Cuckoo Search Algorithm with Deep Search

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

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

***Abstract*—Cuckoo search algorithm (CS) is a kind of bionic swarm optimization algorithm, which is simple and convenient. Although it has obvious advantages, it cannot converge to the optimal solution when dealing with high dimensional complex problems, so its global search ability needs to be improved. In this paper, opposition-based learning (OBL) strategy and local enhanced search are introduced to improve the basic CS. In the selection stage, an opposition-based swarm (ob-swarm) is generated on the basis of the original swarm. If an old cuckoo individual from the original swarm may be discarded, three cuckoo individuals are randomly selected from the ob-swarm to produce a new individual. The old one will be replaced by the new one if the new one is better. At the end of each generation, a potential optimal solution will be searched for locally around the current global optimal solution in the evolution direction. This local searching operation can make up the problem that the search step may be not appropriate. The simulation results show that the improved algorithm improves the global search ability, convergence speed and convergence precision of the algorithm.**

Keywords-cuckoo search algorithm; bionic swarm optimization algorithm; paralleled algorithm; opposition-based learning; deep search

I. INTRODUCTION

Cuckoo search algorithm (CS) [1-5] 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 Xinshe Yang in Cambridge University and Suash Deb in C. V. Roman Engineering Institute in 2009. [3]

Thanks to its simplicity and effectiveness, CS has attracted the attention of scholars from the date of its birth and has been applied to many fields successfully. [6-10] For some complex problems, CS cannot meet the accuracy requirements of the optimal solution, and its performance needs to be further improved.

In this paper, the improved CS with deep search (DSCS) is proposed. The opposition-based learning strategy and local enhanced search are introduced in DSCS. DSCS has a better global search ability, search precision and convergence speed.

II. CUCKOO SEARCH ALGORITHM

There are three rules in CS [1-3]:

Rule 1. A cuckoo only lays one eggs each times, and stores it in a random nest.

Rule 2. The nest with high quality egg will be retained to the next generation.

Rule 3. The number of available nests is fixed, and the probability to find the exotic egg is $p_a \in (0,1)$.

Based on these three rules, CS includes four steps, initialization, search, selection and judgment of the. The steps are as follows:

(1) Initialization. Randomly initialize the positions of N nests $X_0 = \left(x_1^0, x_2^0, \cdots, x_N^0\right)$, calculate their object values F_0, and choose the best position.

(2) Search. Generate new positions in Lévy flight method $X_t = \left(x_1^t, x_2^t, \cdots, x_N^t\right)$, calculate their object values F_t, and choose the best position again.

Lévy flight generally use random walk strategy as in (1) [3].

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

where, $X_{t,i}$ is the ith solution in generation t; X_{best} is the current best solution; α_0 is the step constant, generally $\alpha_0 = 0.01$; u and v are standard normal random variables; β is the control factor of Lévy flight, generally $\beta = 1.5$; ϕ uses (2) to express.

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

(3) Selection. Discard bad positions in discovery probability p_a, and use (3) to generate the same number of new positions. Calculate their object values F_t, and choose the best position again.

$$X_{t+1,i} = X_{t,i} + r\left(X_{t,j} - X_{t,k}\right). \tag{3}$$

where, r is the scaling factor, which is a standard normal random variable; $X_{t,j}$ and $X_{t,k}$ are two random solutions in generation t.

(4) Ending judgment. If the iterative termination conditions are satisfied, then stop the algorithm, or return to (2).

From the above steps, we can see that CS not only uses Lévy flight method, but also introduces the elitist reservation strategy. Local search and global search are well combined in CS. The selection step increases the position diversity, so that the algorithm can effectively avoid the local optimum solution. [4]

III. Opposition-based Learning Strategy

Opposition-based learning (OBL) [11] had been proposed in 2005, but it was used in differential evolution algorithm by Rahnamayan et al [12] until 2008. OBL is to calculate the opposition-based solution of a feasible solution, then evaluate the original solution and the opposition-based solution at the same time, and select the better one. [13]

Definition 1. Let $x \in [a,b]$ be a real number, then its opposition-based number is as in (4).

$$x' = a + b - x . \tag{4}$$

As can be seen from definition 1, the distance between x and a is $|x-a| = x-a$, and the distance between x' and b is $|b-x'| = |b-(a+b-x)| = |x-a| = x-a$. The two distances are equal. The positions of x and x' are as in Fig. 1.

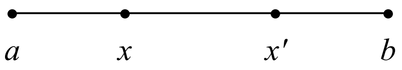


Figure 1. A real number and its opposite-based number

Definition 2. Let $P = (x_1, x_2, \cdots, x_D)$ be a point in D-dimensional space, $x_i \in [x_{i\min}, x_{i\max}]$, then its opposition-based point is as in (5).

$$\begin{aligned} &P' = (x_1', x_2', \cdots, x_D') \\ &x_i' = x_{i\min} + x_{i\max} - x_i \end{aligned} . \tag{5}$$

As can be seen from Fig. 1, since the feasible solution and the opposition-based solution are located at the two sides of the search space, when the OBL strategy is introduced, the search area can be enlarged and the global search ability is enhanced.

IV. DSCS Model

In DSCS, two improvements are introduced. The first one is to generate an ob-swarm with OBL strategy, and use it in position replacement operation. The second one is to deeply search for a potential better solution in the evolutionary direction around the current global optimum solution in the end of each generation.

A. *Generate Opposition-based Swarm*

In [14], it is proved that the opposition-based solution of the current feasible solution has a probability 50% to be more close to the optimal solution. Although the feasible solution and opposition-based solution have the equal probabilities to be reserved to the next generation under the same conditions, it can improve the probability of obtaining the optimal solution if feasible solution and opposition-based solution are both used and the better one is chosen.

In the selection stage of CS, once a position is discarded, three random positions from the origin swarm will be chosen to generate a new position with (3) and the discarded position will be replaced by the new position. The quality of the new location is closely related to the quality of the original individuals. In the selection stage of DSCS, an ob-swarm is obtained from the origin swarm with (5). The detailed steps are as follows:

Let population size be *N*, dimension be *D*, position matrix of origin swarm be $NEST_{N\times D}$, the upper and lower bounds of each dimension be $UP_{N\times D}$ and $LOW_{N\times D}$, then the opposition-based matrix $NEST'_{N\times D}$ is as in Eq. (6).

$$NEST'_{n\times d} = LOW_{n\times d} + UP_{n\times d} - NEST_{n\times d} . \tag{6}$$

In DSCS, the three random positions used in (3) will be chosen from $NEST'_{n\times d}$. This change effectively enlarges the search area and accelerates the search speed of the global optimal solution.

B. *Local Deep Search*

Let the dimension of firefly individuals be 2, the coordinate of the former global best position *A* be $X_A = \lfloor x_1^t, x_2^t \rfloor$, the coordinate of the current global best position *B* be $X_B = \lfloor x_1^{t+1}, x_2^{t+1} \rfloor$, *B* be better than *A*, the evolution direction be $\overrightarrow{AB}$, then position relation between *A* and *B* is as in Fig. 2.

Because the search step is not necessarily the best, there will be better position than *B*. If the search step is too small, the better one will exist in direction $\overrightarrow{BC}$. Instead, the better one will exist in direction $\overrightarrow{BA}$, if the search step is too large.

In DSCS, in order to make up for the difficulty of obtaining the optimal step size, a local deep search method is proposed. It is to search for better solutions both in the directions $\overrightarrow{BA}$ and $\overrightarrow{BC}$ with a proper step size Δ from *B*. The detailed flows are as in Fig. 3. There are four steps.

Step 1. Calculate the coordinate difference of *A* and *B* with (7).

$$X_{BA} = X_B - X_A = \lfloor x_1^{t+1} - x_1^t, x_2^{t+1} - x_2^t \rfloor . \tag{7}$$

Step 2. Calculate the local search step size Δ with (8).

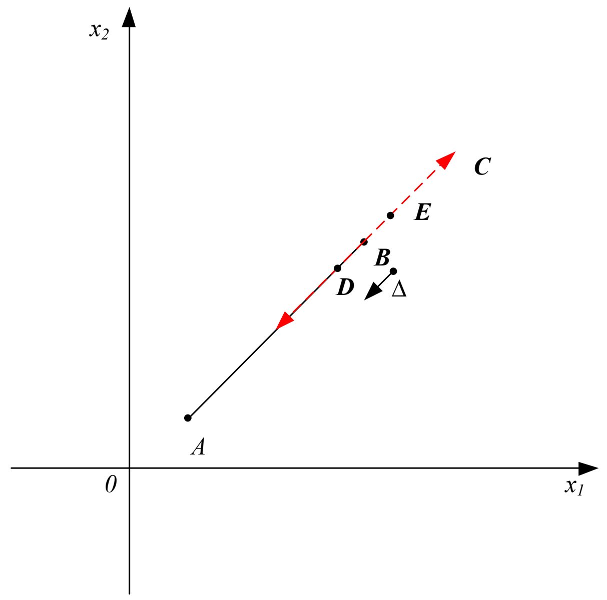
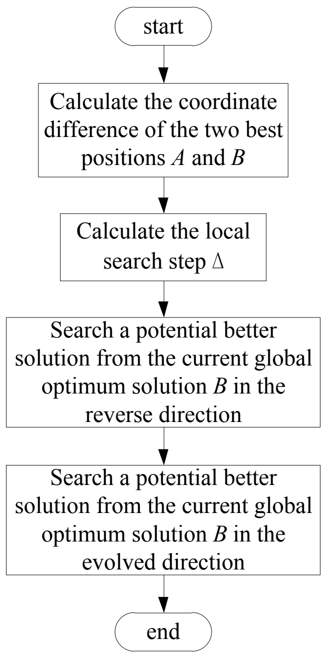


Figure 2. Local deep search schematic diagram



start
Calculate the coordinate difference of the two best positions *A* and *B*
Calculate the local search step Δ
Search a potential better solution from the current global optimum solution *B* in the reverse direction
Search a potential better solution from the current global optimum solution *B* in the evolved direction
end

Figure 3. Flow chart of local deep search

where, η is the scale factor which needs to be optimized in practice.

Step 3. Search for potential better position from position B. The detailed search flow is as in Fig. 4. There are seven steps as follows:

(1) Set search index i as 1 and reset the sign for better solution.

(2) Set the coordinate XPi of position i with (9).

$$\Delta = \eta X_{BA} . \tag{8}$$

$$X_{Pi} = X_B - i * \Delta . \tag{9}$$

(3) Analyse the performances of position i and the current best position. If position i is better then go to next step, or jump to step (5).

(4) Update the best position message and set the sign for better solution.

(5) If the sign for better solution is true, jump to step (7), or go to next step.

(6) If the value of i reaches the one-way position searching maximum numbe imax, then go to next step, or return to step (2). imax needs to be optimized in practice.

(7) Exit.

Step 4. Search for potential better position from position B in opposition direction. The detailed search flow is similar as in step 3. The only difference is that the coordinate X_{Pi} of position i is set with (10).

$$X_{Pi} = X_B + i * \Delta . \tag{10}$$

The 2-dimension problem solution can be applied to other dimension problem. The only difference is to change the coordinate Equation.

V. DSCS Simulation

This section compares DSCS with CS in order to verify the validity. 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 of DSCS and CS are as follows:

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

Variable parts: discovery probability $pa = 0.05$, Lévy flight control factor $\beta = 1.3$, step control factor $\alpha_0 = 0.1$.

DSCS own parts: scaling factor $\eta = 150$, one-way position searching maximum number $i_{\max} = 15$.

Each simulation runs 30 times separately. The simulation result is shown in Table Ⅱ, and the summary is shown in Table Ⅲ. As can be seen from the two tables, the performance of DSCS is better than CS. In the simulation of test function f_4, the success times increase to 15 from 0, raising the success rate by 50%. While in the simulations of other five test functions, both algorithms obtain 100% success rate, but the convergence speed and convergence precision of DSCS are better.

In order to observe the algorithm performances better, the finish condition no longer includes convergence precision.

The only finish condition is maximum iteration number which is $t_{\max} = 2000$. Other parameters remain unchanged. Take single-modal function f_1 and multi-modal function f_4 as examples, the average optimal object value curves of the two algorithms are shown in Fig. 5 and Fig. 6. As can be seen in Fig. 5 and Fig. 6, the convergence speed and convergence precision of DSCS are higher than that of CS, regardless of whether the test function is a simple single-modal function or a complex multi-modal function.

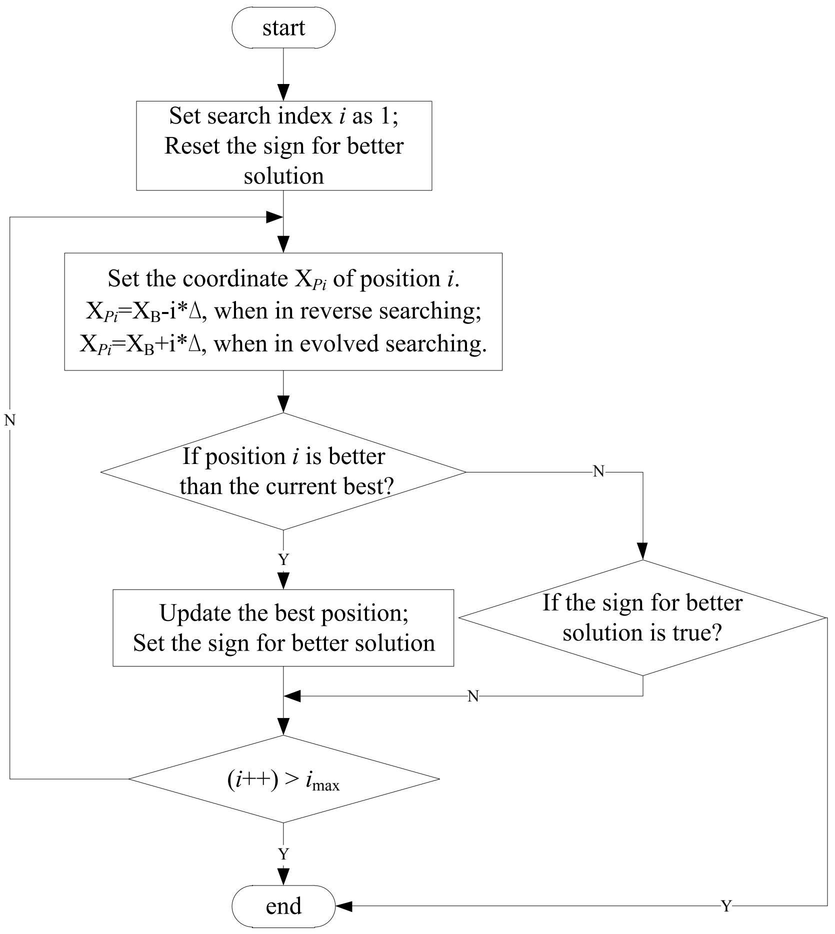

start
Set search index i as 1; Reset the sign for better solution
Set the coordinate X_{Pi} of position i. $X_{Pi}=X_B-i*\Delta$, when in reverse searching; $X_{Pi}=X_B+i*\Delta$, when in evolved searching.
If position i is better than the current best?
Update the best position; Set the sign for better solution
If the sign for better solution is true?
$(i{+}{+}) > i_{\max}$
end
N
Y
N
N
Y
Y

Figure 4. Flow chart of searching for better position

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	$f_2(x) = \sum_{j=1}^{n} j x_j^4$	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 DSCS AND CS

test function	algorithm	success times (average success iteration)	average optimal value	standard deviation of optimal value
f_1	CS	30（412）	8.9362e-06	1.0158e-06
	DSCS	30（276）	3.9564e-45	5.7821e-45
f_2	CS	30（347）	8.3972e-06	1.4807e-06
	DSCS	30（211）	8.4396e-74	2.6206e-73
f_3	CS	30（1044）	9.4701e-06	5.0536e-07
	DSCS	30（690）	1.1842e-20	2.5623e-20
f_4	CS	0	16.3860	5.2497
	DSCS	15（1199）	1.8530	4.5640
f_5	CS	30（576）	9.1673e-06	6.1861e-07
	DSCS	30（298）	0	0
f_6	CS	30（718）	9.2997e-06	5.7136e-07
	DSCS	30（502）	4.4409e-15	0

TABLE III. SUMMARY RESULT OF f_1 ~ f_6 SIMULATION

test index / algorithm type	success times lead times	average success iteration lead times	average optimal value lead times	standard deviation of optimal value lead times
CS	0	0	0	0
DSCS	1	6	6	6

The simulation results show that the convergence speed, convergence precision and global search ability of the algorithm are all improved at different levels after the introduction of the opposition-based learning strategy and the local enhanced search operation.

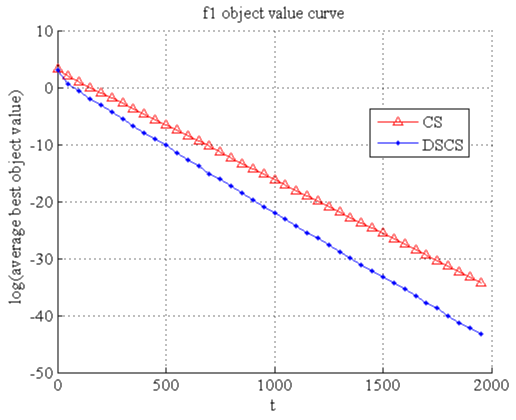


Figure 5. Average best object value curve of f_1

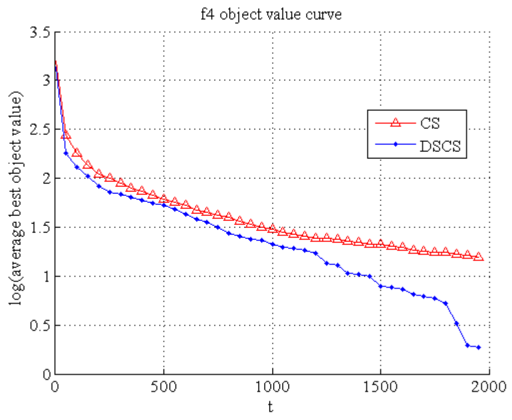


Figure 6. Average best object value curve of f_4

VI. CONCLUSIONS

In the selection stage of the algorithm DSCS, an opposition-based swarm is generated based on the original swarm, then three individuals are randomly selected to produce a new individual to replace the discarded individual from the original swarm. In the evolution of each generation, it is similar to search for two symmetric regions at the same time, which accelerates the convergence speed and improves the global search ability. Simulations of six standard test functions show that the performance of DSCS is significantly better than that of CS. Global searching ability, convergence speed and convergence precision of DSCS are much better. DSCS shows good robustness on optimization simulation both for simple single-modal function and complex multi-modal function.

REFERENCES

[1] X. S. Yang, Nature-inspired metaheuristic algorithm. Luniver Press, 2008.

[2] X. S. Yang, Cuckoo search and firefly algorithm theory and applications. Springer, 2014.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] H. R. Tizhoosh, “Opposition-based learning: A New Scheme for Machine Intelligence,” International Conference on Computational Intelligence for Modelling, Control and Automation, 2005, pp. 695-701.

[12] S. Rahnamayan, H. R. Tizhoosh, M. A. Salama, “Opposition-based differential evolution,” Evolutionary Computation, IEEE Transactions on, 2008, vol. 12, no.1, pp. 64-79.

[13] X. Y. Zhou, Z. J. Wu, H. Wang, K. S. Li, H. Y. Zhang H, “Elite Opposition-based particle swarm optimization,” Acta Electronica Sinica, 2013, vol. 41, no. 8 pp. 1647-1652.

[14] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, “Enhancing particle swarm optimization using generalized opposition-based learning,” Information Sciences, 2011, vol. 181, no. 20, pp. 4699-4714.