy the following conditions:

(i) $C_{k_1} \cap C_{k_2} = \varnothing, \forall k_1 \neq k_2$

(ii) $\bigcup_{k=1}^{K} C_k = X$

(iii) $C_k \neq \varnothing, \forall k = 1, 2, \ldots, K.$

Usually, the Euclidean distance is used to measure the difference or similarity between two data points X_{q_1} and X_{q_2}. Let $z = \{z_k\}$, for $k = 1, 2, \ldots, K$, represent the centers of the K cluster C_k. The intracluster distance of cluster C_k is the sum of the distances of all data points in the cluster to z_k given by (4) in the following:

$$D(X, C_k) = \sum_{X_q \in C_k} \left\| X_q - z_k \right\|. \tag{4}$$

The quality of the clustering results for the dataset can be measured by the sum of the intracluster distances over all clusters given by (5) in the following [31]:

$$F = F(C_1, C_2, \ldots, C_k) = \sum_{k=1}^{K} D(X, C_k) = \sum_{k=1}^{K} \sum_{X_q \in C_k} \left\| X_q - z_k \right\|. \tag{5}$$

F defined in (6) is used as the fitness function in clustering in the following.

3. The Chaotic Starling Particle Swarm Optimization Algorithm

Some efforts have been made in order to enhance the search performance and convergence speed of PSO. In this section, a chaotic mapping method and a DDT are introduced into the PSO algorithm to improve the global search ability, and a local search technique based on starling birds is added to improve the convergence speed. This improved PSO method is the CSPSO algorithm. The major components and the details of the CSPSO algorithm are discussed in this section.

3.1. Dynamic Update of CSPSO Parameters. Three components, velocity of the previous iteration, individual factor, and social factor, are used to update the velocity of a particle. Three parameters, w, r_1, and r_2, control the relative contributions of the three components. The inertia weight w controls the influence of the velocity of the previous iteration. The cognitive coefficients c_1 and c_2 balance the contributions of the individual and social factors. Furthermore, the acceleration coefficients r_1 and r_2 control the proportion retained in the individual and social factors. Each parameter plays an important role in the velocity update and also consequently affects the particle position update. Instead of using fixed values, logistic mapping is used to generate the acceleration coefficients r_1 and r_2 in each generation. In addition, the inertia weight is adjusted using an exponential function. The modifications of these parameters are shown in (6) and (7):

$$r_p(t+1) = 4 * r_p(t) * \left(1 - r_p(t)\right), \quad r_p(t) \in (0,1),\ p = 1, 2, \tag{6}$$

$$w(t) = w_{\min} + \left(w_{\max} - w_{\min}\right) * e^{(-t/(t_{\max} - t))}, \tag{7}$$

where $w_{\min}$ and $w_{\max}$ are the minimum and maximum of the inertia weight and $t_{\max}$ is the maximum number of iterations.

3.2. The Dynamic Disturbance Term. As an optimization algorithm based on swarm intelligence, PSO uses the

trajectory of particles in the population by comparing the fitness function values to select the local and global best positions. Because of local optimal solutions, the velocity of a particle is gradually decreasing during the later iterations. Therefore, the majority of the particles fall into, and cannot easily jump out of, local optimal points, called premature convergence and evolutionary stagnation. To overcome these problems, a method based on DDT [32] is used to change the velocity update of each particle. DDT is embedded into velocity updating when the velocity of a particle becomes too low or when its position stays unchanged in the middle and final generations. The modified velocity is updated using (8) and the DDT, represented by T, is updated using (9) in the following:

$$v_{ij}(t+1) = w * v_{ij}(t) + c_1 * r_1 * \left(p_{ij}(t) - x_{ij}(t)\right) + c_2 * r_2 * \left(p_{gj}(t) - x_{ij}(t)\right) + T, \tag{8}$$

$$T = m * \left(\frac{\mathrm{t}}{\mathrm{t}_{\max}} - 0.5\right), \tag{9}$$

where m is an accommodation coefficient with a value between 0 and 1. This improvement helps in preventing a particle from flying out of the boundary of the search space when its velocity is too high in early generations, and in escaping from a local optimal point by increasing its velocity so as to improve the overall searching capacity. Besides, T increases linearly to avoid oscillation and to keep a relatively stable search direction in the optimization process.

3.3. Local Search Based on Starling Birds. In nature, a starling bird spreads some kinds of information to its nearby neighbors in order to defend its position, and the information coming from one starling bird can spread to everywhere in the swarm. Inspired by this phenomenon, a local search method based on starling bird behavior is used in the developed PSO procedure. Each particle seeks a new position and velocity in the neighborhood by a weighted method that can be seen as a kind of information communication between an individual and its nearby neighbors. This mechanism tries to seek a better solution in the local exploration around the particle and reduces the time needed in local search [25].

Dynamic Neighborhood Selection. For each particle i, the neighborhood H is a subset of the particle population. The particles in the population are listed in decreasing order of the fitness-Euclidean distance ratio (FER) from particle i [23], and the Ne particles ranked on the top are taken as the ones in the neighborhood of the particle. FER is determined by (10) in the following:

$$FER\left(x_i, x_{i_1}\right) = a * \left(\frac{\left(F\left(x_i\right) - F\left(x_{i_1}\right)\right)}{ED\left(x_i, x_{i_1}\right)}\right), \quad i_1 = 1, 2, \ldots, M, \tag{10}$$

where $\mathrm{ED}(x_i, x_{i_1})$ is the Euclidean distance between x_i and x_{i_1}, $\alpha = \|s\|/(\mathrm{F}(x_w) - \mathrm{F}(x_b))$, x_b and x_w are the best and worst positions of the particles in the current population, s is the size of search space defined as $s = \sqrt{\sum_j^D (x_j^u - x_j^l)^2}$, and x_j^u and x_j^l are the maximum and minimum values of the jth dimension of the data points in the dataset. The number of neighbors Ne is the size of the neighborhood. In order to explore different search scope, a dynamic strategy is used to adjust the size of the neighborhood specified in (11) in the following:

$$Ne = Ne_{\min} + \mathrm{INT}\left(t * \left(\frac{\left(Ne_{\max} - Ne_{\min}\right)}{t_{\max}}\right)\right), \tag{11}$$

where $Ne_{\max}$ and $Ne_{\min}$ are the maximum and minimum sizes of the neighborhood and $\mathrm{INT}(a)$ is the integral function taking only the integer part of a.

Position Update. The position of particle i is adjusted using the information of the particles in the neighborhood. Using the weighted positions of the neighbors, the new position of particle i is determined by (12):

$$\widehat{x}_i = x_i + r_3 * \left(\left(\frac{1}{Ne}\right) * \sum_{i_1 \in H} x_{i_1}\right), \tag{12}$$

where x_i is the current position of particle i, $\widehat{x}_i$ is the new position after the adjustment, and r_3 is a randomly generated real number uniformly distributed between −1 and 1.

Velocity Update. Similar to the update of the position, the velocity of particle i is updated using the velocity information of the particles in the neighborhood. The new velocity of particle i is determined by (13) in the following:

$$\widehat{v}_i = v_i + r_4 * \left(\left(\frac{1}{Ne}\right) * \sum_{i_1 \in H} v_{i_1}\right), \tag{13}$$

where v_i and $\widehat{v}_i$ are the current and updated velocities of particle i and r_4 is a randomly generated real number uniformly distributed between 0 and 1. The collective responses of starling birds are described by the pseudocode shown in Algorithm 2.

4. The CSPSO for Clustering Problems

4.1. Particle Selection. In the previous section, a local search method based on starling bird behavior is used in the neighborhood of a particle to search for better solutions. Time needed by this method will increase. In order to accelerate search speed and reduce time needed, the local search method is not applied to all particles. Two methods are adopted to select particles from the population. The local search method is then applied to all the particles selected with these two methods.

An Euclidean Distance Method. In each generation, all the particles are sorted in ascending order of their Euclidean distances from the global best. The particles are divided into S_e groups of approximately equal size based on their

Input: Ne
for Particle i
Find the Ne neighbors in the neighborhood H of particle i using FER
The new position of particle i: $\widehat{x}_i = x_i + r_3 * ((1/Ne) * \sum_{i_1 \in H} x_{i_1})$
The new position of particle i: $\widehat{v}_i = v_i + r_4 * ((1/Ne) * \sum_{i_1 \in H} v_{i_1})$
if Particle($\widehat{i}$).Fitness<Particle(i).Fitness
Particle(i).Fitness=Particle($\widehat{i}$).Fitness;
Particle(i).Position=Particle($\widehat{i}$).Position;
Particle(i).Velocity=Particle($\widehat{i}$).Velocity;
End if
Return Particle i
End for
Output: Particle i

Algorithm 2: Starling Birds Collective Responses.

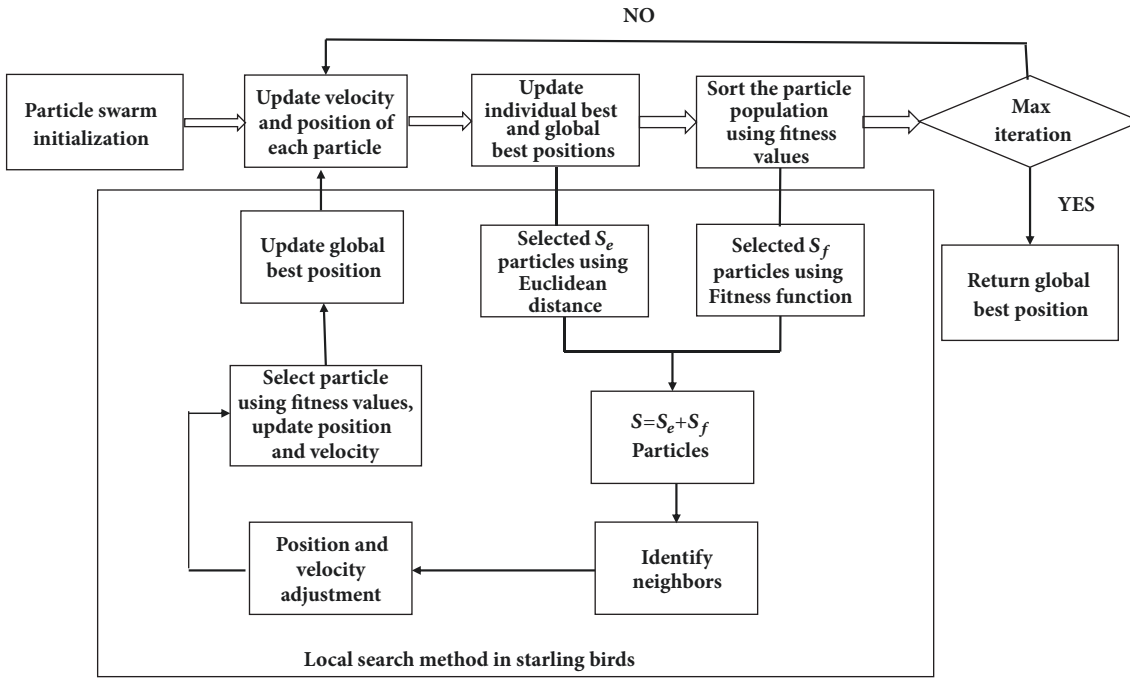


Figure 2: A local search method in the CSPSO algorithm.

Euclidean distances to the global best. The particle ranked on the top of each group is selected. As a result, a total of S_e particles are selected. The particles selected in this method diversify the search and enhance exploration of the CSPSO procedure.

A Fitness Function Method. In each generation, all the particles are sorted in the ascending order of their fitness function values. A total of S_f particles ranked on the top are selected. The particles selected in this method focus on the promising regions in the search space so as to enhance exploitation of the CSPSO procedure.

The two particle selection methods described above are used in CSPSO to improve its convergence and increase diversity. As a result, a total of $S = S_e + S_f$ particles are selected. In the implementation, $S_e = S_f$ is used. Particles selected through the Euclidean distance method are diversified in the search space to effectively avoid premature convergence. The particles selected through the fitness function method will lead the search direction. Figure 2 gives more details about the process of particle selection.

4.2. Population Initialization. In the implementation of CSPSO, each cluster center is the position of a particle, each cluster center represents a cluster, and different cluster centers represent different clusters. The set of cluster centers $z = \{z_k\}$, for $k = 1, 2, \ldots, K$, represents a partitioning result and also represents the position of a particle $x_i = \{z_{i1}, z_{i2}, \ldots, z_{iK}\}$. In the initialization, the initial position of each particle is generated randomly in the entire search space. Dimension j of the position of particle i represented by x_{ij} is generated

Input: $M, t_{\max}, Ne_{\min}, Ne_{\max}, w_{\min}, w_{\max}$
Particle Swarm Population Initialization
for $i = 1$ to M (Population Size)
Particle(i).Best.Position=Particle(i).Position;
Particle(i).Best.Fitness=Particle(i).Fitness;
if Particle(i).Best.Fitness<Best Particle. Fitness
Best Particle.Position=Particle(i).Best.Position;
Best Particle.Fitness=Particle(i).Best.Fitness;
End if
End for
for $t = 1$ to $t_{\max}$ (Max Iteration)
Algorithm 1: The Standard PSO Algorithm
(See Algorithm 1)
Particle Selection
S_e = Euclidean Distance Function (Best Particle.Position);
S_f = Fitness Function (Particle(i).Fitness);
$S = S_e + S_f$
Algorithm 2: Starling Birds Collective Responses
for $i = 1$ to S
Find the Ne neighbors in the neighborhood H of particle i using FER.
The new position of particle i: $\widehat{x}_i = x_i + r_3 * ((1/Ne) * \sum_{i_1 \in H} x_{i_1})$;
The new position of particle i: $\widehat{v}_i = v_i + r_4 * ((1/Ne) * \sum_{i_1 \in H} v_{i_1})$;
if Particle($\widehat{i}$).Fitness<Particle(i).Fitness
Particle(i).Fitness=Particle($\widehat{i}$).Fitness;
Particle(i).Position=Particle($\widehat{i}$).Position;
Particle(i).Velocity=Particle($\widehat{i}$).Velocity;
if Particle($\widehat{i}$).Fitness<Best Particle.Fitness
Best Particle.Position=Particle($\widehat{i}$).Best.Position;
Best Particle.Fitness=Particle($\widehat{i}$).Best.Fitness;
End if
End if
End for
$Ne = Ne_{\min} + \mathrm{INT}\left(t * \left(\frac{(Ne_{\max} - Ne_{\min})}{t_{\max}}\right)\right)$;
End for
Output: Best Particle.Fitness

Algorithm 3: A Chaotic Starling PSO Algorithm.

randomly between x_j^l and x_j^u, where x_j^l and x_j^u are the minimum and maximum values of dimension j of the search space and also are the limits of the positions of the particles. Therefore, $x_{ij} \in [x_j^l, x_j^u]$, for $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, D$. The values of x_j^l and x_j^u are determined as (14):

$$x_j^l = \min\left(X_{ij} \mid i = 1, 2, \ldots, N\right),$$
$$x_j^u = \max\left(X_{ij} \mid i = 1, 2, \ldots, N\right), \quad \text{for } j = 1, 2, \ldots, d, \tag{14}$$

where X_{ij} is dimension j of data point X_i. In particular, the dimension D of a particle is $\mathrm{D} = K * d$.

4.3. Boundary Handling. If a particle flies outside or across the boundary of the search space, this particle no longer represents a solution. The position and velocity of such a particle are adjusted. If the position of a particle in dimension j is below the lower (above the upper) limit of that dimension, it is set to the lower (upper) limit of that dimension. Accordingly, the velocity of this particle along dimension j is reversed. These adjustments can limit the scope of the positions and change the velocities, as well as changing the directions of the flying paths, of the particles. The position and velocity of such a particle are adjusted as follows:

$$\widehat{x}_{ij} = x_j^l, \quad \text{if } x_{ij} < x_j^l,$$
$$\widehat{x}_{ij} = x_j^u, \quad \text{if } x_{ij} > x_j^u, \tag{15}$$
$$\widehat{v}_{ij} = -v_{ij},$$

where $\widehat{x}_{ij}$ is the new value of dimension j of the position and $\widehat{v}_{ij}$ is the new value of dimension j of the velocity of particle i after the adjustments.

4.4. The Pseudocode of the Proposed CSPSO. The pseudocode of the proposed CSPSO is presented in Algorithm 3.

Table 1: Benchmark Functions.

Benchmark Functions	Function Expression	Domain	$X_{\min}$	$F_{\min}$
Sphere	$f_{\min} = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$\{0\}^D$	0
Rastrigin	$f_{\min} = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	$\{0\}^D$	0
Griewank	$f_{\min} = \sum_{i=1}^{D} \left(\frac{x_i^2}{4000}\right) - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	$\{0\}^D$	0
Rosenbrock	$f_{\min} = \sum_{i=1}^{D} \left[\left(x_{i+1}^2 - x_i^2\right)^2 + \left(1 - x_i\right)^2\right]$	$[-2.048, 2.048]^D$	$\{1\}^D$	0

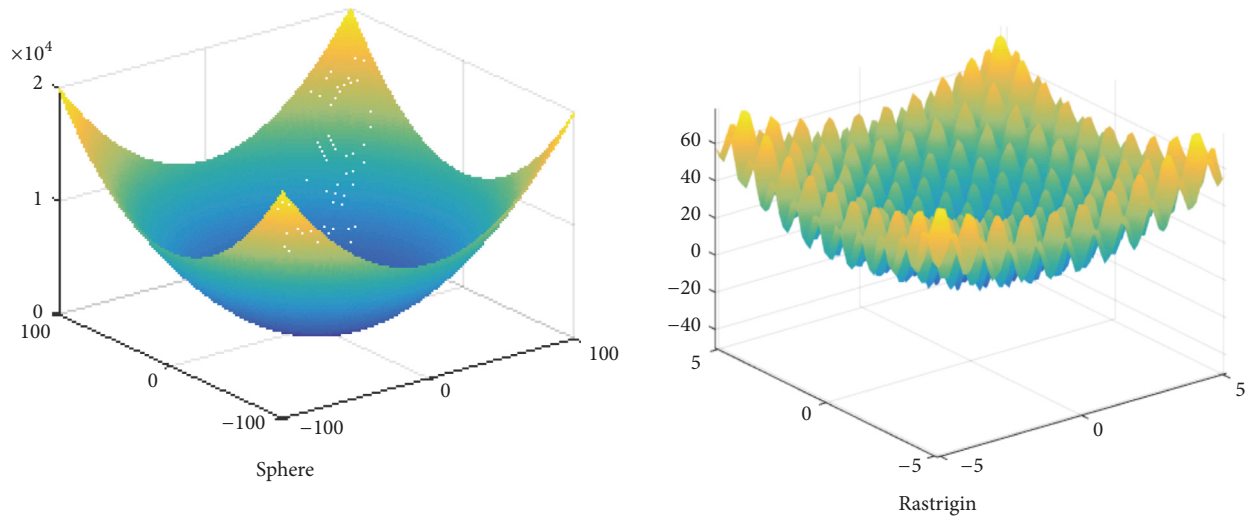


Figure 3: The Sphere and Rastrigin Function for $D = 2$.

5. Simulation Experiments

Three experiments are conducted to evaluate the performance of CSPSO. The first experiment validates the optimal parameter values in CSPSO using the Sphere and Rastrigin [33] benchmark functions. The second experiment validates the performance of CSPSO using four classical numerical benchmark functions [33]. These functions, all to be minimized, and their domains are presented in Table 1. The third experiment checks the effectiveness of CSPSO in data clustering using some datasets from the Artificial Datasets [34] and the UCI Machine Learning Repository [35]. CSPSO is implemented in MATLAB and all the experiments are conducted on a LENOVO desktop computer with an Intel 3.20 GHz i5-4460/P4 processor and 8GB of RAM in a Windows 8 environment.

5.1. Experiment on Parameter Settings. As described above, the values of the parameters have important influences on the performance of CSPSO. This section focuses on checking the influences of the three critical parameters in CSPSO, the inertia weight w, the size of the neighborhood Ne, and the number of selected particles S, so as to find appropriate values for them. The Sphere and Rastrigin functions are used to study the influences of different values of the parameters in CSPSO. The shape and ranges of the Sphere and Rastrigin functions for $D = 2$ are depicted in Figure 3. The dimension of benchmark functions is $D = 10$ in the experiment. CSPSO ran 30 times for each of the two benchmark functions.

For the validity of the experiments and fair comparisons, parameters not to be tested in the experiments are kept at the same values. These parameters are the population size $M = 50$, the max number of iterations $t_{\max} = 100$, and the cognitive coefficients $c_1 = c_2 = 2$.

The influences of the inertia weight w is tested first. Its value is adjusted according to (8). The values of $w_{\min}$ and $w_{\max}$ directly determine the decline speed of w and consequently affect the convergence speed of CSPSO. Their values are varied in the experiments. When the value of w varies, the minimum and maximum sizes of the neighborhood are kept at $Ne_{\min} = 2$ and $Ne_{\max} = 5$, and the number of selected particles is kept at $S = 10$. The best and mean values of the Sphere and Rastrigin functions are reported in Table 2.

The differences in these results are obvious. The optimal values of the Sphere and Rastrigin functions were obtained at different values of w. However, the mean values are different for the Rastrigin function because it is multimodal with many local optima. Generally, it is difficult to find global optimal solutions for multimodal functions with traditional optimization algorithms. Based on these results, the optimal

Table 2: Results when varying the inertia weight w.

Inertia weight ($w_{\min} - w_{\max}$)	Sphere	Rastrigin	Mean Time taken by Sphere	Mean Time taken by Rastrigin
0.2–0.6	0(0.0022)	0(0.9684)	3.3773	3.1321
0.2–0.9	0(0)	0(1.6269)	3.1235	3.1358
0.4–0.9	0(0)	0(1.7282)	3.1258	**3.1309**
0.4–1.2	**0(0)**	**0(0.7567)**	**3.1229**	3.1359

Table 3: Results when varying the size of the neighbourhood Ne.

Neighbor size ($Ne_{\min} - Ne_{\max}$)	Sphere	Rastrigin	Mean Time taken by Sphere	Mean Time taken by Rastrigin
2-5	0(0.1382)	0(1.0071)	3.1503	3.1420
2-7	**0(0)**	**0(0.7359)**	**3.1389**	**3.1416**
2-9	0(0)	0(1.3716)	3.1403	3.1446

Table 4: Results when varying the number of selected particles S.

S_e, S_f	S	Sphere	Rastrigin	Mean Time taken by Sphere	Mean Time taken by Rastrigin
5	**10**	**0(0)**	**0(0.6586)**	**3.1808**	**3.1966**
7	14	0(0.7567)	0(1.5407)	4.2845	4.2645
9	18	0(2.5609)	0(2.4015)	5.3850	5.3840

values of the lower and upper limits of the inertia weight are set to $w_{\min} = 0.4$ and $w_{\max} = 1.2$ where the minimum mean values are achieved.

The influences of the size of the neighborhood Ne is examined next. In CSPSO, the size of the neighborhood Ne increases gradually to enhance the local search ability. Large values will lead to slow convergence and small values will restrain the local search ability. An appropriate value balances exploration and exploitation. The lower and upper limits on the size of the neighborhood are set to different values and the best and mean values of the Sphere and Rastrigin functions are reported in Table 3.

Table 3 shows that the best results are obtained when the size of the neighborhood is between the lower and upper limits $Ne_{\min} = 2$ and $Ne_{\max} = 7$. The influences of the number of selected particles S are then tested. It has influences on both convergence and search ability. Different values for S_e and S_f are used and the results are reported in Table 4.

The results in Table 4 show that the best number of selected particles is 10 when the mean values of the test functions and computation time are both low. It is easy to see that more particles do not always lead to better results. Because of randomness built into the algorithm, sometimes most of the particles may just gather around local optima.

Based on these results, the lower and upper limits on the inertia weight $w_{\min}$ and $w_{\max}$ are set to 0.4 and 1.2, respectively, the lower and upper limits on the size of the neighborhood $Ne_{\min}$ and $Ne_{\max}$ are set to 2 and 7, respectively, and the number of selected particles S is set to 10. These parameter values in CSPSO are kept the same and are used in the following experiments.

5.2. Benchmark Functions. In this section, the 4 classical numerical benchmark functions presented in Table 1 are used to validate the effectiveness of CSPSO. The dimension of the benchmark functions, which determines the dimension of the search space, is set to $D = 10, 20$, and 30. The performance of CSPSO is compared with those of PSO, fitness-Euclidean distance ratio particle swarm optimization (FER-PSO) [23], and starling particle swarm optimization (SPSO) [25]. PSO is the basic swarm intelligent algorithm that was developed by Kennedy and Eberhart [26]. FER-PSO utilizes the individual best of a neighbor selected based on the Euclidean distance and fitness function value to replace the individual best of its own. The SPSO uses the collective responses of the particles to adjust the positions and velocities of the particles when the algorithm stagnates.

The values of the adjustable parameters in these competitive algorithms are the best ones reported in the respective references as listed in Table 5. These competitive algorithms were also run for 50 independent times each so as to get some meaningful results. The best fitness function values for these functions obtained by these algorithms are reported in Table 6.

Figure 4 shows the convergence of some functions of the three algorithms. Compared with PSO and FER-PSO in Figures 4(b) and 4(c), the curves of the function values obtained by CSPSO decline slowly at the beginning of the evolutionary process because of the time taken by local search. Hence, it has a slow convergence speed. Relying on DDT and chaotic mapping, CSPSO can find lower fitness values than other algorithms after about half of evolutionary process rather than falling into local optima. The local search strategy in the neighborhood helps in finding better solutions in the nearby neighborhood of the current position of a particle. In particular, the curve of the Sphere function obtained by CSPSO disappeared after the best value 0 has been found because this value cannot be marked on the chart. As shown in Figure 4(d) for the Rosenbrock function, a multimodal function, and traditional algorithms like PSO,

Table 5: Parameter settings in the experiment.

Parameters	PSO	ABC [22]	FER-PSO [23]	GABC [24]	SPSO [25]	CSPSO
Population (M)	50	50	50	50	50	50
t_{max}	200	200	200	200	200	200
c_1, c_2	2,2	–	2,2	–	2,2	2, 2
r_1, r_2	(0, 1)	(−1, 1)	(0.1, 0.4)	(−1, 1)	(0, 1)	(0, 1)
(w_{min}, w_{max})	1	–	1	1	(0.2, 0.4)	(0.4. 1.2)
Neighbors (Ne)	–	–	7	–	7	2–7
Selected (S)	–	–	–	–	–	10
Sub-population	–	–	–	–	14	–
Stagnant limit	–	–	–	–	2	–
Trial limit	–	10	–	10	–	–

Table 6: Best fitness function values obtained by different algorithms for the test functions.

Function	D	PSO	FER-PSO	SPSO	CSPSO
Sphere	10	0	0	**0**	**0**
	20	0.5285	0.1613	8.2232e-12	**0**
	30	2.4235	0.8891	5.1645e-12	**0**
Rastrigin	10	0.0727	0.03647	**0**	**0**
	20	0.3595	0.1895	**0**	**0**
	30	1.0374	0.4847	**0**	**0**
Griewank	10	0.5965	0.1135	**0**	**0**
	20	1.4575	1.0994	**0**	**0**
	30	3.2947	1.6874	**0**	**0**
Rosenbrock	10	6.3220	6.8162	5.2486	**0.7037**
	20	19.8643	19.2219	18.7899	**9.8431**
	30	38.1130	33.7894	28.7863	**18.6928**

FER-PSO, and SPSO are easily trapped into local optima, but CSPSO can find the smallest function values among those found by these algorithms. These results show that CSPSO has strong global search ability through the use of DDT and chaotic mapping. It can quickly escape local optima.

As the results of the 4 classic functions show in Table 6, CSPSO and SPSO perform better than PSO and FER-PSO on the Rastrigin and Griewank functions, and CSPSO performs better than the other three algorithms on the Sphere and Rosenbrock functions. In general, the results show that CSPSO has more powerful global search ability than traditional algorithms like PSO, FER-PSO, and SPSO both in single modal and multimodal functions. Furthermore, as indicated by the results in Table 6, the better performance of CSPSO than others provides evidence that the improvement in CSPSO is effective in enhancing the performance of PSO and CSPSO can find the global optimum more often than other algorithms in the experiments.

5.3. Clustering Problems. In this section, experiments on clustering problems with different datasets are performed and results of different algorithms, including CSPSO, are reported and discussed. Eight benchmark datasets are used. They are Data_3_2, Data_5_2, Data_10_2, and Data_4_3 from the work reported in [34] and Iris, Glass, Contraceptive Method Choice (CMC), and Wine from the UCI Machine Learning Repository [35]. More details about these datasets are given in Table 7. Data_5_2 and Data_4_3 are also plotted in Figure 5.

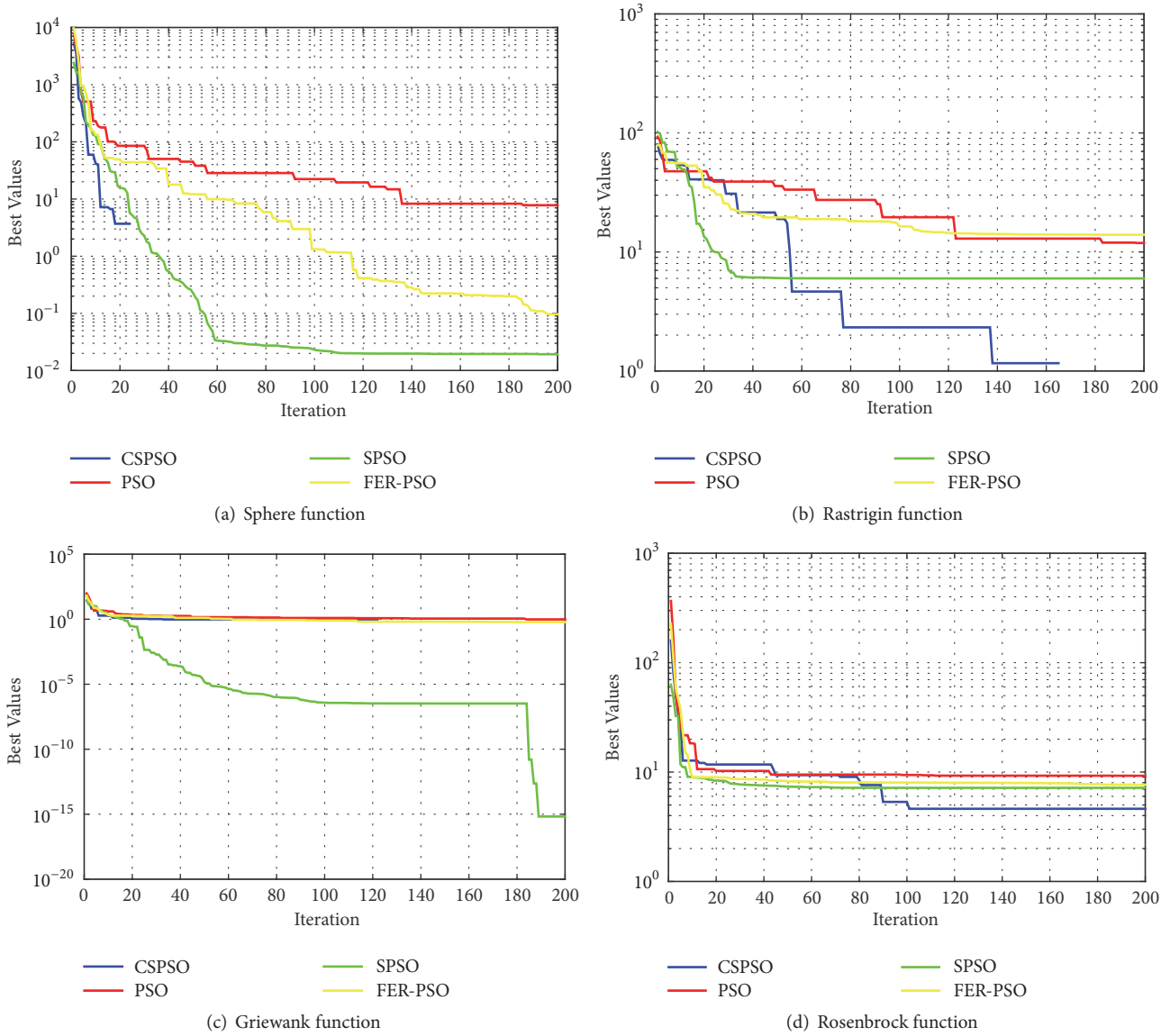
The optimized values of the parameters in the other algorithms shown in Table 5 are adopted in the experiments. In order to perform fair comparisons, all the algorithms run on all the clustering datasets for 50 times to eliminate the effects random factors. As simple statistics, best values (Best), worst values (Worst), mean values (Mean), and standard deviations (S.D.) are used as the evaluation criteria to measure the effectiveness of these clustering algorithms. The environment of experiment is the same for all clustering algorithms.

Figure 6 shows the convergence of these algorithms on four datasets for typical runs of the algorithms. The fitness functions of CSPSO decline slowly at the beginning of the evolutionary process and continue to decline, rather than dropping into local optimum, at the middle of the evolutionary process, possibly because DDT gives the particles high velocities. Local search based on Euclidean and fitness neighborhood structure also helps the particles find better positions. This phenomenon is more evident on high-dimensional datasets such as the Glass dataset.

To be more clear, Figure 7 gives more details about the convergence of these algorithms near the end of the search process on three out of the four same datasets.

Table 7: Description of the datasets used in the experiments.

Datasets	Clusters (K)	Features (d)	Total instances (N)	Instances in each clusters
Data_3_2	3	2	76	(25,25,26)
Data_5_2	5	2	250	(50,50,50,50,50)
Data_10_2	10	2	500	(50,50,50,50,50,50,50,50,50,50)
Data_4_3	4	3	400	(100,100,100,100)
Iris	3	4	150	(50,50,50)
Glass	6	9	214	(70,17,76,13,9,29)
CMC	3	9	1473	(629,334,510)
Wine	3	13	178	(59,71,48)



(a) Sphere function

(b) Rastrigin function

(c) Griewank function

(d) Rosenbrock function

Figure 4: Convergence of the algorithms on some functions ($D = 10$).

CSPSO has the best performance among all these clustering algorithms. ABC is the first converging to a local optimal point possibly because of the influence of randomly selected neighbors. GABC has a poor performance for large data volumes, such as the Data_5_2 and Glass datasets, possibly due to the local search behavior of onlooker bees. FER-PSO has apparently better convergence performance than other algorithms possibly due to the strategy that only one of the neighbors is selected to guide the search of a particle, but it is easily trapped in local optimal points in the later generations

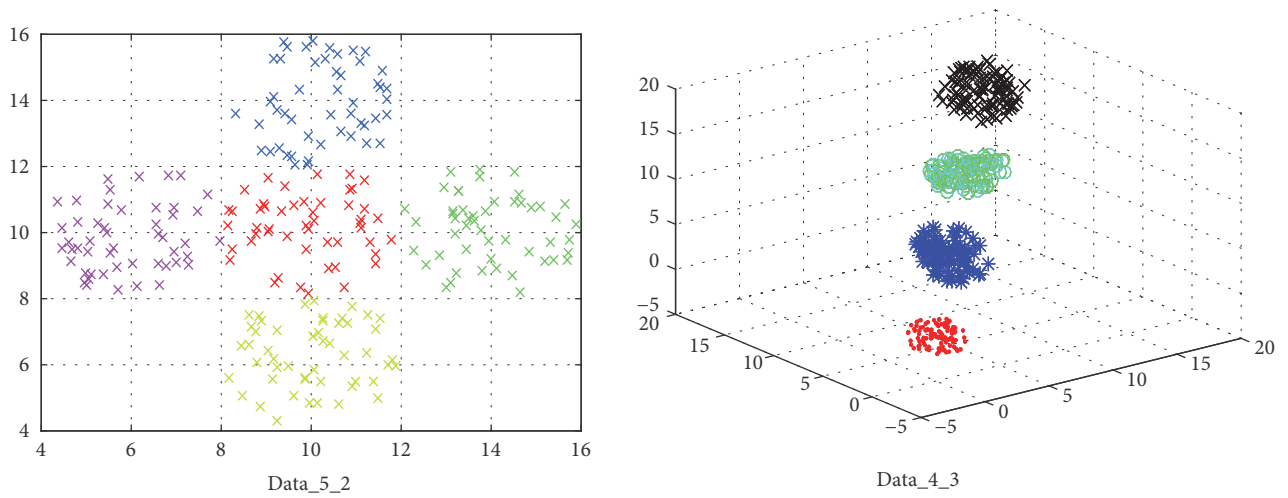


Figure 5: Distributions of data points in some datasets.

of the search process. SPSO shows a better performance in searching for a global optimum, but its local search strategy is not working on high-dimensional datasets. The above results and discussions reveal the different performances among PSO, ABC, FER-PSO, GABC, SPSO, and CSPSO and verify the good performance of CSPSO as a clustering algorithm.

It can be seen from Table 8 that CSPSO has a better performance on these classical clustering problems. For the Data_3_2, Data_5_2, and Data_10_2 datasets, CSPSO obtained lower mean fitness values than others, but other algorithms perform better than CSPSO on the best fitness values. This is possibly due to the randomness of the swarm intelligence algorithms. For the Data_4_3 datasets, because of their large numbers of data points, the ability of local search in CSPSO is affected by the selected neighbors. ABC has better stability on this dataset, but has a poor ability in searching for a global optimum than the others due to the random strategy. CSPSO outperforms the other algorithms and obtains lower mean fitness values than others on the Iris, Glass, CMC, and Wine datasets. Therefore, it has better performance on high-dimensional data. This proposed algorithm with embedded DDT and chaotic mapping has better global search ability and is more stable than traditional PSO through the use of a local search method based on starling birds. Furthermore, as the sizes of the datasets increase in both the dimension and number of data points, CSPSO has better performance than PSO, ABC, GABC, FER-PSO, and SPSO. It is more capable of dealing with big data in the current information age.

6. Conclusions

Because the performance of standard PSO depends on its parameter values, parameter adjustment is one of the most useful ways for PSO to obtain good exploration and exploitation abilities. CSPSO proposed in this study enhances the overall convergence of the searching process. Nonlinear adjustment of the inertia weight and the chaotic search method based on logistic mapping enhance the global search performance of the traditional PSO and help particles escape from local optima. DDT added to velocity update increases the velocities of the particles in later iterations and also helps particles escape from local optima. The local search strategy based on behavior of starling birds utilizes the information of the nearest neighbors in the neighborhood and determines a new collective position and a new collective velocity. The two particle selection methods, Euclidean distance and fitness function value, maintain population diversity of the particles. These improvements help particles avoid stagnation and find better solutions in the search space.

The results of simulation experiments in both benchmark function optimization and classical clustering problems show the effectiveness of CSPSO. Compared with the traditional PSO and other evolutionary algorithms, CSPSO has better convergence properties and stronger global search ability. However, as for other swarm intelligence algorithms, CSPSO may be trapped into, and may stagnate around, local optimal points and, therefore, may be unstable when applied to problems with multiple local optima. In future works, CSPSO will be applied to other combinatorial optimization problems, such as scheduling problems. Other metaheuristic methods and optimization techniques may also be used to improve the performance of PSO.

Data Availability

All the datasets used in the simulation experiments came from the Artificial Datasets, available at http://www.isical.ac.in/~sanghami/data.html (accessed June 2017), and from the UCI Machine Learning Repository, available at http://archive.ics.uci.edu/ml/datasets.html (accessed June 2017).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

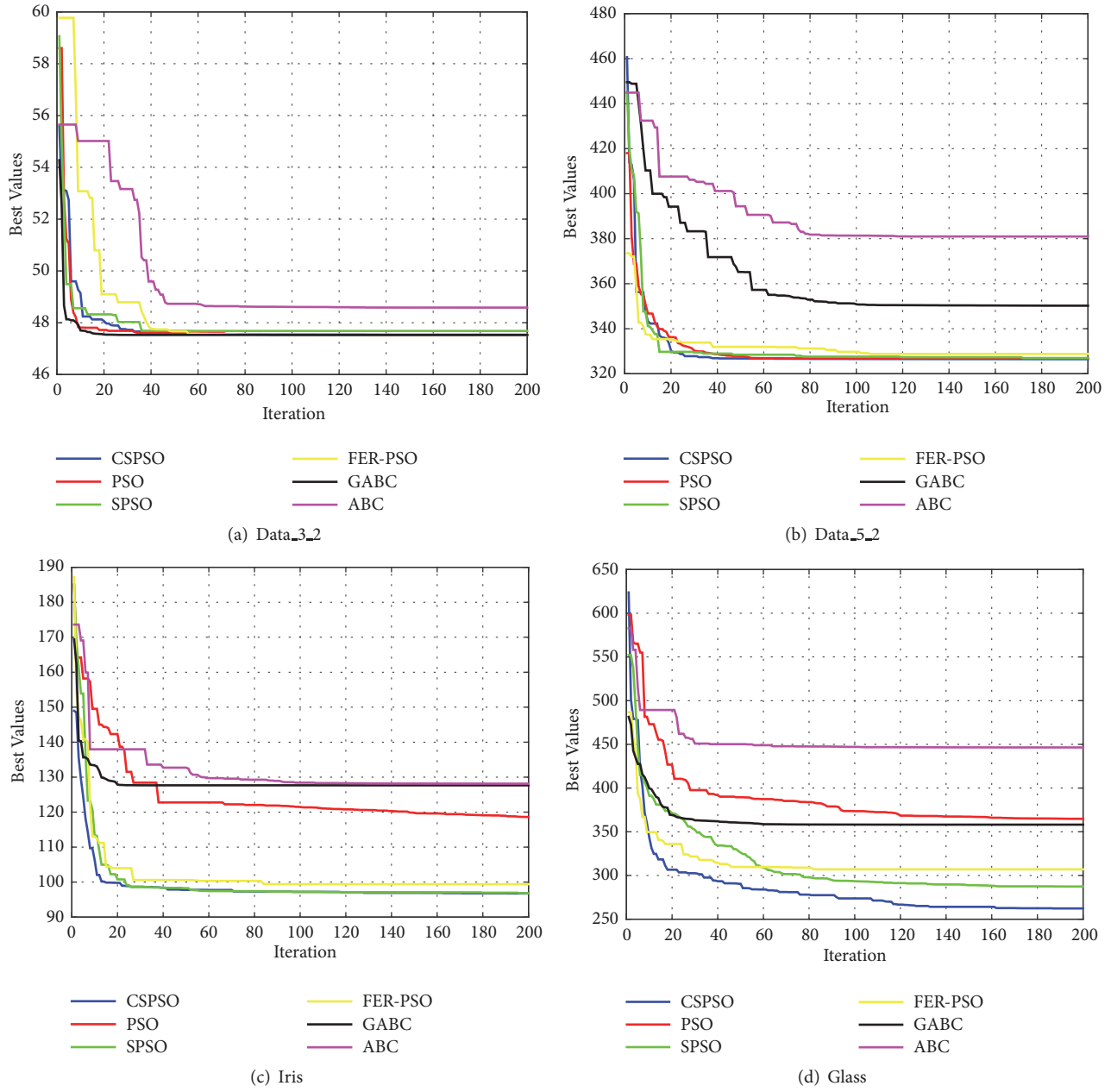


Figure 6: Convergence of the algorithms on some datasets.

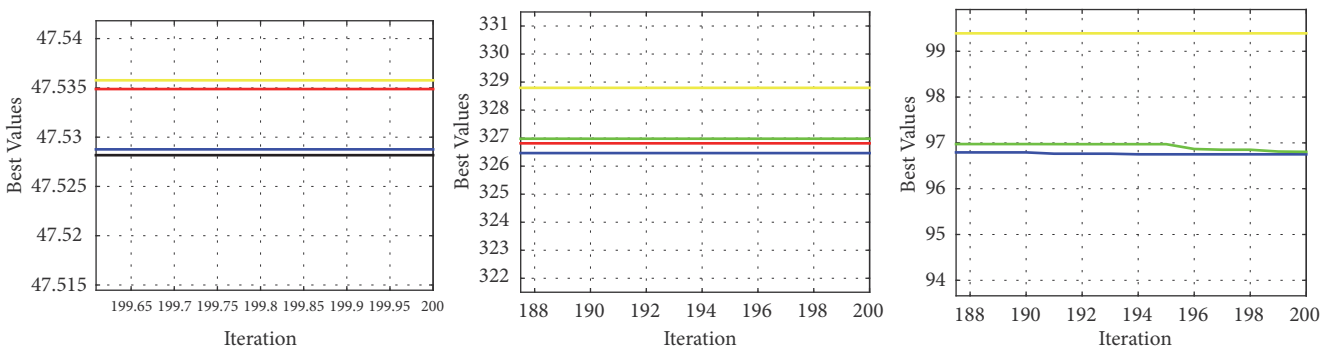


Figure 7: Convergence of the algorithms on the Data_3_2, Data_5_2, and Iris datasets.

Table 8: Comparison of performance of CSPSO with other clustering algorithms.

Datasets	Parameters	Algorithms					
		PSO	ABC	FER−PSO	GABC	SPSO	CSPSO
Data_3_2	Best	47.5287	47.7630	47.5290	**47.5282**	47.5733	**47.5282**
	Worst	59.1935	60.1039	49.7856	59.1895	53.1721	**47.5382**
	Mean	48.4644	50.9702	47.8771	48.2279	47.8934	**47.5300**
	S.D.	3.1956	3.1345	0.4967	2.7975	0.8551	**0.0022**
Data_5_2	Best	**326.4386**	337.5538	326.9987	329.7155	326.4666	326.4432
	Worst	392.2899	411.0786	332.6651	364.8400	327.0877	**326.9676**
	Mean	329.6249	373.6977	328.7897	343.5928	326.7493	**326.5300**
	S.D.	12.9978	20.1046	1.1594	9.6214	0.1536	**0.0942**
Data_10_2	Best	1.0793e+03	916.4320	875.6264	1.3134e+03	**843.2274**	849.8712
	Worst	1.4459e+03	1.2718e+03	1.2068e+03	1.1657e+03	1.1827e+03	**1.0141e+03**
	Mean	1.2875e+03	1.0608e+03	1.0066e+03	63.8046	941.8921	**918.2542**
	S.D.	70.4484	76.1788	67.1964	1.3134e+03	85.5520	**50.5344**
Data_4_3	Best	751.8897	1.0937e+03	830.5456	750.4056	749.9506	**749.6738**
	Worst	1.2866e+03	1.6347e+03	1.3753e+03	1.8485e+03	1.2726e+03	**1.2725e+03**
	Mean	838.8670	1.3838e+03	1.0312e+03	1.1489e+03	793.8641	**784.5028**
	S.D.	181.8515	**122.3579**	150.0554	271.3938	142.5136	125.1508
Iris	Best	100.5216	113.1201	98.0975	96.6829	**96.6605**	**96.6605**
	Worst	130.6931	156.8960	102.9780	127.6869	98.6906	**98.0888**
	Mean	113.7655	133.1161	100.3761	105.0487	96.9465	**96.9194**
	S.D.	7.1771	11.0706	1.2211	11.2307	0.3921	**0.2931**
Glass	Best	329.1487	333.7214	278.6565	288.5152	235.3554	**229.6544**
	Worst	436.5615	503.1607	326.5351	402.5881	309.0997	**278.9187**
	Mean	376.4035	422.3809	297.3875	346.4078	280.5624	**257.0244**
	S.D.	21.5371	32.0309	11.1730	26.9515	14.5045	**10.3633**
CMC	Best	5.8449e+03	5.9895e+03	5.6153e+03	5.6704e+03	5.5615e+03	**5.5404e+03**
	Worst	6.9260e+03	7.2137e+03	5.9728e+03	6.4143e+03	5.7676e+03	**5.5861e+03**
	Mean	6.1772e+03	6.4821e+03	5.7297e+03	5.8864e+03	5.6426e+03	**5.5562e+03**
	S.D.	186.7542	243.2599	73.0180	140.8947	44.6957	**8.7806**
Wine	Best	1.6308e+04	1.6399e+04	1.6312e+04	1.6309e+04	1.6305e+04	**1.6298e+04**
	Worst	1.6798e+04	1.7317e+04	1.6376e+04	1.6482e+04	1.6356e+04	**1.6313e+04**
	Mean	1.6438e+04	1.6705e+04	1.6336e+04	1.6355e+04	1.6322e+04	**1.6304e+04**
	S.D.	95.0097	219.7057	15.9233	35.4218	12.0030	**3.3799**

Acknowledgments

This research project was partially supported by the National Natural Science Foundation of China (61472231, 61502283, 61640201), Humanities and Social Science Research Fund of the Ministry of Education of China (12YJA630152), and the Shandong Social Science Foundation of China (16BGLJ06, 11CGLJ22).

References

[1] X. Liu, H. Liu, and H. Duan, "Particle swarm optimization based on dynamic niche technology with applications to conceptual design," *Advances in Engineering Software*, vol. 38, no. 10, pp. 668–676, 2007.

[2] Z. Bao and T. Watanabe, "Mixed constrained image filter design using particle swarm optimization," *Artificial Life and Robotics*, vol. 15, no. 3, pp. 363–368, 2010.

[3] S. Sahin and M. A. Cavuslu, "FPGA Implementation of Wavelet Neural Network Training with PSO/iPSO," *Journal of Circuits, Systems and Computers*, vol. 27, no. 6, 2018.

[4] D. Wu and H. Gao, "A BP and Switching PSO Based Optimization Approach for Engine Optimization," *National Academy of Science Letters*, vol. 40, no. 1, pp. 33–37, 2017.

[5] Q. Jia and Y. Guo, "Hybridization of ABC and PSO algorithms for improved solutions of RCPSP," *Journal of the Chinese Institute of Engineers, Transactions of the Chinese Institute of Engineers,Series A/Chung-kuo Kung Ch'eng Hsuch K'an*, vol. 39, no. 6, pp. 727–734, 2016.

[6] C. Yue, B. Qu, and J. Liang, "A Multi-objective Particle Swarm Optimizer Using Ring Topology for Solving Multimodal Multi-objective Problems," *IEEE Transactions on Evolutionary Computation*, 2017.

[7] X. Liu and J. Xue, "A Cluster Splitting Technique by Hopfield Networks and P Systems on Simplices," *Neural Processing Letters*, pp. 1–24, 2017.

[8] Y. Zhao, X. Liu, and W. Wang, "Spiking neural P systems with neuron division and dissolution," *PLoS ONE*, vol. 11, no. 9, Article ID e0162882, 2016.

[9] X. Liu, Y. Zhao, and M. Sun, "An Improved Apriori Algorithm Based on an Evolution-Communication Tissue-Like P System with Promoters and Inhibitors," *Discrete Dynamics in Nature and Society*, vol. 2017, 2017.

[10] N. Netjinda, T. Achalakul, and B. Sirinaovakul, "Particle Swarm Optimization inspired by starling flock behavior," *Applied Soft Computing*, vol. 35, pp. 411–422, 2015.

[11] A. El Dor, D. Lemoine, M. Clerc, P. Siarry, L. Deroussi, and M. Gourgand, "Dynamic cluster in particle swarm optimization algorithm," *Natural Computing*, vol. 14, no. 4, pp. 655–672, 2015.

[12] Y. M. B. Ali, "Unsupervised Clustering Based an Adaptive Particle Swarm Optimization Algorithm," *Neural Processing Letters*, vol. 44, no. 1, pp. 221–244, 2016.

[13] G. Armano and M. R. Farmani, "Multiobjective clustering analysis using particle swarm optimization," *Expert Systems with Applications*, vol. 55, pp. 184–193, 2016.

[14] B. Niu, Q. Duan, L. Tan, C. Liu, and P. Liang, "A Population-Based Clustering Technique Using Particle Swarm Optimization and K-Means," in *Advances in Swarm and Computational Intelligence*, vol. 9140 of *Lecture Notes in Computer Science*, pp. 145–152, Springer International Publishing, Cham, 2015.

[15] X. Zhao, W. Lin, J. Hao, X. Zuo, and J. Yuan, "Clustering and pattern search for enhancing particle swarm optimization with Euclidean spatial neighborhood search," *Neurocomputing*, vol. 171, pp. 966–981, 2016.

[16] E. Çomak, "A modified particle swarm optimization algorithm using Renyi entropy-based clustering," *Neural Computing and Applications*, vol. 27, no. 5, pp. 1381–1390, 2016.

[17] K. K. Bharti and P. K. Singh, "Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering," *Applied Soft Computing*, vol. 43, pp. 20–34, 2016.

[18] W. Song, W. Ma, and Y. Qiao, "Particle swarm optimization algorithm with environmental factors for clustering analysis," *Soft Computing*, vol. 21, no. 2, pp. 283–293, 2017.

[19] R. Liu, J. Li, J. Fan, C. Mu, and L. Jiao, "A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization," *European Journal of Operational Research*, vol. 261, no. 3, pp. 1028–1051, 2017.

[20] X. Chu, B. Niu, J. J. Liang, and Q. Lu, "An orthogonal-design hybrid particle swarm optimiser with application to capacitated facility location problem," *International Journal of Bio-Inspired Computation*, vol. 8, no. 5, pp. 268–285, 2016.

[21] D. Wang, D. Tan, and L. Liu, "Particle Swarm Optimization: An overview," *Soft Computing*, vol. 22, no. 2, p. 22, 2018.

[22] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report TR0, 2005.

[23] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio," in *Proceedings of the the 9th annual conference*, pp. 78–85, London, England, July 2007.

[24] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[25] A. Laudani, F. R. Fulginei, G. M. Lozito, and A. Salvini, "Swarm/flock optimization algorithms as continuous dynamic systems," *Applied Mathematics and Computation*, vol. 243, pp. 670–683, 2014.

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

[27] K. K. Bharti and P. K. Singh, "Chaotic gradient artificial bee colony for text clustering," *Soft Computing*, vol. 20, no. 3, pp. 1113–1126, 2016.

[28] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[29] F. S. Hover and M. S. Triantafyllou, "Application of polynomial chaos in stability and control," *Automatica*, vol. 42, no. 5, pp. 789–795, 2006.

[30] X. Liu, L. Xiang, and X. Wang, "Spatial cluster analysis by the adleman-lipton DNA computing model and flexible grids," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 894207, 32 pages, 2012.

[31] H. Peng, X. Luo, Z. Gao, J. Wang, and Z. Pei, "A Novel Clustering Algorithm Inspired by Membrane Computing," *The Scientific World Journal*, vol. 3, no. 22, Article ID 929471, 8 pages, 2015.

[32] B. Ufnalski and L. M. Grzesiak, "Plug-in direct particle swarm repetitive controller with a reduced dimensionality of a fitness landscape - a multi-swarm approach," *Bulletin of the Polish Academy of Sciences—Technical Sciences*, vol. 63, no. 4, pp. 857–866, 2015.

[33] A. O. Griewank, "Generalized descent for global optimization," *Journal of Optimization Theory and Applications*, vol. 34, no. 1, pp. 11–39, 1981.

[34] "Datasets," http://www.isical.ac.in/~sanghami/data.html.

[35] H. C. Blake, *UCI Repository of Machine Learning Databases*, 1998.