# Particle Swarm Optimization and Cuckoo Search Paralleled Algorithm

Yang Xiaodong
Electronic and Information Engineering Deptartment
Shunde Polytechnic
Shunde, China
e-mail: 653488297@qq.com

Cai Zefan
Electronic and Information Engineering Deptartment
Shunde Polytechnic
Shunde, China
e-mail: 12266423@qq.com

***Abstract*—Particle swarm optimization algorithm and cuckoo search algorithm both are bionic swarm optimization algorithms, which are simple and convenient. They have been applied to many fields. However, the algorithms have obvious disadvantages. When they are applied to complex optimization problems, they cannot obtain the optimal solutions, so some measures must be adopted in order to improve their global search ability. In this paper, particle swarm optimization algorithm and cuckoo search algorithm evolve in parallel. At the end of each generation, the better solution of the two algorithms is selected as the global optimal solution. The simulation results show that the paralleled algorithm absorbs the advantages of the two algorithms, improves the global search ability and the average convergence speed, and enhances the robustness of the algorithm. The new algorithm is able to solve complex optimization problems more efficiently.**

***Keywords-cuckoo search algorithm; particle swarm optimization algorithm; paralleled algorithm***

## I. INTRODUCTION

Particle swarm optimization algorithm [1-3] (PSO) is a swarm intelligence algorithm to simulate the foraging behavior of birds, which was proposed by Kennedy J and Eberhart R C in 1995. PSO includes a speed update equation and a position update equation. It just needs to adjust a few parameters, and its program is easy to realize.

Cuckoo search algorithm (CS) [4-8] is a kind of heuristic algorithm, which is inspired by the young bird feeding behavior of the cuckoo in a parasitic way, and contains the Lévy flight behavior of birds and drosophila. It was jointly developed by Yang Xinshe in Cambridge University and Deb Suash in C. V. Roman Engineering Institute in 2009. [3]

Thanks to their simplicity and effectiveness, they have attracted the attention of scholars from the date of their birth and have been applied to many fields successfully [7-18]. The start convergence speed of the standard PSO is very fast, but its global search ability is weak. While the global search ability of CS is better, but its convergence speed is slow. In some complex problems, both PSO and CS cannot find the global optimal solutions. Their performances need to improve. This paper proposes PSO and CS paralleled algorithm (PSOCSPA), which absorbs the advantages of PSO and CS and introduces a sharing mechanism of two algorithms. PSOCSPA obviously improves the global search ability and convergence speed.

## II. PARTICLE SWARM OPTIMIZATION ALGORITHM

The basic idea of PSO is to randomly initialize a population of ideal particles, which have no volume or quality. Each particle is considered as a feasible solution for the optimization problem. Advantages and disadvantages of particles are evaluated by fitness functions. Each particle will move in the feasible solution space, and the direction and distance will be determined by a velocity variable. Usually the particles will follow the current optimal particle in every generation, and finally reach the best place. In each generation, the particles will follow two extremes: the optimal solution of itself and the optimal solution of the entire population. Assuming that the particle swarm population size is M and the search space dimension is D, the state property of particle i at time t is set as follows.

Position: $x_i^t = \left(x_{i1}^t, x_{i2}^t, \cdots, x_{id}^t\right)^T$, $x_{id}^t \in \left[L_d,\ U_d\right]$,

$L_d$ *and* $U_d$ is the lower and upper bounds of the search space;

Speed: $v_i^t = \left(v_{i1}^t, v_{i2}^t, \cdots, v_{id}^t\right)^T$, $v_{id}^t \in \left[v_{\min,d},\ v_{\max,d}\right]$,

$v_{\min}$ *and* $v_{\max}$ is the minimum speed and maximum speed;

Individual particle optimal position:

$p_i^t = \left(p_{i1}^t, p_{i2}^t, \cdots, p_{id}^t\right)^T$;

Global particle optimal position:

$p_g^t = \left(p_{g1}^t, p_{g2}^t, \cdots, p_{gd}^t\right)^T$, where $1 \le d \le D$, $1 \le i \le M$.

The position of particle *i* at the time *t*+1 is updated with Eq. (1) and (2).

$$v_{id}^{t+1} = v_{id}^t + c_1 r_1 \left(p_{id}^t - x_{id}^t\right) + c_2 r_2 \left(p_{gd}^t - x_{id}^t\right). \tag{1}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1}. \tag{2}$$

where, $r_1$ and $r_2$ is random numbers in evenly distributed [0, 1]; $c_1$ and $c_2$ are learning factors, generally, $c_1 = c_2 = 2$.

There are 4 steps in the original particle swarm optimization algorithm.

Step 1: PSO initialization. Set the initial value of various parameters of PSO. The parameters are $L_d$ and $U_d$, $c_1$ and

978-1-5090-6352-9/17/$31.00 ©2017 IEEE

$c_2$, maximum Iterations $T_{max}$, Convergence precision $\xi$, the speed scope $[v_{min}, v_{max}]$, $x_i^0$ and $v_i^0$. Calculate the fitness value $F_i$ of particle i, set $F_i$ as its extreme fitness value $F_{iBest}$, and set $x_i^0$ as its best position $p_i$. Set the best $F_{iBest}$ as the global extreme fitness value $F_{gBest}$, and record the corresponding $p_i$ as the global best position $p_g$ and i as the index number g of $p_g$.

Step 2: PSO evaluation. Calculate $F_i$ of each particle. Update $F_{iBest}$ with $F_i$, and $p_i$ with $x_i^t$, if $F_i$ is better than $F_{iBest}$. Update $F_{gBest}$ with the best $F_{iBest}$, $p_g$ with the corresponding $p_i$, and g as i, if the best $F_{iBest}$ is better than $F_{gBest}$.

Step 3: PSO Update. Update the speed of each particle with Eq. (1) and position with (2). Make the speed in the scope $[v_{min}, v_{max}]$.

Step 4: Ending judgment. If the end conditions reach, then stop, otherwise return to step 2.

## III. CUCKOO SEARCH ALGORITHM

When Xin-She Yang and Suash Deb described CS, they defined three rules as follows [4-6].

Rule 1: Each cuckoo can just lay one egg at a time, and dumps it in a random nest;

Rule 2: The best nests with high-quality eggs will be chosen in the next generation;

Rule 3: The number of available host nests is fixed, and the egg laid by another cuckoo will be discovered by the host bird with a probability, and the host bird will either get rid of the egg, or simply abandon the nest and build a completely new nest.

Based on these three rules, the concrete steps of CS are as follows:

(1) Initialization. Randomly generate an initial population of n nests at the position $X_0 = \left(x_1^0, x_2^0, \cdots, x_N^0\right)$, then evaluate their fitness values $F_0$ and find the current global best one.

(2) Search. Update the positions with the new positions $X_t = \left(x_1^t, x_2^t, \cdots, x_N^t\right)$ by Lévy flights, then evaluate their fitness values $F_t$ and find the current best global value. Record the best global value and its corresponding position.

(3) Selection. Choose a random number *r* from a uniform distribution [0, 1]. Update $X_{t+1}$ if $r > p_a$. Then evaluate their fitness values $F_t$ again and find the current best global value. Record the best global value and its corresponding position.

(4) Judgment. If the stop criterion is met, then the best global position is found. Otherwise, return to step (2).

In search step, Lévy flights use the random walk strategy shown as in [6]

$$X_{t+1,i} = X_{t,i} + \alpha \oplus Le'vy(\beta). \tag{3}$$

where $X_{t,i}$ represents i[th] solution in t[th] generation; $\alpha$ is the step size used to control the range of random search. A bigger $\alpha$ is better for global search and a smaller $\alpha$ is better for local search. In the whole search, $\alpha$ should change from big to small. In general, $\alpha$ is limited in the fixed range, $\alpha = O(L/10)$, where L is the search range. Reference [4] adopted the step size shown as in

$$\alpha = \alpha_0 \left(X_{t,i} - X_{best}\right). \tag{4}$$

where, $\alpha_0$ is constant which is set to 0.01 in the most time; $X_{best}$ is the current global optimal solution.

In (3) $\oplus$ represents point multiplication; $Le'vy(\beta)$ obeys Lévy probability distribution as in

$$Le'vy(\beta) \sim u = t^{-1-\beta}, 0 < \beta \le 2. \tag{5}$$

Reference [6] adopted (9) to calculate $Le'vy(\beta)$.

$$Le'vy(\beta) \sim \frac{\phi \times u}{|v|^{1/\beta}}, \beta = 1.5. \tag{6}$$

where $u$ and $v$ are the standard normal distribution numbers. $\phi$ is as in

$$\phi = \left( \frac{\Gamma(1+\beta) \times \sin(\pi \times \beta/2)}{\Gamma\left(\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{(\beta-1)/2}\right)} \right)^{1/\beta}. \tag{7}$$

According to (3)~(7), the update equation in Lévy flights is shown as in

$$X_{t+1,i} = X_{t,i} + \alpha_0 \frac{\phi \times u}{|v|^{1/\beta}} \left(X_{t,i} - X_{t,best}\right). \tag{8}$$

In selection step, after some bad positions are discarded, the same number of new positions will be produced by (9).

$$X_{t+1,i} = X_{t,i} + r\left(X_{t,j} - X_{t,k}\right). \tag{9}$$

where *r* is a scaling factor, which is a random number in uniform distribution [0, 1]. $X_{t,j}$ and $X_{t,k}$ are two solutions in t[th] generation.

In the above steps, CS algorithm not only uses the Lévy flights search method, but also introduces the elite retention strategy. A good combination of local search and global search is obtained in this algorithm. The selection step has increased the diversity of positions. In the case of sufficient number of iterations, CS can converge to the global optimal solution in probability 1. [6]

## IV. PSO AND CS PARALLELED ALGORITHM

The adaptability of single bionic swarm optimization algorithm is limited. Some algorithm is successful in solving a problem, but it may be unsuitable for other problem. For example, in the 6 standard test functions simulation, CS can obtain the optimal solutions of 5 functions in a probability 1 and misses the optimal solution of another function. Instead, PSO can obtain the optimal solution of the function in a probability 1, which cannot be obtained by CS. In addition, the start convergence speed of PSO is very fast, but the latter speed become slow, while the convergence speed of CS is moderate in the whole evolutionary cycle. The mixed algorithm can overcome the disadvantages of PSO and CS and improve the overall performance.

### *A. PSOCSPA Model*

In PSOCSPA, PSO and CS evolve in parallel. In the end of each generation, the better solution of PSO and CS will be selected as the global optimal solution. The flow of PSOCSPA is shown as in Fig. 1. There are 5 steps:

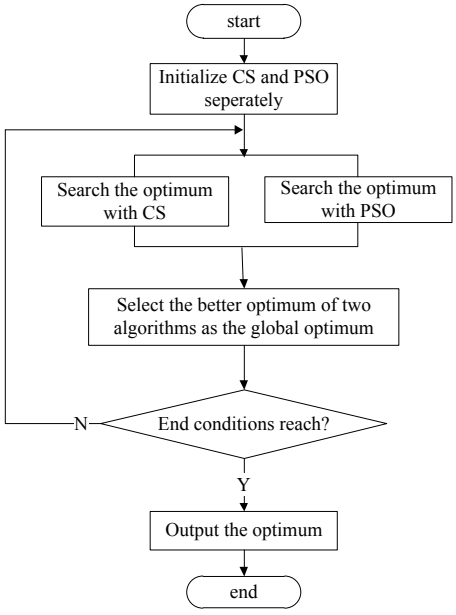


Figure 1. Flow chart of PSOCSPA.

(1) Initialize. Initialize PSO and CS separately. The population sizes, individual dimensions and object functions are same.

(2) Evolve in parallel. PSO and CS separately evolve and search for the optimal solution. That is to say that the two algorithms will finish the complete evolution alone in each iteration. There is no primary and secondary point. PSO and CS both have their own population optimums.

(3) Save the global optimal solution. In the end of each generation, select the better solution of PSO and CS as the global optimal solution.

(4) Judge algorithm process. If the iterative termination conditions are satisfied, then go to next step, or return to (2).

(5) Output the result. Output the global optimal solution and exit.

### *B. PSOCSPA Simulation*

This section compares PSOCSPA with PSO and CS in order to verify the validity. The simulation platform is WIN7+MATLAB R2012b. The 6 test functions are shown in Table Ⅰ, where $f_1 \sim f_3$ are single-modal functions and $f_4 \sim f_6$ are multi-modal functions.

During the simulation, the parameters are as follows:

Population size $N = 30$, individual dimension $D = 20$, the range of each dimension is $[-20, 20]$, convergence precision $\xi = 10^{-5}$, and maximum iteration number $t_{\max} = 2000$.

Each simulation runs 30 times separately. The simulation result is shown in Table Ⅱ.

As can be seen from Table Ⅱ, in all the 6 standard test functions simulation, PSOCSPA can converge to the set precision within the iterations in a probability 100%, while PSO and CS both cannot do this. The average convergence iteration number is a little bigger than PSO and much smaller than CS. PSOCSPA absorbs the advantages of both PSO and CS through the parallel search. Both the global search ability and convergence speed are improved a lot.

Take $f_1$ and $f_4$ as examples, PSO, CS and PSOCSPA evolve for 2000 iterations, the average best fitness curves of 30 simulations are shown in Fig. 2 and Fig. 3. As can be seen from Fig. 2, the fitness of PSO is better than that of CS in the start 1100 iterations and the situation reverses after 1100 iterations. The fitness of PSOCSPA is nearly the same as PSO in the start 1100 iterations, and nearly the same as CS after 1100 iterations. As can be seen from Fig. 3, CS is unable to converge to the setting accuracy. It appears premature convergence and stagnation behavior. PSO and PSOCSPA converge quickly to a high accuracy.

TABLE I. STANDARD TEST FUNCTIONS

| function | formula | dimension | scale | optimum |
|---|---|---|---|---|
| Sphere | $f_1(x)=\sum_{j=1}^{n}(x_j^2)$ | 20 | $[-20,20]$ | 0 |
| Quarticfunction | $f_2(x)=\sum_{j=1}^{n}jx_j^4$ | 20 | $[-20,20]$ | 0 |
| Schwefel | $f_3(x)=\sum_{j=1}^{n}\lvert x_j\rvert+\prod_{j=1}^{n}\lvert x_j\rvert$ | 20 | $[-20,20]$ | 0 |
| Rastrigin | $f_4(x)=\sum_{j=1}^{n}\left(x_j^2-10\cos(2\pi x_j)+10\right)$ | 20 | $[-20,20]$ | 0 |
| Griewank | $f_5(x)=\frac{1}{4000}\sum_{j=1}^{n}x_j^2-\prod_{j=1}^{n}\cos\left(\frac{x_j}{\sqrt{j}}\right)+1$ | 20 | $[-20,20]$ | 0 |
| Ackley | $f_6(x)=-20*\exp\left(-0.2*\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}\right)-\exp\left(\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)\right)+20+\exp(1)$ | 20 | $[-20,20]$ | 0 |

TABLE II. SIMULATION RESULT OF PSOCSPA, PSO AND CS

| test function | algorithm | success times (average success iteration) | average optimal value | standard deviation of optimal value |
|---|---|---|---|---|
| $f_1$ | PSO | 29（30） | 1.3333e+01 | 7.3030e+01 |
| | CS | 30（412） | 8.9362e-06 | 1.0158e-06 |
| | PSOCSPA | 30（64） | 6.6424e-07 | 2.1005e-06 |
| $f_2$ | PSO | 29（36） | 5.3333e+03 | 2.9212e+04 |
| | CS | 30（347） | 8.3972e-06 | 1.4807e-06 |
| | PSOCSPA | 30（31） | 7.8099e-07 | 2.4697e-06 |
| $f_3$ | PSO | 15（54） | 1.4000e+01 | 1.6733e+01 |
| | CS | 30（1044） | 9.4701e-06 | 5.0536e-07 |
| | PSOCSPA | 30（445） | 3.8424e-06 | 5.9581e-06 |
| $f_4$ | PSO | 30（31） | 0 | 0 |
| | CS | 0 | 1.6386e+01 | 5.2497e+01 |
| | PSOCSPA | 30（33） | 0 | 0 |

TABLE II. SIMULATION RESULT OF PSOCSPA, PSO AND CS (CONTINUE)

| test function | algorithm | success times (average success iteration) | average optimal value | standard deviation of optimal value |
|---|---|---|---|---|
| $f_5$ | PSO | 27（65） | 1.4534e-02 | 5.6581e-02 |
| | CS | 30（576） | 9.1673e-06 | 6.1861e-07 |
| | PSOCSPA | 30（97） | 9.6986e-07 | 3.0670e-06 |
| $f_6$ | PSO | 29（32） | 3.9411e-01 | 2.1586e+00 |
| | CS | 30（718） | 9.2997e-06 | 5.7136e-07 |
| | PSOCSPA | 30（309） | 3.8414e-06 | 5.9641e-06 |

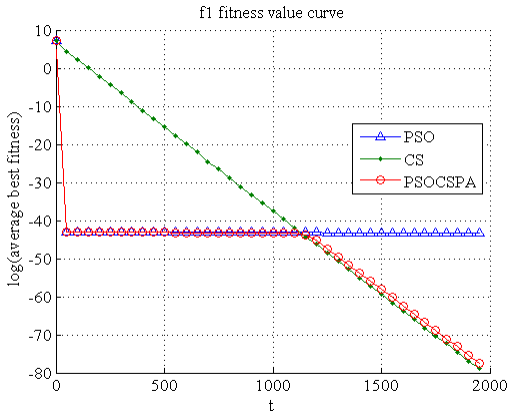


Figure 2. Average best fitness curve of function $f_1$.

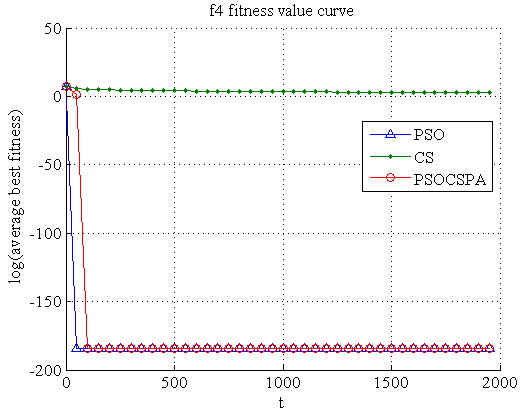


Figure 3. Average best fitness curve of function $f_4$.

## V. CONCLUSIONS

Single bionic swarm optimization algorithm cannot adapt to all the optimization problems. Different optimization algorithms have their own suitable occasions. It can avoid the weaknesses and improve the adaptability to mix two different bionic swarm optimization algorithms. PSOCSPA combines the advantages of both CS and PSO. The simulation results of 6 standard test functions show that the performance of PSOCSPA is obviously better than PSO and CS. The global search ability and convergence speed of CSAPSO are superior and it shows good robustness.

## REFERENCES


[1] J. Kennedy, R. Eberhart, "Particle swarm optimization," IEEE International Conference on Neral Networks, 1995, pp. 1942-1948.

[2] Y. Shi, R. Eberhart, "A Modified Particle Swarm Optimizer," Proceedings of the IEEE World Congress on Computational Intelligence, 1998, pp. 69-73.

[3] Y. Shi, R. Eberhart, C. Empirical, "Study of Particle Swarm Optimization," Proceeding of the World Multiconference on Syatemies, Cybernetics and Informatics, Orando, FL, 2000, pp. 1945-1950.

[4] X. S. Yang, Nature-Inspired Metaheuristic Algorithm. Luniver Press, 2008.

[5] X. S. Yang, Cuckoo Search and Firefly Algorithm Theory and Applications. Springer, 2014.

[6] X. S. Yang, S. Deb, "Cuckoo Search via Le´vy flights," Proceeding of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), pp. 210-214.

[7] X. S. Yang, S. Deb, "Engineering optimization by cuckoo search," Int. J. Math. Model. Num.Opt. 2010, vol. 1, no. 4, pp. 330-343.

[8] X. S. Yang, S. Deb, "Multiobjective cuckoo search for design optimization," Comput. Oper. Res. 2013, vol. 40, no. 6, pp. 1616-1624.

[9] A. H. Gandomi, X. S. Yang, A. H. Alavi, "Cuckoo search algorithm: a meteheuristic approach to solve structural optimization problems," Engineering with Computers, 2013, vol. 29, no. 1, pp. 17-35.

[10] A. H. Gandomi, X. S. Yang, S. Talatahari, S. Deb, "Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization," Comput. Math. Appl. 2012, vol. 63, no. 1, pp.191-200.

[11] H. Jiang, R. Qi, N. Meiling, W. Zhang, "Modified cuckoo search algorithm and its application to optimization in multiple-effect evaporation," Computers and Applied Chemistry, 2014, vol. 31, no. 11, pp. 1363-1368.

[12] S. L. He, J. H. Han, "Parameter Selection of Support Vector Regression Based on Cuckoo Search Algorithm", Journal of South China Normal University (Natural Science Edition), 2014, vol. 46, no. 6, pp. 33-39.

[13] L. Li, B. Niu, Particle swarm optimization algorithm. Metallurgical Industry Press, 2009.

[14] T. Rui, J. W. Zhu, X. S. Jiang. "Solving PUMA robot inverse kinematics based on simulated annealing particle swarm optimization," Computer Engineering and Applications, 2010, vol. 46 no. 3 pp. 27-29.

[15] Y. Li, J. J. Wang, L. H. Cao, "Simulated Annealing Particle Swarm Optimization Algorithm of Optimal Load Dispatch in Power Plan," Proceedings of the CSU-EPSA, 2011, vol. 23, no. 3, pp. 40-44.

[16] Y. Zheng, W. M.. Cheng, Y. Cheng, X. Wu. "Reliability Optimization for Cylindrical Gear Reducer based on Simulate Anneal Particle Swarm Optimization," Mechanical Drive, 2010, vol. 34, no. 10, pp. 43-47.

[17] Y. Sun, X. K. Wang, D. B. Guo, Y. H. Zhang, "The Optimized Algorithm for Data Reconciliation Based on Particle Swarm Optimization," Journal of Test and Mearsurement Technology, 2015, vol. 29, no. 2, pp. 149-157.

[18] W. Li, L. Zhou, X. L. Chai, "Joint Application of SA-PSO and PDA in Sensor Registration," J. Zhengzhou Univ. (Nat. Sci. Ed.), 2015, vol. 47, no. 1, pp. 69-73.