Particle Swarm Optimization with Comprehensive Learning & Self-adaptive Mutation

Hao Tan, Jianjun Li, Jing Huang
College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha Hunan 410000, China
E-mail: jsutanhao@126.com

***Abstract*-As a representative method of swarm intelligence, Particle Swarm Optimization (PSO) is an algorithm for searching the global optimum in the complex space through cooperation and competition among the individuals in a population of particle. But the basic PSO has some demerits, such as relapsing into local optimum solution, slowing convergence velocity in the late evolutionary. To solve those problems, an particle swarm optimization with comprehensive learning & self-adaptive mutation(MLAMPSO) was proposed. The improved algorithm made adaptive mutation on population of particles in the iteration process, at the same time, the weight and learning factors were updated adaptively. It could enhance the ability of PSO to jump out of local optimal solution. The experiment results of some classic benchmark functions show that the improved PSO obviously improves the global search ability and can effectively avoid the problem of premature convergence.**

Keywords-Particle Swarm Optimization, adaptive mutation, weight, learning factors, convergence

I. INTRODUCTION

Particle Swarm Optimization(PSO) is an evolutionary algorithm of the global search resulting from the research on the social behaviors of birds' flocking or fish schooling, which is proposed by Dr Kennedy and Eberhar in 1995[1]. Compared with other algorithms, PSO has a lot of advantages, such as fast convergence speed, self learning and memory ability, parallel processing ability and simple coding[2]. Therefore it is popular as soon as it is proposed. It has been widely used in many related engineering fields, such as function optimization, neural network training, fuzzy system control and so on[3-5]. However PSO is also easy to be premature and to fall into the local optimum, which led to a lot of difficulties in many applications. To solve these problems, many scholars put forward the improved algorithms: ① The adjustments and constraints on relative parameters[6,7]; ②The binary particle swarm optimization algorithm[8]; ③ The genetic particle swarm optimization algorithm[9]; ④The cooperative particle swarm algorithm[10]; ⑤ The hybrid particle swarm algorithm[11].

For each particle in the group, the position and velocity are used as the particles' characteristics to describe them, which represents a possible solution of the solved problem. The coordinate of the position and velocity of each particle are used to calculate the fitness value by the appropriate objective functions, which is used as the basis for measuring the global optimal solution. This paper proposes a new improved PSO algorithm--Particle Swarm Optimization with comprehensive learning & self-adaptive mutation(MLAMPSO), which can well solve the defects of falling into local optimum solution. The mutation probability of the particle is calculated by the particle swarm fitness variance in the process of each iteration. At the same time, weight and learning factors are updated adaptively. Therefore, particles developed their own special skill and learned from the optimal particle as well, which can effectively avoid the particle swarm optimization algorithm falling into local optimal solution.

II. BASIC PARTICLE SWARM OPTIMIZATION ALGORITHM (PSO) AND ITS EARLY MATURITY

The flock foraging behavior is simulated in PSO algorithm[6]. It considered each solution of optimization problem as a bird which is called "particle". The individual position moving is considered as the process of seeking the optimal location, which is also called as the process of seeking the global optimal solution. The vector of particle i can be represented as:

$$X_i = (x_{i1}, x_{i2}, \cdots, x_{iD}), i = 1,2,...,N \quad (1)$$

Where the target search space dimension is represented by D, the particle number is represented by N.

The flight speed of particle i is also a vector of D dimensions, which is represented as:

$$V_{\mathrm{i}} = (v_{i1}, v_{i2}, \cdots, v_{iD}), i = 1,2,...,N \quad (2)$$

The optimal location of particle i is searched in current known as the individual extremum, which is represented as:

$$p_{best} = (p_{i1}, p_{i2}, \cdots, p_{iD}), i = 1,2...,N \quad (3)$$

The optimal location of the whole particle swarm is searched in current known as the global optimal solution, which is represented as:

$$g_{best} = (p_{g1}, p_{g2}, \cdots, p_{gD}) \quad (4)$$

The position and speed of each particle are updated by the following iterative formulas put forward by Kennedy and Eberhart, which are represented as:

$$\mathrm{v}(t+1) = \omega v(t) + c_1 r_1 (p_{best} - x(t)) + c_2 r_2 (g_{best} - x(t)) \quad (5)$$

$$x(t+1) = x(t) + v(t+1) \quad (6)$$

Where the current evolutionary iteration number is represented by t. The inertia weight is represented by ω (generally $\omega = 1$). p_{best} is the optimal location of particle itself in the process of iteration t in history. g_{best} is the global optimal location for all the particles. c_1 and c_2 ($c_1 > 0, c_2 > 0$) are the acceleration constant and according to the experience they usually take the value of 2. r_1 and r_2 are the random number which are in [0,1]. Y Shi et al took a large of experiments to show that if the inertia weight decreased linearly in the process of iteration, the convergence of the algorithm would be significantly enhanced. Therefore the inertia weight updating formula can be represented as:

$$\omega = \omega_{\max} - \frac{Iter_i}{\mathrm{Iter}}(\omega_{\max} - \omega_{\min}) \qquad (7)$$

Where $\omega_{\max}$ is maximum inertia weight and $\omega_{\min}$ is minimum inertia weight. $\mathrm{Iter_i}$ is the current iteration number and Iter is the number of total iterations.

Generally particle swarm optimization algorithm uses real number to encode. So there is no complex operation like selection, crossover and mutation operation of genetic algorithm. Compared with genetic algorithm, the structure is relatively simple and running speed is relatively fast. However, when one of the particles found the current optimal position, other particles would be closed to the optimal position quickly. If the current optimal position is not the global optimal solution, the particle swarm can't search again in the solution space. Algorithm will be falling into local optimal solution, thus it will appear the so-called premature convergence phenomenon[12, 13].

III. THE IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM(MLAMPSO)

From the basic particle swarm optimization algorithm, we can see that the position of the particle in the next moment is controlled by the particle's current velocity and current location. While the current velocity of particle depends on originally velocity, individual extremum and global extremum of the particle. The global extremum is the optimal solution for particle in current. If the algorithm appeared premature convergence, the result must be a local optimal solution. The distance and direction of particle moving are controlled by the velocity and its direction respectively. Therefore, changing the velocity and its direction can make the particles to search in other areas again. At the same time, the particles also learn themselves adaptively and update the adaptive learning factors constantly. The velocity and position of particles are limited, which can avoid the particles flying out of the solution space area. In the subsequent searching process, the algorithm will seek a new individual extremum as well as the global extremum until the global optimal solution is found.

Assuming that the number of the whole particle swarm is n . $\mathrm{f_i}$ is the fitness value of particle i . $\mathrm{f_{avg}}$ is the current average fitness value of all particles. σ^2 is the current particle swarm fitness variance, which is represented as:

$$\sigma^2 = \frac{1}{n-1}\sum_{\mathrm{i=1}}^{n}\left(\frac{f_i - f_{avg}}{f}\right)^2 \qquad (8)$$

$$f = \begin{cases} \max\{|f_i - f_{avg}|\}, \max\{|f_i - f_{avg}|\} > 1 \\ 1, others \end{cases} \qquad (9)$$

Where f is the normalized factor, which mainly takes the constraint on the current particle swarm fitness variance.

Since the particles would find a better position under the effect of the current global extremum, the algorithm proposed in this paper would take a mutation operation at a certain probability. The mutation probability formula can be represented as:

$$p_m = \begin{cases} p, \sigma^2 < \sigma^2_{\max} and f(g_{best}) > f_d \\ 0, others \end{cases} \qquad (10)$$

Where p is the random number which are in [0.1, 0.2]. σ^2_{max} is the largest particle swarm fitness variance in all the past iterations. f_d is the theory optimal value of the fitness function.

This paper uses the method of random disturbance to take a mutation operation on the current global extremum, which is represented as:

$$g_{best_{\mathrm{k}}} = g_{best_{\mathrm{k}}} * (1 + \eta * p_m) \qquad (11)$$

Where η is the random number which are in [-1, 1].

Learning factors c_1 and c_2 are updated by the adaptive dynamic method. If they are too large, it would lead to flying away from the target area in the searching process. Therefore appropriate learning factors can accelerate convergence speed, it can avoid to falling into the local optimal solution as well. Learning factors updating formula can be represented as:

$$\mathrm{c}_i^{k+1} = c_{\min} + \frac{Iter^{\mathrm{k}}}{Iter}\left(c_i^k - \frac{c_{\min} + c_{\max}}{2}\right), \mathrm{i} = 1{,}2 \qquad (12)$$

There is no doubt that the velocity needs changing when it is too high (such as $\mathrm{v}_{id}^{k+1} \notin [-v_{\mathrm{d}\max}, v_{d\max}]$), which can avoid the particles flying out of target area. The velocity updating formula is represented as:

$$\mathrm{v}_{id}^{k+1} = \mathrm{v}_{id}^{k+1} - \alpha \mathrm{v}_{d\max} \qquad (13)$$

Where α is the random number which is in [-0.5, 0.5].

Similarly, when the particle flies away from the target area (such as $x_{id}^{k+1} \notin [-x_{\mathrm{d}\max}, x_{d\max}]$), the position updating formula is represented as:

$$x_{id}^{k+1} = x_{id}^{k+1} - \delta x_{d\max} \qquad (14)$$

Where δ is the random number which is in [-0.5, 0.5].

From what has been discussed above, this paper uses the variance analysis to take mutation on the current global extremum. At the same time, learning factors are updated adaptively, the velocity is limited and solution space area is controlled. Particle swarm optimization with comprehensive learning & self-adaptive mutation (MLAMPSO) is proposed in this paper. Its process is as follows:

Step 1 Initialize the particle's position and velocity in the particle swarm and other related parameter values;

Step 2 Initialize the current position as p_{best} and the best position of all the particles as g_{best};

Step 3 If the particle swarm fitness variance is tending to zero and the optimal solution is tending to the theory optimal solution it will turn to Step 7, or Step 4-6 will be executed;

Step 4 Take the following operations for all the particles:

① Update parameters by the formula (7) and (12), update velocity and position of the particle by the formula (5) and (6), calculate the fitness value of the particle;

② If the fitness value of a certain particle is superior than the individual extremum it will be updated by the formula (3) ;

③ If the fitness value of a certain particle is superior than the global optimal solution it will be updated by the formula (4) ;

Step 5 Calculate the particle swarm fitness variance by the formula (8) and (9), calculate the mutation probability of the particle by formula (10);

Step 6 Decide whether a mutation operation will be took or not, if it is, mutation operation will be took by formula (11) and turn to Step 3;

Step 7 Output the result and the algorithm is over.

IV. THE EXPERIMENTAL RESULTS

This paper verified the effectiveness of the improved algorithm on Windows8 operating system by using experiment simulation platform Matlab7.1. Three kinds of test experiments are took: ① PSO experiment; ② AMPSO optimization experiment[15]; ③ MLAMPSO optimization experiment. It used four typical functions to test the algorithm of this paper, which are shown in table 1, including function form, dimension and searching scope. The experimental results are shown in table 2. The fitness curves of three algorithms are shown in Fig1-4: Fig 1 shows that the three algorithms all get the ideal results. Fig 2-4 show that AMPSO and PSO are falling into a local optimal solution in the optimization process while the fitness value of MLAMPSO is much closed to the desired results. It broke the deadlock falling into a local optimal solution. Therefore, MLAMPSO algorithm is superior to AMPSO and PSO. The optimization results are becoming more and more accurate.

TABLE I. THE DESCRIPTIONS OF TEST FUNCTIONS

Functions	***Formulas***	***Dimension***	***Scope***
Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2]$	30	$[-100,100]^{30}$
Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-100,100]^{30}$
Rastrigrin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	30	$[-100,100]^{30}$
Ackly	$f(x)=-20\exp(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	30	$[-100,100]^{30}$

TABLE II. THE OPTIMAL SOLUTION OF THREE ALGORITHMS

Functions	***Theory optimal solution***	***PSO***	***AMPSO***	***MLAMPSO***
Rosenbrock	0	0.1362	0	0
Griewank	0	0.7973	0.5538	0.1129
Rastrigrin	0	2.2132	1.8434	1.3236
Ackly	0	0.3142	0.2876	0.0142

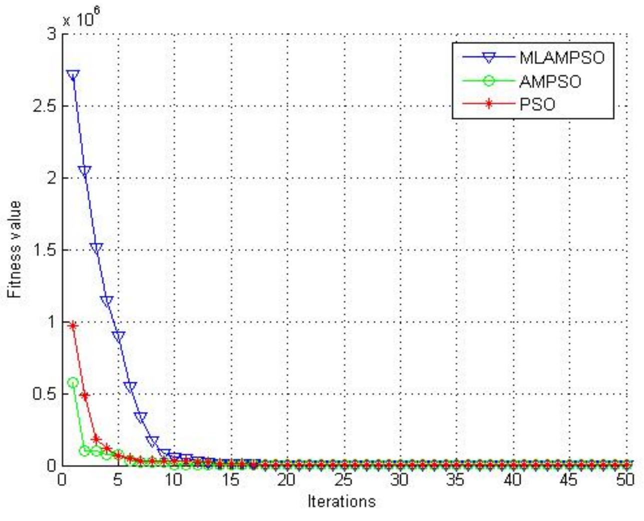

Fig. 1. The fitness curve of Rosenbrock

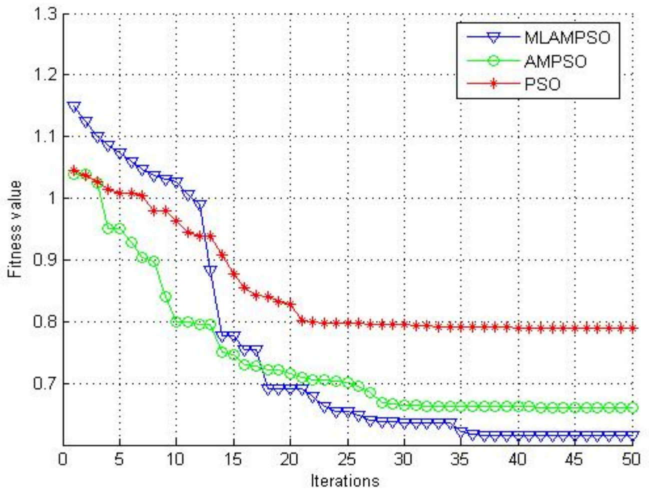

Fig. 2. The fitness curve of Griewank

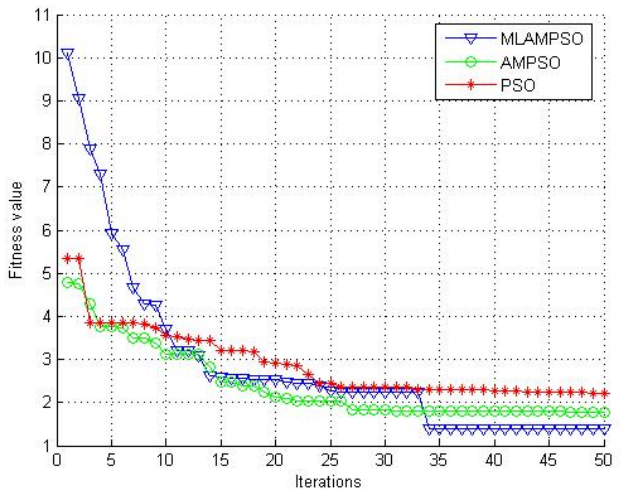

Fig. 3. The fitness curve of Rastrigrin

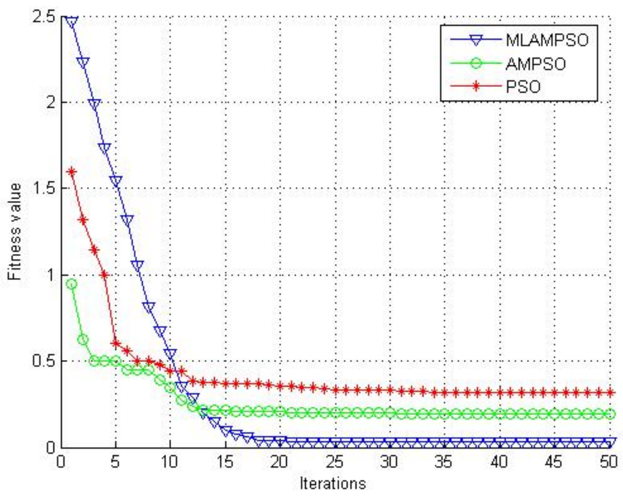

Fig. 4. The fitness curve of Ackly

V. Conclusion

This paper proposes a multidirectional learning self-adaptive mutation particle swarm optimization algorithm. It makes full use of the diversity and adaption of swarm intelligence in individual learning, which is based on the basic particle swarm optimization algorithm. At the same time, it combines with the method of variance analysis and other improved excellent particle swarm optimization algorithms take part in it. The defect of falling into local optimal solution has been well solved and the robustness of the algorithm has been enhanced greatly. It takes experiments using four international classic optimization functions. The experiment results show that the multidirectional learning adaptive mutation particle swarm optimization algorithm can easily avoid falling into local optimal solution in the process of iterative optimization. The ability of jumping out of local optimum is enhanced obviously. It can get better optimization results with much smaller population and evolutionary iterations. The stability of the solution convergence has been improved as well.

References

[1] Kennedy J, Eberhart R C. “Particle swarm optimization.” Proc of IEEE International Conference on Neural Networks. USA : IEEE Press,1995:1942-1948.

[2] Liang J J, Qin A K, Suganthan P N, et al. “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions.” IEEE Transactions on Evolutionary Computation,2006,10 (3):281-295.

[3] Meissner M, Schmuker M, Schneider G. “Optimized ParticleSwarm Optimization（OPSO）and its application to artificial neural network training.” BMC Bioinformatics, 2006, 7:125-130.

[4] Zhou C, Gao HB, Gao L, et al. “Particle swarm optimization (PSO)algorithm.” Appl Res Comput 2003;20(12):7－11.

[5] E. Bonabeau, M. Dorigo, G. Theraulaz. “Swarm Intelligence: From Natural to Artificial Systems.” Oxford University Press, New York, 1999, 13:224-230.

[6] Shi Y, et al. “A modified particle swarm optimizer.” In: IEEE World Congress on Computational Intelligence, 1998, 69-73

[7] Shi Y, Eberhart R C. “Fuzzy Adaptive particle swarm optimization.” In: Proc of the Congress on Evolutionary Computation, Seoul Korea, 2001,123-129

[8] J Kennedy, R C Eberhart. “A discrete binary version of the particle swarm algorithm.” Piscataway, NJ: IEEE Service Center, 1997,33(4):126-132.

[9] Yi-Tung Kao, Erwie Zahara. “A hybrid genetic algorithm and particle swarm optimization for multimodal functions.” Applied Soft Computing ,2008,8:849–857

[10] Van den Bergh F, Engelbrecht A P. “Effcets of swarm size cooperative particle swarm optimizers.” In: Proc of the third Genetic and Evolutionary computation conf, San Francisco, USA, 2001,1124-1130.

[11] Guobin Gong. “Adaptive constrained optimization hybrid Particle Swarm Optimization algorithm.” Computer Engineering and Applications, 2013, 49(9):50-53

[12] Fu Duan, Tongfen Su. “Modified immune particle swarm optimization algorithm and its application.” Journal of Computer Applications, 2010,30(7):1883-1884.

[13] Baoning Liu, Weiguo Zhang, Guangwen Li, Rui Nie. “Improved multi-objective particle swarm optimization algorithm.” Journal of Beijing University of Aeronautics and Astronautics, 2013, 39(4):458-462

[14] Zhensu Lv, Zhirong Hou. “Particle Swarm Optimization with Adaptive Mutation.” Acta Electronica Sinica, 2004, 32(3):416-420