Cooperative Particle Swarm Optimization Algorithm with Cloud Mutation Operator Based on Normal Cloud Model

Jiahui Luo and Ying Gao

***Abstract*—Particle Swarm Optimization (PSO) algorithm is an intelligent optimization algorithm originating in bird predation behavior, which is widely used in functional optimization, neural network training, pattern classification, system control and related field. Although the Particle Swarm Optimization algorithm's convergence speed is very fast, it is easy to cause premature convergence and poor performance in multi-peak problems. In order to solve these shortcomings, this paper proposed a Cooperative Particle Swarm Optimization algorithm based on normal Cloud Model with cloud mutation operator, and improved it in three aspects. Firstly, the normal cloud mutation operator was added in the process of particles update, which improved the search performance of Particle Swarm Optimization algorithm on multi-peak problems. Second, this paper used the Huffman tree's construction process to divide the particles into different sub-populations, and this method ensured the diversity of the population. Thirdly, the whole encoding process was saved and transmitted by using the paradigm Huffman coding, which reduced the spatial complexity of the algorithm. At the same time, the inertia weight in the algorithm was optimized by the method of adaptive inertia weight and stochastic inertia weight, which balanced the local and global search ability of the particles and ensured the synergy of the algorithm. In this paper, we used the simulation experiment method to conduct 50 independent experiments on four common functions. By comparing the mean of optimal solution, variance, average convergence time, average convergence generation and other indicators of PSO, CH-PSO, HC-PSO and CH-HC-PSO algorithms, a phenomenon was found that the cooperative Particle Swarm Optimization algorithm with cloud mutation operator based on normal Cloud Model is better than the other three algorithms in performance, which effectively solves the problems of premature convergence and multi-peak optimization performance. This algorithm was applied to solve the high-dimensional problems effectively.**

***Index Terms*—Cloud model, cloud mutation operator, paradigm Huffman coding, particle swarm optimization.**

I. Introduction

Originating in the simulation of bird predation behavior, the Particle Swarm Optimization (PSO) algorithms an intelligent optimization algorithm which was proposed by Kennedy and Eberhart in 1995 [1]. There are many advantages of this algorithm, such as fast convergence speed, few adjustment parameters and simple structure and so on. Hence, it is widely used in functional optimization, neural network training, pattern classification, system control and related field [2], [3]. There is no doubt that the Particle Swarm Optimization algorithm has the advantage of fast convergence speed. However, there are still many problems in the execution of the algorithm. For instance, since all the particles are close to the global optimal direction, the particles lose diversity in the process of evolution, which are easy to cause premature convergence [4]. Since the particles learn each other, constructing neighborhood in the entire D-dimensional space, a "dimension disaster" would be formed when the search space dimension is high.

The Cloud Model (CM) algorithm is an uncertainty conversion model between qualitative concept and quantitative representations, expressed by linguistic values and proposed by Dr. Li Deyi from the basic principles of species evolution in nature [5], [6]. These methods revealed the randomness and fuzziness between linguistic values and deterministic exact values.

At present, many scholars at home and abroad combined Cloud Model algorithms with other intelligent optimization algorithms and proposed many new algorithms. For example, the genetic algorithm and Cloud Model algorithm were combined to propose an evolutionary algorithm based on Cloud Model in [7]. And a cloud generator was used to replace the crossover and mutation operators of genetic algorithms and a cloud genetic algorithm was proposed in [8]. Whereas the Cloud Model algorithm was introduced into the evolution mechanism of Particle Swarm Optimization algorithm in [9] and [10], and these methods improved effectively the optimization ability of Particle Swarm Optimization algorithm.

Through the improvement of the above algorithm, the particles were validly prevented from falling into the local optimal solution, and the diversity of the population was enhanced, and the good result was obtained in the function optimization. However, the above research results will still cause premature convergence, and there is still room for improvement in the optimization of multi-peak problems.

Aiming at the shortcomings of the standard Particle Swarm Optimization algorithm which is easy to cause premature convergence and its poor performance in multi-peak problems, this paper proposed a cooperative Particle Swarm Optimization algorithm based on normal Cloud Model with cloud mutation operator. The algorithm absorbs the advantages of the above research results, and it made related improvements in three aspects. First, the randomness and fuzziness of the Cloud Model algorithm were used to improve the search performance of the Particle Swarm Optimization algorithm on multi-peak problems. Second, the method in [11] was improved and applied in this paper. And the Huffman tree's construction process was used to divide

Manuscript received May 13, 2019; accepted August 7, 2019. This work was supported by Science and Technology Projects of Guangdong Province, China, under Grant Nos. 2016B010127001.

J. Luo and Y. Gao are with the School of Computer Science and Education Software, Guangzhou University, 510006 China (Corresponding author: Y. Gao; e-mail: 2111706006@e.gzhu.edu.com; gaoying_gzhu@outlook.com).

the subpopulation of the Particle Swarm Optimization algorithm, the tree was divided two children of the same parent node into different subpopulations. In the middle, the diversity of the population was guaranteed. Thirdly, the paradigm Huffman coding was used to save and transmit the entire encoding processing the process of encoding with Huffman tree. Through the improvement of this algorithm, the space complexity of the algorithm and the waste of storage space were reduced by these methods. By using the Cloud Model algorithm and Huffman coding to optimize the Particle Swarm Optimization algorithm, the particles are largely prevented from falling into the local optimal solution, which achieves the goal of limiting the premature convergence of the particles.

II. Particle Swarm Optimization Algorithm

The Particle Swarm Optimization algorithm is an intelligent optimization algorithm proposed by Kennedy and Eberhart in 1995 [1]. In the Particle Swarm Optimization algorithm, the potential solution for each optimization problem is a particle in the search space. There are many parameter of each particle, including the fitness value defined by the optimized function, the velocity, direction and distance of flight.

The basic principle of the Particle Swarm Optimization algorithm generates a group of random particles during initialization, and then it generates an optimal solution by iteration. In the process of iteration, the particles update themselves by tracking individual optimality and population optimality. Finally, the optimal solutions in the optimization problems are found.

If the population size of the particles is *M* and the target search spaces' dimensional is *D*, the evolution process of each particle's learning are as follow:

$$V_{id}^{k+1} = V_{id}^{k} + c_1 r_1 (P_{id} - X_{id}^{k}) + c_2 r_2 (P_{gd} - X_{id}^{k}) \quad (1)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad (2)$$

In this formula, the parameter X_{id}^{k} represents the position of the i^{th} (i=1, 2… M) particles after the k^{th} iteration in the d^{th} search space. The parameter V_{id}^{k+1} represents the position of the i^{th} particle after the $(k+1)^{th}$ iteration in the d-dimensional search space. The parameter V_{id}^{k} represents the velocity of the i^{th} particle after the k^{th} iteration in the d-dimensional search space. The parameter V_{id}^{k+1} represents the velocity of the i^{th} particle after the $(k+1)^{th}$ iteration in the d-dimensional search space. The parameters c_1 and c_2 are learning factors, they are greater than zero, which are called acceleration coefficients. Generally, they equal 2. The parameters r_1 and r_2 are random numbers on [0, 1]. The parameter P_{id} represents the optimal position of the i^{th} particle in the d-dimensional search space. The parameter P_{gd} represents the optimal position of the group in the d-dimensional search space.

On the basis of the basic Particle Swarm Optimization algorithm, Shi *et al.* published a paper at the 1998 International Conference on Evolutionary Computation [12], which modified the Particle Swarm Optimization algorithm and introduced the inertia weight ω. It improve to formula (1), the new formula is as follows:

$$V_{id}^{k+1} = \omega V_{id}^{k} + c_1 r_1 (P_{id} - X_{id}^{k}) + c_2 r_2 (P_{gd} - X_{id}^{k}) \quad (3)$$

The formula consists of three parts. The first part is the "inertia" or "momentum" part, which reflects the habit of particle motion. The second part is the "cognitive" part, which reflects the memory of particle motion for its own experience. The third part is the "social" part, which reflects the ability of particles to share historical experience through group cooperation.

Although the Particle Swarm Optimization algorithm has a simple structure and a fast running speed, when searching in the solution space, some particles would be happen an "oscillation" phenomenon near the optimal solution, which affects the accuracy of the result. In order to improve the global search ability of the algorithm, the parameters of the Particle Swarm Optimization algorithm need to be tuned.

According to formula (2) and formula (3), the parameters affecting the Particle Swarm Optimization algorithm mainly include inertia weight ω and acceleration factors c_1 and c_2.

Inertia weight is a very important parameter in the Particle Swarm Optimization algorithm. It describes the degree of influence of historical factors on current factors in the iterative process. Larger ω is beneficial to improve the global search ability of the algorithm, whereas smaller ω can enhance the local search ability of the algorithm.

In most Particle Swarm Optimization algorithm improvements, in order to maintain the search ability and convergence of the algorithm, the common methods include linear decrease weight method, adaptive inertia weight method and stochastic inertia weight method.

A. Linear Decrease Weight Method

The first method is using linear decrease weight method to control inertia weight's parameter settings. The formula is as follows:

$$\omega = \omega_{\max} - t * \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \quad (4)$$

In formula (4), the parameter $\omega_{\max}$ is the maximum value of the inertia weight ω; the parameter $\omega_{\min}$ is the minimum value of the inertia weight ω. The parameter t is the current iteration number of the population, and the parameter $t_{\max}$ is the maximum number of iterations that the population can reach. By adjusting the inertia weight ω, the synergy of the algorithm is guaranteed.

B. Adaptive Inertia Weight Method

The second method is using adaptive inertia weight method to control inertia weight's parameter settings. The formula is as follows:

$$\omega = \begin{cases} (\omega_{\max} - \omega_{\min}) * (f - f_{\min}), f \le f_{avg} \\ \omega_{\max}, f > f_{\min} \end{cases} \quad (5)$$

In the formula (5), the parameter $\omega_{\max}$ is the maximum value of the inertia weight ω; the parameter $\omega_{\min}$ is the minimum value of the inertia weight ω. The parameter f represents the current objective function value of the particle. The parameter f_{avg} represents the average target value of all the current particles, and the parameter $f_{\min}$ represents the minimum target value of all the current particles. Since the inertia weight changes as the objective function value of the particle changes, it is called an adaptive weight. By adjusting the inertia weight ω, the global search ability and local search ability of the Particle Swarm Optimization algorithm are effectively balanced.

C. Stochastic Inertia Weight Method

The third method is using stochastic inertia weight method to control inertia weight's parameter settings. The inertia weight is set to a random number obeying the Gaussian distribution. And the inertia weight is continuously and dynamically adjusted by the influence of the random factor to eliminate the local premature convergence, the search performance of the algorithm was improved in the global. In the early stage of the algorithm search, the random inertia weight was used to ensure that ω has the opportunity to take smaller values, thus t the convergence speed of this algorithm was accelerated. In the later stage of the algorithm search, the random inertia weight was used to ensure that ω has the opportunity to take larger values, thus the convergence accuracy of this algorithm was improved. The optimization parameter is setting by using random inertia weight. The formulas are as follow:

$$\omega = \mu + \sigma * N(0,1) \tag{6}$$

$$\mu = \mu_{\min} + (\mu_{\max} - \mu_{\min}) * \xi \tag{7}$$

In the formula (6), N (0, 1) represents a random number of a standard normal distribution, and the parameter ξ represents a random number between 0 and 1. Since the selected inertia weight is random, the influence of historical speed on the current speed is also random. So the method can balance the global search ability and the local search ability, and also it can overcome the shortcomings of linear decrease inertia weight.

D. Parameter Adjustment

The above three methods are widely used in various optimization problems, but in the solution of many optimization problems, using the linear decrease weight method to reduce the inertia weight cannot be well matched with the optimization process. So we need to compare the latter two methods, and we would choice the methods to finish this experiment.

III. Cloud Model Algorithm

The Cloud Model is a mathematical model that quantitatively transforms deterministic knowledge, portraying the randomness and ambiguity of people's objects in the objective world [5], [6]. It combines qualitative analysis with quantitative analysis to provide mathematical methods for the processing and analysis of objective things. The definition of the Cloud Model is shown in Definition 1.

Definition 1 [6], [13]: U is assumed to a quantitative neighborhood expressed by exact values, and C is the corresponding qualitative concept on U. If any element x in the neighborhood U has a random number with a stable tendency, the distribution of x on the neighborhood U is called a Cloud Model (Cloud), and this metric is denoted as $C(x)$. Each x is called a cloud drop. When $C(x)$ obeys a Gaussian distribution, it is called a normal Cloud Model.

The characteristics of the Cloud Model are characterized by the mathematical expectation *Ex*, entropy *En* and super entropy *He* [13]. These three variables reflect the overall quantitative characteristics of the Cloud Model qualitative concept *C*. Mathematical expectation *Ex* is a typical sample of conceptual quantification, which is the point in the neighborhood space that it can represent the qualitative concept *C*. Entropy *En* refers to the range of cloud drop groups that it can be accepted by the qualitative concept *C* in the neighborhood space. It reflects the uncertainty of the qualitative concept and describes the fuzziness of the Cloud Model. The larger the value of entropy *En*, the greater the fuzziness and randomness. The super-entropy *He* is a measure of uncertainty, which represents the degree of condensation of cloud droplets, which is the entropy of entropy. The larger the super-entropy He, the greater the uncertainty. The algorithm for generating cloud drops is called a cloud generator. Among them, the normal cloud generator is a special cloud generator, and the algorithm is executed as shown in Algorithm 1.

Algorithm 1: Basic Normal Cloud Generator [14]:

INPUT: {*Ex*, *En*, *He*}, *n*

OUTPUT: {(x_1, μ_1), (x_2, μ_2)… (x_n, μ_n)}

for i = 1 to n

$En' = \mathrm{RAND}(En, He)$ // generate a normal random number with an expected value of En and a variance of *He*

$$x_i = \mathrm{RAND}(Ex, En')$$

$$\mu_i = e^{\frac{-(x_i - Ex)^2}{2(En')^2}}$$

drop(x_i, μ_i) / / generate the i^{th} cloud drop

IV. Cooperative Particle Swarm Optimization Algorithm

Through the optimization of the inertia weight ω, the speed of particle flight is effectively controlled, and the global and local search ability of the particle is balanced. In order to improve the convergence speed and optimization performance of the algorithm whereas ensuring particle diversity, a cooperative Particle Swarm Optimization algorithm with cloud mutation operator based on normal Cloud Model is proposed in this paper.

In this algorithm, the co-evolution of each particle is implemented based on the Huffman tree. The Huffman tree refers to the tree with the smallest weighted path. The weighted path formula is:

$$\mathrm{WPL} = \sum_{k=1}^{n} \mathrm{W}_k l_k \tag{8}$$

In this formula, the parameter l_k is the path length from the root to the kth node. The parameter W_k is the weight of the k^{th} node. The parameter n is the sum of the points, and when the *WPL* is the minimum, the tree is a Huffman tree, and the relative path of the weight is the shortest.

After the Huffman tree is successfully constructed, the Huffman tree can be encoded. Among them, the left child uses "0" to indicate encoding, whereas the right child uses "1" to indicate encoding. And the root encoding from the root node to the leaf node is called Huffman encoding.

In this paper, the construction process of Huffman tree is applied to the Particle Swarm Optimization algorithm. The individual optimal solutions of the particle are used as the weight of each node. By comparing the weights, a complete Huffman tree is constructed step by step. According to the principle of Huffman coding, Huffman coding is performed on the individual optimal solutions of each particle.

Since the optimal values of the two children of the same parent node in the Huffman tree are the closest, in order to avoid the algorithm falling into local optimum and leading to premature convergence, these nodes were divided the left and right child nodes of the same parent node into two sub-populations in this paper. In this way, the diversity of the population is guaranteed and the performance of the algorithm is improved. Since the Huffman tree is a binary tree, the population can be divided into a left subgroup and a right subgroup when the population was divided. In the initial stage, we are using the construction process of the Huffman tree to construct the population, it is not necessary to define the number of populations, and the particles is divided by the left subgroup and the right subgroup of the binary tree. This method balances the global and local search capabilities of the entire population. In the search process, the left subgroup and the right subgroup are searched independently, and the local optimal solution and the global optimal solution of the particle are searched in the left subgroup and the right subgroup respectively. Finally, the evolutionary information is shared by the information version to achieve the purpose of co-evolution.

However, since the Huffman tree needs to know the structure of the Huffman tree during the encoding process. And the encoder saves and transmits the Huffman tree for the decoder. This process results in waste of storage space and increased overhead. Therefore, the Huffman trees need to be improved to reduce unnecessary waste. The paradigm Huffman coding was originally proposed by Schwartz [1964], which is a subset of Huffman coding [15]. The structure of the Huffman coding tree can be reconstructed with very little data using some mandatory rules. Paradigm Huffman coding requires that code word of the same length be binary representations of consecutive integers, whereas the first code with the smallest length of the agreed code word starts at 0. It is calculated as follows:

$$f(i) = 2(f(i-1)+1) \tag{9}$$

It can be seen from the above formula that the decoder can recover the structure of the entire Huffman coding tree according to the length of each code word, without saving and transmitting the entire encoding process, so the waste of storage space can be reduced by this method.

V. Cooperative Particle Swarm Optimization Algorithm with Cloud Mutation Operator Based on Normal Cloud Model

This paper proposed a cooperative Particle Swarm Optimization algorithm based on normal Cloud Model with cloud mutation operator. It combines the advantages of Cloud Model and cooperative Particle Swarm Optimization algorithm, which not only it improves the fuzziness and randomness of the algorithm, but also it ensures the diversity of the population and increases convergence speed.

When the cooperative Particle Swarm Optimization algorithm with Cloud Mutation Operator based on normal Cloud Model is used to find the extreme value, it is necessary to set the metric to determine whether the particle falls into the local optimal solution. The method of judgment is shown in Definition 2.

Definition 2: Given the thresholds *N* and *K* in advance, when the global extreme value has not evolved for *N* consecutive generations or the evolutionary path amplitude is less than *K*, the particles can be considered to be locally optimal, and all particles are passed through the normal cloud according to the global extreme value. Finally, the generator performs the mutation operation.

When searching for the optimal solution, if the current solution has a large fitness, the current value is very close to the optimal solution. In this case, it should be searched in a small range to improve the convergence speed of the algorithm. Instead, you need to expand your search to prevent premature convergence. According to this basic principle, the CH-PSO algorithm is improved and the CH-HC-PSO algorithm is designed. The implementation steps of the algorithm are as follows:

Step 1: Setting the parameter values such as the number of particle groups, the number of dimensions, and the maximum number of iterations;

Step 2: Setting the parameter values of the initialization inertia weight and the acceleration factor;

Step 3: Randomly initializing the position and velocity of the particles in the population;

Step 4: Calculating the fitness value of the particle under the constraint condition;

Step 5: Calculating the individual extreme value Pi_{best} and the global extreme value Pg_{best};

Step 6: Constructing a Huffman tree according to the individual extreme value Pi_{best}, and generating a paradigm Huffman coding of the Huffman tree;

Step 7: The population is divided into two sub-populations, with the last bit of the Huffman coding as the division criterion. The left child uses "0" to indicate encoding, whereas the right child uses "1" to indicate encoding.

Step 8: The two sub-populations calculate the fitness value of the particle separately, and obtaining the individual optimal solution and the global optimal solution. The individual extreme value of the left sub-population is recorded as Pli_{best}, and the global extreme value of the left sub-population is recorded as Plg_{best}. The individual extreme value of the right sub-population is recorded as Pri_{best}, and the global extreme value of the right subpopulation is recorded as Prg_{best}.

Step 9: Comparing the global extreme value Plg_{best} of the left subpopulation, the global extreme value Prg_{best} of the

right subpopulation, and the global extreme value Pg_{best} of the population, and the results are shared into the information board;

Step 10: Taking the maximum value to update the global optimal solution Pg_{best} of the population;

Step 11: determines whether the variation threshold N is reached. If it is reached, each particle is mutated according to definition 2. Otherwise, the inertia weight is adjusted by using formula (5) or formula (6);

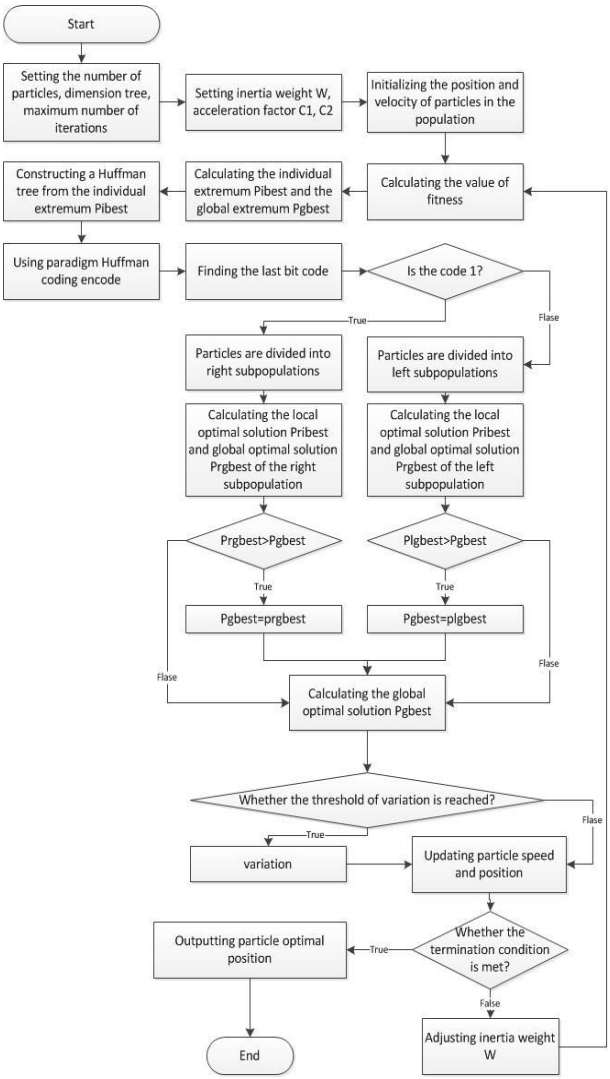


Fig. 1. Algorithm running process.

Step 12: The velocity and the position of the particle are updated by using formula (1) and formula (2);

Step 13: Determine whether the maximum number of iterations has been reached, if the condition is met, step 14 is performed, and if the condition is not met, step 4 is returned;

Step 14: The optimization operation is terminated and the particle optimal position is outputted.

The above algorithm combines the cloud model and the particle swarm optimization algorithm to effectively prevent premature convergence.

The algorithm flow diagram is shown in Fig. 1.

VI. Simulation and Results Analysis

This experiment used four fitness functions to test. The expressions of the fitness functions are shown in Table I. The experimental dimension is 10 and the number of iterations is 2000. In these experiments, the fitness ranges are [-20, 20].

In order to optimize the inertia weight of the algorithm, we conducted 50 experiments and compared the above four methods of inertia weight processing. The comparison results are shown in Table II and Table III.

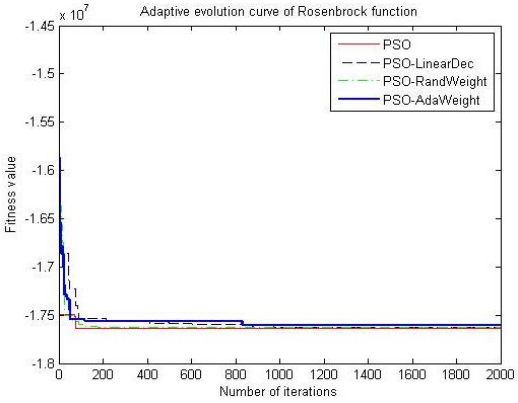


Fig. 2. Adaptive evolution curve of Rosenbrock function.

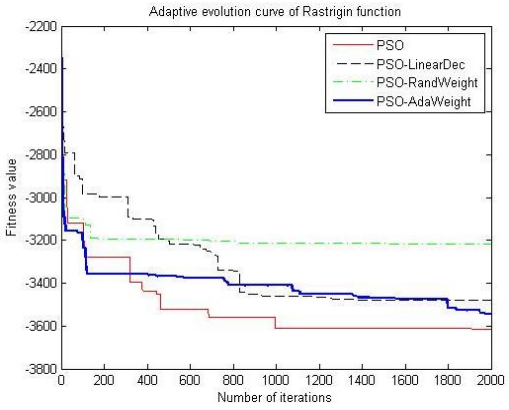


Fig. 3. Adaptive evolution curve of Rastrigin function.

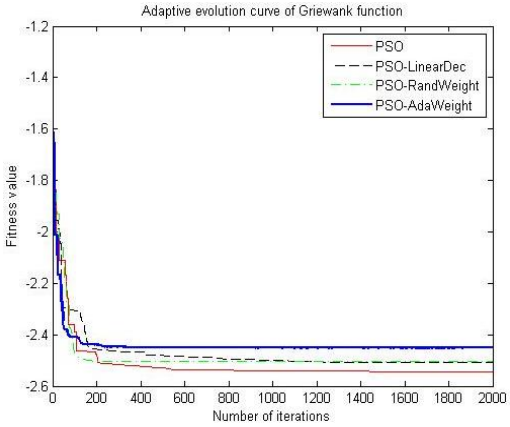


Fig. 4. Adaptive evolution curve of Griewank function.

From the analysis of Table II and Table III, the adaptive inertia weight method runs more time, and its spatial complexity is larger than other methods. However, from the mean of the optimal solution of the four fitness functions, the adaptive inertia weight method can get the most excellent solution, so we choices the adaptive inertia weight method for experiments. A comparison of the four methods of inertia weight adjustment can be seen from Fig. 2 to Fig. 5.

In order to prove the effectiveness of the algorithm, we compared four algorithms to test their performance. The algorithms included the basic Particle Swarm Optimization

algorithm (BPSO), the Cloud Hyper mutation Particle Swarm Optimization algorithm (CH-PSO), the Huffman-based Cooperative Particle Swarm Optimization algorithm (HC-PSO), Cooperative Particle Swarm Optimization algorithm with Cloud Mutation Operator Based on Normal Cloud Model (CH-HC-PSO). The specific results of the comparative experiment are shown in Table IV and Table V.

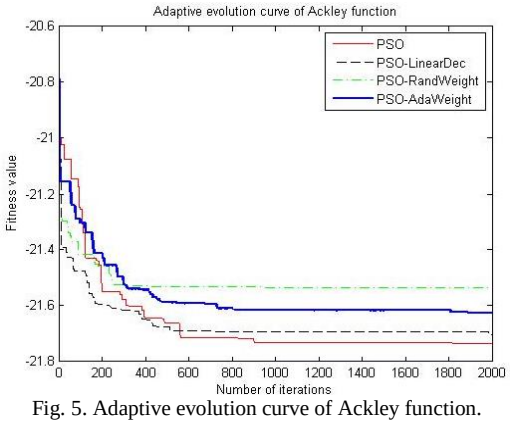


Fig. 5. Adaptive evolution curve of Ackley function.

TABLE I: FITNESS FUNCTION LIST

Function	Expression	Range
Rosen-brock	$f_1(x) = \sum_{i=1}^{n} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	$X \in$ [-2. 048,2. 048]
Rastrigin	$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10(\cos 2\pi x_i) + 10)$	$X \in$ [-5. 12,5. 12]
Griewank	$f_3(x) = \frac{1}{4000} * \sum_{i=1}^{n} x_i^2 - \prod_{i}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$X \in$[-600,600]
Ackley	$f_4(x) = -20 * e^{-0.2 * \sqrt{\frac{1}{n} * x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)} + 20 + e$	$X \in$[-32,32]

TABLE II: COMPARISON OF FOUR INERTIA WEIGHT PROCESSING METHODS (A)

Function	Method	Worst	Best	Mean
Rosenbrock	Common	-1. 7613e+07	-1. 7633e+07	-1. 7624e+07
	LinearDec	-1. 7603e+07	-1. 7634e+07	-1. 7618e+07
	RandWeight	-1. 7617e+07	-1. 7630e+07	-1. 7623e+07
	AdaWeight	-1. 7623e+07	-1. 7633e+07	-1. 7629e+07
Rastrigin	Common	-3. 5335e+03	-3. 8098e+03	-3. 6871e+03
	LinearDec	-3. 3329e+03	-3. 6315e+03	-3. 4593e+03
	RandWeight	-3. 4798e+03	-3. 8615e+03	-3. 6268e+03
	AdaWeight	-3. 4790e+03	-3. 7647e+03	-3. 6594e+03
Griewank	Common	-2. 4060	-2. 5049	-2. 4678
	LinearDec	-2. 2832	-2. 3687	-2. 3295
	RandWeight	-2. 3436	-2. 3935	-2. 3603
	AdaWeight	-2. 4191	-2. 5088	-2. 4735
Ackley	Common	-21. 6584	-21. 7131	-21. 6852
	LinearDec	-21. 6106	-21. 7239	-21. 6806
	RandWeight	-21. 4792	-21. 6555	-21. 5964
	AdaWeight	-21. 6790	-21. 7725	-21. 7146

From the analysis of Table IV and Table V, the Cooperative Particle Swarm Optimization algorithm with Cloud Mutation Operator Based on Normal Cloud Model runs more time, but the number of iterations about it are smaller than the other three algorithms. However, it can be seen from the average of the four algorithms compared that the Cooperative Particle Swarm Optimization algorithm with Cloud Mutation Operator Based on Normal Cloud Model has the smallest fitness value. Therefore, the algorithm is better than the other three algorithms in solving the optimization problem.

TABLE III: COMPARISON OF FOUR INERTIA WEIGHT PROCESSING METHODS (B)

Function	Method	Average running time
Rosenbrock	Common	1. 057
	LinearDec	1. 117
	RandWeight	1. 081
	AdaWeight	1. 532
Rastrigin	Common	1. 413
	LinearDec	1. 482
	RandWeight	1. 478
	AdaWeight	2. 272
Griewank	Common	1. 816
	LinearDec	1. 857
	RandWeight	1. 995
	AdaWeight	3. 096
Ackley	Common	1. 929
	LinearDec	2. 038
	RandWeight	3. 389
	AdaWeight	2. 239

TABLE IV: COMPARISON OF FOUR ALGORITHMS (A)

Function	Algorithm	Worst	Best	Mean
Rosenbrock	PSO	7. 313E+01	1. 926E+01	2. 911E+01
	CH-PSO	7. 678E+01	2. 405E-01	2. 701E+01
	HC-PSO	7. 857E+01	2. 373E-01	2. 207E+01
	CH-HC-PSO	7. 278E+01	2. 178E-01	2. 133E+01
Rastrigin	PSO	8. 127E+01	1. 531E+01	4. 234E+01
	CH-PSO	1. 291E+02	1. 496E+01	4. 295E+01
	HC-PSO	1. 229E+02	1. 990E+01	4. 478E+01
	CH-HC-PSO	1. 319E+02	1. 982E+01	4. 235E+01
Griewank	PSO	9. 408E-01	6. 207E-03	7. 676E-02
	CH-PSO	4. 505E-01	6. 505E-16	4. 919E-02
	HC-PSO	4. 490E-01	6. 661E-16	4. 649E-02
	CH-HC-PSO	5. 648E-01	6. 419E-16	4. 549E-02
Ackley	PSO	8. 611E+00	3. 271E-01	4. 164E+00
	CH-PSO	5. 394E-04	8. 963E-09	1. 202E-05
	HC-PSO	4. 332E-04	8. 516E-09	1. 105E-05
	CH-HC-PSO	4. 371E-04	8. 497E-09	1. 109E-05

TABLE V: COMPARISON OF FOUR ALGORITHMS (B)

Function	Algorithm	Average running time	Average number of iterations
Rosenbrock	BPSO	0. 345	23
	CH-PSO	0. 305	23
	HC-PSO	0. 323	21
	CH-HC-PSO	0. 411	20
Rastrigin	BPSO	0. 202	23
	CH-PSO	0. 234	22
	HC-PSO	0. 239	20
	CH-HC-PSO	0. 379	18
Griewank	BPSO	0. 353	40
	CH-PSO	0. 305	36
	HC-PSO	0. 324	38
	CH-HC-PSO	0. 387	26
Ackley	BPSO	0. 302	23
	CH-PSO	0. 367	21
	HC-PSO	0. 387	20
	CH-HC-PSO	0. 479	16

VII. CONCLUSION

In this paper, a Cooperative Particle Swarm Optimization algorithm with Cloud Mutation Operator Based on Normal Cloud Model is proposed. The traditional Particle Swarm Optimization algorithm is improved. The two children with non-leaf nodes in the Huffman tree are closely related. In principle, a large population is divided into two independent sub-populations, and the evolution of the particles was carried out independently, and the Huffman Cloud Model algorithm was used to achieve rapid convergence, which ensures the diversity of the population and avoids the premature convergence. In terms of parameter control, this paper uses the adaptive inertia weight strategy to make the inertia weight continuously and dynamically adjusted by the influence of current environment's factors, so as to eliminate

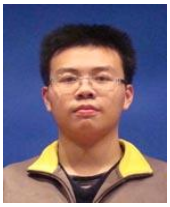
the local premature convergence and improve the search performance of the algorithm in the whole. Through the combination of Cloud Model, Huffman coding and particle swarm optimization, the algorithm achieves better performance in both macro and micro, and reduces storage space and improves storage utilization through the paradigm Huffman coding.

ACKNOWLEDGMENT

We gratefully acknowledge the valuable cooperation of the members of my laboratory in preparing this paper.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. the IEEE International Conference on Neural Networks, Piscataway, NJ: IEEE Service Center*, 1995, pp. 1942-1948.
[2] D. L. Donoho, "De-noising by soft-hresholding," *IEEE Trans. Inform Theory*, vol. 41, no. 3, pp. 613-627, 1995.
[3] D. X. Zhang, Z. H. Guan, and X. Z. Liu, "An adaptive particle swarm optimization algorithm for dynamically changing inertia weight," *Control and Decision*, vol. 11, pp. 1253-1257, 2008.
[4] Y. G. Li, W. H. Gui, C. H. Yang, and Z. S. Chen, "An elastic particle swarm optimization algorithm," *Control and Decision*, vol. 01, pp. 95-98, 2008.
[5] D. Y. Li, H. J. Meng, and X. M. Shi, "Affiliated cloud and subordinate cloud generator," *Computer Research and Development*, vol. 6, pp. 15-20, 1995.
[6] C. Z. Liu *et al.*, "Statistical analysis of normal cloud model," *Information and Control*, vol. 34, no. 2, pp. 236-239, 2005.
[7] G. W. Zhang *et al.*, "Evolutionary algorithm based on cloud model," *Chinese Journal of Computers*, vol. 31, no. 7, pp. 1082-1091, 2009.
[8] Z. H. Dai *et al.*, "Cloud genetic algorithm and its application," *Chinese Journal of Electronics*, vol. 35, no. 7, pp. 1419-1424, 2006.
[9] Y. J. Zhang, N. F. Shao *et al.*, "A cloud-based particle swarm optimization algorithm based on cloud model," *Pattern Recognition and Artificial Intelligence*, vol. 24, no. 1, pp. 90-96, 2011.
[10] H. Y. Xu, Y. B. Tian, and T. A. Huang, "Cloud adaptive particle swarm optimization algorithm based on cloud variation," *Computer Simulation*, vol. 29, no. 11, pp. 251-255, 2012.
[11] J. J. Wang, "Huffman coding synergy particle swarm optimization algorithm," *Computer and Modernization*, vol. 6, pp. 82-85, 2015.
[12] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. the IEEE International Conference on Evolutionary Computation, Piscataway*, 1998, pp. 69-73.
[13] D. Y. Li, C. W. Liu, and W. Du, "Uncertainty artificial intelligence," *Journal of Software*, vol. 15, no. 11, pp. 1583- 1594, 2004.
[14] G. H. Liu *et al*., "Software implementation of cloud generator," *Journal of Computer Applications*, vol. 24, no. 1, pp. 46-48, 2007.
[15] T. Z. Shao and D. J. Shang, "An improvement of huffman coding application — paradigm Huffman coding," *Science & Technology Innovation Review*, vol. 21, pp. 29-31, 2008.



Jiahui Luo received the B. S. degree from Guangzhou University, Guangzhou, China, in 2015.

He is currently pursuing the M. S. degree with Guangzhou University, Guangzhou, China. His main research interests include data mining, pattern recognition and intelligent optimization algorithms.

Ying Gao received the Ph. D. degree from the South China University of Technology, Guangzhou, China, in 2002.

He is currently a professor with the School of Computer Science and Educational Software, Guangzhou University, Guangzhou. His main research interests include intelligent optimization algorithms, pattern recognition and signal processing.