Applied Mathematics and Computation 266 (2015) 1083–1092

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Cuckoo search algorithm based on frog leaping local search and chaos theory

Xueying Liu [a,b,*], Meiling Fu [b]

[a] College of Arts and Sciences, Shanghai Maritime University, Shanghai 201306, China
[b] College of Science, Inner Mongolia University of Technology, Hohhot 010051, China

ARTICLE INFO

Keywords:
Cuckoo search
Chaos theory
Frog leaping algorithm
Inertia weight

ABSTRACT

Cuckoo algorithm is a novel optimization algorithm in the field of heuristic intelligence algorithms. Given the strong random leaping in solution space search, careful local searches are susceptible to falling into the local optimum. Thus, the latter phase of the optimization slows down and the accuracy diminishes. To improve the performance of the algorithm, this paper proposes an improved cuckoo search that utilizes chaos theory to enhance the variety of the initial population. Then, this study introduces inertia weight into the Lévy flight random search to improve global searching capability. Finally, it applies the local search mechanism of the frog leaping algorithm to enhance local search and further improve the search speed and convergence precision of the algorithm. Typical test functions are employed to verify the performance of the improved algorithm. Comparison results with other algorithms indicate that the improved algorithm displays strong optimizing accuracy and high speed. Furthermore, this algorithm is confirmed to be convergent.

1. Introduction

Swarm intelligence algorithms are a new type of evolutionary algorithm. These algorithms include genetic algorithms [1], difference evolution (DE) [2], particle swarm optimization [3], shuffled frog leaping algorithms (SFLA) [4], and monkey algorithms [5], which are inspired by natural laws, as well as the intelligent behavior of biological populations. These algorithms have demonstrated their unique capacities, as well as their applicability in science and engineering technology. Each intelligence algorithm corresponds to one practical heuristic source. For example, the biogeography algorithm [6,7] applies a global optimization method according to the distribution characteristics of biology in geography. Glowworm swarm optimization [8] is a random optimization algorithm that simulates natural bioluminescence behavior. Wolf pack algorithm [9] is a leadership strategy-based algorithm that follows the predation tendencies of wolf packs. Therefore, the remarkable concept and wide applicability of intelligent optimization algorithms are consistently explored and developed by numerous researchers.

Cuckoo search (CS) was proposed by Cambridge scholars Yang Xin She and Deb Suash. This global search algorithm [10] is inspired by the behaviors of cuckoos in locating nests and laying eggs [11] and by the Lévy flight of insects [12]. CS is simple and operates under few controlled parameters, optimal search paths, strong optimizing capability, and ease of use. Considering these advantages, CS has been widely applied in practical engineering optimization problems. Nonetheless, fundamental CS exhibits slow convergence rate and low convergence precision. Thus, many scholars have developed various improvement methods. For example, Literature [13] enhanced CS by solving a function optimization problem. Literature [14] initiated a CS based on Gaussian

* Corresponding author. Tel.: +86 13664882077.
E-mail address: xyliu@aliyun.com (X. Liu).

http://dx.doi.org/10.1016/j.amc.2015.06.041
0096-3003/

distribution. Literature [15] presented a CS based on decision-maker and disturbance factors. Literature [16] improved CS for global optimization. Literature [17] enhanced CS by solving an unconstrained optimization problem. Literature [18] presented a new and complicated CS. Literature [19] mixed a DE algorithm with a CS algorithm. Literature [20] proposed a self-adapting, step-length CS. All of these studies served to enhance the search performance of the algorithm.

To improve the performance of the algorithm further, the current study improves CS and utilizes the initial population of chaos sequence to maintain the variety of the population. Then, the study introduces inertia weight into a Lévy flight random search to enhance global searching capability. Finally, this research uses the local search mechanism of the frog leaping algorithm to conduct a local search for local optimal solutions and to accelerate the convergence of the algorithm.

2. Cuckoo search algorithm

In nature, cuckoos randomly seek nests in which to lay their eggs. To simulate this process, the following three ideal hypotheses are developed:

(1) Each cuckoo lays an egg once and allows it to incubate in a randomly chosen nest.
(2) A group of bird nests is selected at random, and the best nests are maintained until the next generation.
(3) The number of available bird nests n is fixed. The possibility that the owner of a nest locates the nest of a foreign bird is denoted by $p_a \in [0, 1]$.

Given the premise of the hypotheses above, the nest-seeking behavior of cuckoos follows the Lévy flight model. The path and location update formula for this behavior is written as follows:

$$x_i^{t+1} = x_i^t + \alpha \oplus L(\lambda), \quad i = 1, 2, 3 \ldots n \tag{1}$$

where x_i^t and x_i^{t+1} represent the locations of nest i at generations t and $t+1$, respectively; $\oplus$ represents point-to-point multiplication; and $L(\lambda)$ represents a random Lévy search path. The length and direction of this path are uncertain. To apply this formula to CS successfully, Literature [10] introduced the step length-controlled variable α, whose value is a constant over zero. This value varies in different cases, but $\alpha = 0.01$ in general.

$L(\lambda)$ is a Lévy distribution function that complicates integration. Therefore, Yang Xin She converted this function into a probability density function [21] in terms of power through simplification and Fourier transformation.

$$\text{Lévy} \sim u = t^{-\lambda} \ (1 < \lambda < 3), \tag{2}$$

where λ represents the power coefficient. To describe this part in simple and programmable mathematical language, Yang Xin She and Deb Suash adopted the formula of Lévy flight leaping path [22], which was proposed by Mantegna in 1992 to realize CS.

$$s = \frac{\mu}{|v|^{\frac{1}{\beta}}}. \tag{3}$$

Literature [21] proved that the Mantegna algorithm can calculate equivalence. In Formula (3), s represents the Lévy leaping flight path $L(\lambda)$. Furthermore, the relationship between β and λ in formula (2) can be expressed as $\lambda = \beta + 1, 0 < \beta < 2$ ($\beta = 1.5$ [23] in CS). $\mu \cdot v$ represents the random number of normal distribution that follows the normal distribution in formula (4), and the standard deviation of the corresponding normal distribution in this formula is represented by σ_μ, σ_v. The values are shown in formula (5).

$$\mu \sim N\left(0, \sigma_\mu^2\right), \quad v \sim N\left(0, \sigma_v^2\right). \tag{4}$$

$$\sigma_\mu = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \tag{5}$$

Suppose $S = \alpha \oplus L(\lambda) = \alpha \oplus s$, where S represents the path of cuckoos from the location of a previous nest x_i^t to a new location x_i^{t+1} as determined randomly in the solution space based on formula (1). s requires the derivation of two random numbers of normal distributions μ and v from formulas (4) and (5). μ and v are uncertain values; therefore, the length and direction of the path that the cuckoo searches at random according to Lévy flight are highly randomized. As a result, leaping from one region to another is easy. These features benefit the global search capability of the algorithm in the early phase of optimization, thereby enhancing the global optimizing capability of CS.

$r \in [0, 1]$ and p_a are compared through location updates. If $r > p_a$, then the located nest is changed at random; if not, the nest remains unchanged. Finally, the best nest location (x_i^{t+1})with a superior test value is maintained. At this point, the best (x_i^{t+1})is denoted by x_i^{t+1}.

Basic steps of CS:

Step 1: Initialization parameters are set. n best locations are randomly generated, and the corresponding adaptive values of nests are calculated. Then, the current optimal location of the bird and of the global optimal solution are determined and maintained until the next generation.

Step 2: Formulas (1), (3), (4), and (5) are used to update nest locations. Thus, a group of new nest locations are obtained. The post-update nest locations are compared with the original location, and the superior location is integrated into the next generation.

Step 3: A random number of uniform distribution $r \in [0, 1]$ is compared with p_a. If $r > p_a$, then the located nest is changed randomly. Otherwise, the nest remains unchanged.

Step 4: The termination condition of the algorithm is determined. If the condition is satisfied, then the optimal solution becomes an output. Otherwise, Step 2 is repeated.

3. ACS based on frog leaping local search and chaos theory

3.1. Population initialization

Literature [24] reports that an initial population with sufficient variety significantly enhances the global optimizing efficiency of the iterative algorithm for population random search. Basic CS generates the initial population in the search space at random. This process may result in poor population diversity and further influence the efficiency of the algorithm.

Chaos refers to a random state in the deterministic system. This state is the evolution of nonlinear systems through deterministic rules and is a long-term behavior without a fixed period. Chaos is sensitive to initial conditions. Chaotic motion can traverse all of the states according to itself without repeating these states within a certain range. As such, this motion is ergodic. The use of chaos search is more advantageous than that of disorderly random searches. Logistic and Tent maps are frequently used in chaos search. The equation of the Logistic map is as follows [25]:

$$x(t+1) = \varepsilon x(t)(1 - x(t)), \tag{6}$$

where t is the iteration times; $x(t) \in [0, 1]$; and ε is the control parameter. If $\varepsilon = 4$, then the system is in a state of chaos. The Logistic map belongs to the Li-Yorke chaos [26], and the path of $x(t)$ can be ergodic throughout an entire range only when the initial value of logistic map $x^{(0)} \in [0.00, 0.25, 0.25, 0.75]$. The Tent map is related to Devaney chaos [27] can produce a sequence of uniform distributions. This map refers to a type of chaos model that iterates quickly, according to the following formula [28]:

$$x_{n+1} = \begin{cases} 2x_n & 0 \le x_n \le 0.5 \\ 2(1 - x_n) & 0.5 \le x_n \le 1. \end{cases} \tag{7}$$

In this work, the Logistic and Tent maps are used to initialize CS algorithm solutions. Then, a linear transformation formula $x^{(k)} = a + (b - a)x^{(t)}$ is used to map the range of the chaos variable to the domain of the definition of an optimization problem.

3.2. Frog leaping local search

SFLA was officially introduced by Eusuff and Lansey in 2003 as a meta-heuristic collaboration algorithm used with a population. The natural individual foraging behaviors of frogs are simulated in the implementation of SFLA.

The concept of SFLA refers to the division of a group of frogs that live in a wetland into several populations. Each population collaborates to seek for food based on the information shared by each frog. Once each population has independently sought food for a period of time, all populations gather and join as one group, which facilitates information sharing. Afterwards, the group is reclassified in certain ways, and the population forages independently again. The frog population, as well as the individuals, must cooperate mutually and share information to locate food sources immediately by reiterating these actions in a circular manner. Therefore, the frog population with n frogs is divided into m tribal groups. Each tribal group includes p frogs, and different tribal groups represent frog sets with various ideas and information. A local depth search is conducted and internal ideas are exchanged in the solution space among the frogs in the tribal group according to the element strategies of evolution.

$$Step = rand^{*}(P_{bi} - P_{wi}), \tag{8}$$

$$p1_{wi} = p_{wi} + Step - S_{\max} < Step < S_{\max}, \tag{9}$$

where *rand* represents any randomly generated number in $(0, 1)$; $S_{\max}$ denotes the maximum step length of frog leaping; P_{wi} represents the worst frog in tribal group i; and P_{bi} corresponds to the best frog in tribal group i. If the adaptive value of the updated frog is greater than that of the worst frog, the updated frog replaces the former worst frog. Otherwise, a random frog in this tribal group replaces the worst frog. The execution of the update process is repeated until it reaches the preset local update time *Ne*. Once the depth searches across all tribal groups are completed, the frogs from the tribal groups are remixed. The frog group is later divided into m tribal groups. Subsequently, the local search proceeds until the termination criterion of the algorithm is satisfied.

3.3. Introduction of inertia weight

In basic CS, the nest seeking behavior and paths of cuckoos are random. To improve the performance of the algorithm, this study introduces inertia weight into Formula (1). The path and location update formula of the nest seeking behavior of cuckoos is expressed as:

$$x_i^{t+1} = w x_i^{(t)} + \alpha \oplus L(\lambda), \quad i = 1, 2, 3 \ldots n. \tag{10}$$

The introduction of inertia weight can expand the CS search space and facilitate the location of new areas. In the global optimization algorithm, the early phase is generally expected to display a strong global searching capability, whereas the later phase exhibits a high development capability to accelerate convergence.

The experimental result indicates that increased inertia weight w is conducive to jumping out of the local optimum and to the ongoing global optimization. Low inertia weight w benefits local optimization and accelerates algorithm convergence. To balance the global and local search capabilities of the algorithm, the value of inertia weight w decreases as iteration times increase. However, the CS in the actual search process is nonlinear, and the linear decreasing strategy of inertia weight cannot reflect the actual optimization of this process. Thus, this study introduces a linear decreasing strategy of inertia weight [29].

$$w = (2/t)^{0.3}, \tag{11}$$

where t represents the current iteration times.

3.4. Iteration steps of ACS based on frog leaping local search and chaos theory

Step 1: Initialization of relevant parameters: population quantity n, dimensions of search space d, maximum iteration times T, detection probability p_a, range of search space $[Lb, Ub]$, update times of frog leaping search in group Ne, group number m, number of nests in each group p, and iteration counter $t = 1$.

Step 2: Chaos initialization of population individuals: first, n is randomly generated as the vector quantity of the first cuckoo individual, that is, $X = (x_1, x_2, x_3 \ldots x_n)^T$. $x_i \in [0, 1]$ $x_i \in [0, 1]$. Each dimension of$X$is updated with either Formula (6) or (7), and the chaos variable is generated. This variable is further mapped in the value space of the solution using carrier formula $X^{(n)} = Lb + (Ub - Lb)x^{(n)}$.

Step 3: The corresponding adaptive value of each nest location is calculated, and all nest individuals are arranged in descending order based on adaptive value. The order aims to determine the optimal nest location, as well as its adaptive value.

Step 4: n nests are divided into m groups, and each group contains p nests. The best solution P_{bi} and the worst solution P_{wi} in the tribal group are determined. Updates are made according to Formulas (8) and (9). If the location of the worst nest is improved, that is, the adaptive value of $p1_{wi}$ is greater than that of P_{wi}, then the location of the new $p1_{wi}$ replaces the location of the worst nest P_{wi}; that is, $P_{wi} = p1_{wi}$. Otherwise, the random location of a nest in this tribal group replaces the location of the worst nest. Nest individuals are remixed in descending order to construct a new nest population. The corresponding adaptive value of the location of the new nest is calculated, and the optimal location is maintained until the set local search times are reached.

Step 5: Nest locations are updated according to Formulas (3)–(5), (10), and (11).

Step 6: An evenly distributed number is randomly generated as $r \in [0, 1]$. If $r > p_a$, the located nest is moved at random. Otherwise, it remains unchanged. A new group of nest locations can be obtained in this manner.

Step 7: The adaptive value of each nest location is evaluated, and the historical best positions of the nests are then updated through comparison.

Step 8: The termination condition of the algorithm is evaluated. If the condition is satisfied, then the optimal solution is outputted. Otherwise, Step 3 is repeated.

Note: The chaos initialization realized with Formulas (6) and (7) are indicated by ACS1 and ACS2, respectively.

4. Numerical simulation experiment

4.1. Test function

This study selects 14 classical test functions to verify algorithm performance. These functions are classified into unimodal and multimodal functions according to performance as follows.

The sphere function is a separable unimodal function typically used to test the accuracy of an algorithm. The Rastrigin function is a typical inseparable multimodal function that is defined by large numbers of locally optimal solutions. A global optimum is difficult to obtain with a general algorithm. Furthermore, the Rosenbrock function is an inseparable unimodal function that is employed to test the local search capability of the algorithm. The Shaffer function is a complex multimodal function consisting of unlimited local minimums. Given these local maximums near the second-best solutions, the globally optimal solution for this function is difficult to determine.

(1) Sphere function. The global maximum is 0 at (0,0).

$$f_1(x) = \sum_{i=1}^{n} x_i^2 n = 30 - 100 \le x_i \le 100 \quad f_{\min}(0, 0 \cdots 0) = 0$$

(2) Rastrigin function. The global maximum is 0 at (0,0).

$$f_2(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right] n = 30 - 100 \le x_i \le 100 \quad f_{\min}(0, 0 \cdots 0) = 0$$

Table 1
Test parameters.

n	p_a	T	α	m	Ne	$Smax$	p
100	0.25	100	0.01	10	10	50	10

(3) Rosenbrock function. The global maximum is 0 at (1,1)

$$f_3(x)=\sum_{i=1}^{n-1}\left[100\left(x_i^2-x_{i+1}\right)^2+(x_i-1)^2\right] n=30-100\le x_i\le 100 \quad f_{\min}(1,1\cdots 1)=0$$

(4) Griewank function. The global maximum is 0 at (0,0).

$$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{x_i}}+1 n=30-600\le x_i\le 600 \quad f_{\min}(0,0\cdots 0)=0$$

(5) Ackley function. The global maximum is 0 at (0, 0).

$$f_5(x)=-20\exp\left[-0.2\times\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right]-\exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right]+20+e n=30-100\le x_i\le 100 \quad f_{\min}(0,0\cdots 0)=0$$

(6) Camel function. The global minimum is −1.0316285 at (0.08983, −0.7126)

$$f_6(x)=\left(4-2.1x_1^2+\frac{x_1^4}{3}\right)x_1^2+x_1x_2+\left(-4+4x_2^2\right)x_2^2 n=2-100\le x_i\le 100 \quad f(0.08983,-0.7126)=-1.0316285$$

(7) Schwefrl's problem (2.22). The global maximum is 0 at (0, 0).

$$f_7(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i| n=30-100\le x_i\le 100 \quad f_{\min}(0,0\cdots 0)=0$$

(8) Quadric function. The global maximum is 0 at (0, 0).

$$f_8(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2 n=30-100\le x_i\le 100 \quad f_{\min}(0,0\cdots 0)=0$$

(9) Easom function. The global maximum is −1 at (0, 0).

$$f_9(x,y)=-\cos(x)\cos(y)\exp[-(x-\pi)^2-(y-\pi)^2]\ n=2-100\le x_i\le 100 \quad f_{\min}(0,0)=-1$$

(10) Shaffer function. The global maximum is 0 at (0, 0).

$$f_{10}(x)=\left(x_1^2+x_2^2\right)^{\frac{1}{4}}\left[\sin^2\left(50\left(x_1^2+x_2^2\right)^{\frac{1}{10}}+1.0\right]\right. n=2-100\le x_i\le 100 \quad f_{\min}(0,0)=0$$

(11) Schaffer function. The global maximum is 0 at (0, 0).

$$f_{11}(x)=\frac{\sin^2(\sqrt{x_1^2+x_2^2}-0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}+0.5 n=2-100\le x_i\le 100 \quad f_{\min}(0,0)=0$$

(12) Schwefrl's problem 2.21. The global maximum is 0 at (0, 0).

$$f_{12}(x)=\max\{|x_i|\} n=30-100\le x_i\le 100 \quad f_{\min}(0,0\cdots 0)=0$$

(13) Bohachevsky1. The global maximum is 0 at (0, 0).

$$f_{13}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7 n=2-100\le x_i\le 100 \quad f_{\min}(0,0)=0$$

(14) Bohachevsky2. The global maximum is 0 at (0, 0).

$$f_{14}(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)\cos(4\pi x_2)+0.3 n=2-100\le x_i\le 100 \quad f_{\min}(0,0)=0$$

4.2. Results of numerical experiments

4.2.1. Parameter setting

This experiment is compiled and ran in Matlab 2013a on a computer with an i5 processor as a CPU, a 4G computer memory, and a Windows 7 operating system. The parameters set in the experiments are shown in Table 1.

4.2.2. Results of numerical experiments

To test the performance of the ACS algorithm, each function is independently run 30 times during the fixed iteration times. The best value (Best), worst value (Worst), mean (Mean), variance (Variance), and standard deviation (SD) are used to evaluate the convergence speed of the algorithm, as well as the solution accuracy and stability. This study further compares ACS with the evolutionary algorithms used in other studies. The test results are depicted in Table 2.

4.3. Analysis of experiment results

Table 2 suggests that the ACS proposed in this paper is superior to other improved algorithms with respect to most test functions. For functions f_2, f_4, f_5, f_6, f_9, f_{13}, and f_{14}, ACS displays an excellent global optimizing capability. For functions f_1, f_7, f_8, f_{10}, and f_{12}, ACS exhibits accurate optimization capability. For functions f_3 and f_{11}, global optimizing capability is poor. For functions $f_1 \cdot f_7 \cdot f_{10} \cdot f_{11} \cdot f_{12}$, ACS1 is superior to ACS2. For functions $f_2 \cdot f_4 \cdot f_6 \cdot f_9 \cdot f_{13} \cdot f_{14}$, the global optimal solution can only be determined with both ACS1 and ACS2. For functions $f_3 \cdot f_8$, ACS2 is superior to ACS1. Upon comparing the operation times, we discovered that ACS2 is superior to ACS1.

In sum, this study utilizes these classical functions to test ACS and compares its resultant indices (including Best, Worst, Mean, Variance, and SD) with those of other algorithms. The results of various experiments indicate that ACS is significantly superior to other algorithms in terms of optimizing accuracy and global optimizing capabilities.

5. Convergence analysis of the ACS algorithm

5.1. Convergence criterion

Essentially, ACS belongs to random search algorithm. Thus, it can assess whether or not an algorithm is convergent according to the convergence criterion of the random algorithm. Literature [31] presented this criterion for general random optimization algorithms.

The result of the next iteration is $x_{k+1} = D(x_k, \zeta)$ for optimization problem $\langle A, f\rangle$, random optimization algorithm D, results of iteration k, and x_k. A represents a feasible solution space, f denotes fitness function, and ζ corresponds to the ever-searched solutions of the algorithm in iteration D.

In the Lebesgue measure space, the lower bound of the search is defined as:

$$\varphi = \inf\{t | \nu(x \in A | f(x) < t) > 0\},$$

where $\nu(X)$ represents the Lebesgue measurement in set X. Thus, the optimal solution region can be defined as,

$$R_{\varepsilon,M} = \begin{cases} \{x \in A | f(x) < \varphi + \varepsilon\} & \varphi \text{finite} \\ \{x \in A | f(x) < -C\} & \varphi = -\infty, \end{cases}$$

where $\varepsilon > 0$ and C represents a sufficiently large positive number. If the algorithm determines one point in $R_{\varepsilon,M}$, then this algorithm identifies an acceptable or an approximate global optimum.

Condition 1. If $f(D(x, \zeta)) \le f(x)$, and $\xi \in A$, then $f(D(x, \zeta)) \le f(\zeta)$.

Condition 2. For any $B \in A, s.t. \nu(B) > 0, \prod_{k=0}^{\infty}(1 - u_k(B)) = 0$, where $u_k(B)$ represents the probability measure of the iteration of algorithm $D(k)$ on set B.

Lemma [31] (the global convergence of the algorithm)**.** *Suppose f is measurable, the measurable space A is the measurable subset on R^n. If algorithm D satisfies Conditions 1 and 2 and $\{x_k\}_{k=0}^{\infty}$ is the sequence from algorithm D, then* $\lim p(x_k \in R_{\varepsilon,M}) = 1$. *AlgorithmDdisplays global convergence, where $p(x_k \in R_{\varepsilon,M})$.*

5.2. Mathematical definition of the fundamental concepts of ACS

Definition 1 (nest location state and the state space of the nest location [32])**.** Nest location x and historical optimum location pb constitute the nest state, which is denoted as $y = (x, pb)$. $x, pb \in A$ and $f(pb) \le f(x)$. The set of all possible nest location states comprises the state space of the nest location, which is represented by $Y = \{y = (x, pb) | x, pb \in A,\ f(pb) \le f(x)\ \}$.

Definition 2 (population state of the nest location and the population state space of the nest location [32])**.** The set of n nest states is known as the population state of the nest location and is denoted as $q = (y_1, y_2, y_3 \ldots y_n)$. The population state of the nest location constitutes the population state space of the nest location, which is represented by $Q = \{q = (y_1, y_2, y_3 \ldots y_n), y_i \in Y, 1 \le i \le n\}$.

The population space of the nest location contains the historical best position pb of the population and the historical best positions of all nests $pb_i (1 \le i \le n)$, where the best in pb_i is the historical best position of the population $gb = pb^*.f(pb^*) = \min(f(pb_i)),\ \ 1 \le i \le n$.

Table 2
Comparison of fourteen functions of ACS and other algorithms.

Function	Algorithm	Best	Worst	Mean	Variance	SD
f_1	OBDE [2,19]	3.05991167e-86	4.278226213e-05	1.42671462e-06	–	7.810816693e-06
	CLPSO [3,30]	–	–	1.89e-19	–	1.49e-19
	SFL-DE [4]	2.636e-13	4.908e-06	2.029e-07	8.956e-07	–
	SMMA [5]	5.73e-23	9.84e-23	8.18e-23	–	1.27e-23
	BBO [6]	1.34e-19	–	4.64e-19	–	8.78e-19
	BBO-EP [7]	–	–	2.35e-06	–	3.17e-07
	iCS [13]	2.6507e-11	2.3314e-10	1.0733e-10	5.4909e-11	–
	LDCS [15]	2.6051863e-22	2.6348625e-20	9.91e-21	9.32495e-41	–
	ICS [16]	2.9807e-22	4.1015e-20	9.5438e-21	–	1.1279e-20
	MCS [17]	4.68e-16	2.74e-14	7.00e-15	–	9.17e-15
	CS-DE [19]	5.262929834e-42	2.893364915e-23	9.64454971e-25	–	5.282537437e-24
	ASCS [20]	3.1311e-008	4.4939e-006	1.0063e-006	–	–
	ACS1	4.3346e-92	2.3585e-35	8.3683e-37	1.8536e-71	4.3054e-36
	ACS2	2.2955e-91	3.1053e-32	1.2253e-33	3.2354e-65	5.6881e-33
f_2	OBDE [2,19]	7.91516238	2.6503258e+1	1.4083459e+1	–	4.46978243
	CLPSO [3,30]	–	–	2.57e-11	–	6.64e-11
	SMMA [5]	0	0	0	–	0
	BBO [6]	8.57e+00	–	1.24e+01	–	1.24e+00
	BBO-EP [7]	–	–	3.97e-04	–	5.32e-04
	ASGSO [8]	0	6.25e-06	4.14e-07	–	–
	GCS [14]	106.2796	130.0725	114.5563	42.6203	–
	LDCS [15]	0	0	0	0	–
	ICS [16]	1.1939e+01	2.9747e+01	2.2296e+01	–	4.1242e+00
	MCS [17]	4.99e+00	8.49e+00	7.31e+00	–	1.15e+00
	CS-DE [19]	5.96975434	2.8853792e+1	1.5258617e+1	–	4.93736971
	ASCS [20]	9.0260	30.2961	17.9013	–	–
	ACS1	0	0	0	0	0
	ACS2	0	0	0	0	0
f_3	OBDE [2,19]	2.41574953e+1	5.433586738e+2	4.84442222e+2	–	9.45762295e+1
	CLPSO [3,30]	–	–	11	–	14.5
	SFL-DE [4]	4.667	2.920e+02	6.690e+01	5.512e+01	–
	SMMA [5]	27.74	28.27	28.08	–	0.118
	iCS [13]	1.9336e+01	2.6855e+01	2.4982e+01	1.6582e+00	–
	GCS [14]	4.8562	158.8805	25.4291	1.4115e+003	–
	ICS [16]	9.8653e+00	7.9836e+01	2.6678e+01	–	1.3697e+01
	MCS [17]	7.62e-01	5.38e+00	2.30e+00	–	1.64e+00
	CS-DE [19]	3.20788103e-9	1.197328520e+2	3.58514547e+1	–	3.47395156e+1
	ASCS [20]	2.4598e+003	2.0737e+004	7.5411e+003	–	–
	ACS1	0.5862	2.86786e+01	2.37383e+01	7.95310e+01	8.9180
	ACS2	0.2865	2.86989e+01	1.92532e+01	1.14257e+02	1.06891e+01
f_4	OBDE [2,19]	5.55111512e-16	8.77432556e-02	5.16170306e-03	–	1.637476538e-02
	CLPSO [3,30]	–	–	6.45e-13	–	2.07e-12
	SFL-DE[4]	2.418e-08	5.531e-02	9.848e-03	1.356e-02	–
	SMMA [5]	0	0	0	–	0
	BBO [6]	6.86e+00	–	7.82e+00	–	3.74e+00
	BBO-EP [7]	–	–	6.70e-08	–	9.55e-09
	ASGSO [8]	0	3.77e-14	6.340e-15	–	–
	iCS [13]	1.0421e-07	5.8156e-04	6.5011e-05	1.1131e-04	–
	LDCS [15]	0	0	0	0	–
	ICS [16]	1.1102e-16	4.9058e-08	3.1173e-09	–	1.1340e-08
	PCS [18]	0	0.1129	0.0256	0.0017	–
	CS-DE [19]	0	5.13255561e-2	3.92892620e-3	–	9.926222813e-3
	ASCS [20]	1.0968	1.4537	1.2444	–	–
	ACS1	0	0	0	0	0
	ACS2	0	0	0	0	0
f_5	OBDE [2,19]	1.509903313e-14	1.999999560e+1	5.799360575	–	8.5453220626
	CLPSO [3,30]	–	–	2.01e-11	–	9.22e-13
	SFL-DE [4]	3.693e-07	2.738	6.375e-01	7.816e-01	–
	SMMA [5]	4.88e-16	9.87e-16	7.56e-16	–	1.47e-16
	BBO [6]	9.98e–03	–	1.17e–03	–	1.07e–04
	BBO-EP [7]	–	–	9.40e-04	–	5.84e-05
	iCS [13]	1.4553e-04	6.3284e-04	3.1030e-04	1.3540e-04	–
	GCS [14]	1.5285e-005	1.1551	0.0580	0.0667	–
	ICS [16]	7.0179e-10	2.0393e+00	3.0880e-01	–	5.9632e-01
	MCS [17]	4.92e-06	3.14e-04	6.88e-05	–	1.05e-04
	PCS [18]	0.0052	13.0287	7.7304	13.5185	–
	CS-DE [19]	8.881784197e-16	1.999000255e+1	4.037832909	–	8.0175496521

(*continued on next page*)

Table 2 (*continued*)

Function	Algorithm	Best	Worst	Mean	Variance	SD
	ACS1	8.8818e-16	8.8818e-16	8.8818e-16	1.6094e-61	4.0117e-31
	ACS2	8.8818e-16	8.8818e-16	8.8818e-16	1.6094e-61	4.0117e-31
f_6	SMMA [5]	−1.031628434	−1.031623357	−1.31627551	–	1.24e-06
	BBO [6]	1.55e-07	–	1.93e-07	–	2.55e-07
	WPA [7]	−1.0316	−1.0316	−1.0316	–	0
	LDCS [15]	−1.0316284535	−1.0316284535	−1.0316284535	0	–
	ACS1	−1.0316284535	−1.0316284535	−1.0316284535	0	0
	ACS2	−1.0316284535	−1.0316284535	−1.0316284535	0	0
f_7	CLPSO [3,30]	–	–	1.01e-13	–	6.51e-14
	SMMA [5]	3.82e-11	4.83e-11	4.37e-11	–	2.83e-12
	BBO [6]	9.60e-16	–	2.67e-15	–	1.56e-15
	BBO-EP [7]	–	–	5.03e-07	–	3.17e-08
	ICS [16]	2.1208e-10	2.8235e-06	2.1058e-07	–	5.4655e-07
	ACS1	5.5148e-45	4.8549e-21	1.8636e-22	7.9454e-43	8.9137e-22
	ACS2	1.6000e-44	7.0992e-17	2.5205e-18	1.6758e-34	1.2945e-17
f_8	CLPSO [3,30]	–	–	395	–	142
	SMMA [5]	1.31e-21	4.50e-21	3.04e-21	–	9.24e-22
	BBO [6]	2.17e-03	–	5.26e-03	–	8.34e-03
	BBO-EP [7]	–	–	1.59e-03	–	2.90e-03
	ICS [16]	1.1422e+00	1.5802e+01	5.2848e+00	–	3.4241e+00
	ACS1	3.9605e-79	1.9351e-26	6.4692e-28	1.2480e-53	3.5327e-27
	ACS2	1.7161e-92	7.5373e-33	2.7931e-34	2.1040e-66	1.4505e-33
f_9	WPA [9]	−1	-0.9962	−0.9991	–	9.01e-04
	GCS [14]	−1	−1	−1	0	–
	PCS [18]	−1	−0.9998	−1	3.1784e-009	–
	ACS1	−1	−1	−1	0	0
	ACS2	−1	−1	−1	0	0
f_{10}	SFL-DE [4]	3.348e−01	3.221e+01	4.958	6.499	–
	ASGSO [8]	5.30e−03	3.31e−02	1.59e−02	–	–
	LDCS [15]	1.12230814e−42	2.89779804e−37	5.505e−41	4.33858e−81	–
	ACS1	2.8286e−104	1.3682e−41	7.0660e−43	7.8716e−84	2.8056e−42
	ACS2	1.1361e−97	2.0980e−42	7.2610e−44	1.4655e−85	3.8281e−43
f_{11}	OBDE [2,19]	0	4.484833499e−03	5.8039961e−04	–	1.103814401e−03
	CS-DE [19]	0	9.715909871e−3	2.5909093e−3	–	4.369987405e−3
	ACS1	1.2021e−06	9.6819e−04	2.5193e−04	5.0921e−08	2.2566e−04
	ACS2	1.3801e−05	2.8000e−03	4.4163e−04	2.8327e−07	5.3223e−04
f_{12}	SMMA [5]	3.55e−12	4.85e−12	4.22e−12	–	3.64e−13
	BBO [6]	1.49e−13	–	2.35e−13	–	7.54e−12
	BBO-EP [7]	–	–	6.66e−04	–	4.62e−05
	ACS1	6.4602e−44	8.3253e−16	2.8257e−17	2.3082e−32	1.5193e−16
	ACS2	1.0852e−42	2.2721e−14	8.5615e−16	1.6856e−29	4.1056e−15
f_{13}	ASGSO [9]	1.93e−09	1.01e−05	2.05e−06	–	–
	WPA [9]	0	0	0	–	0
	ASCS [20]	0	0	0	–	–
	ACS1	0	0	0	0	0
	ACS2	0	0	0	0	0
f_{14}	ASGSO [8]	4.35e−09	9.68e−05	4.25e−05	–	–
	WPA [9]	0	0	0	–	0
	ASCS [20]	0	5.505e−11	0	–	–
	ACS1	0	0	0	0	0
	ACS2	0	0	0	0	0

5.3. Markov modeling of ACS

Definition 3 (state transition of nest location)**.** Given $\forall y_1 = (x_1, pb_1) \in Y, \forall y_2 = (x_2, pb_2) \in Y$, the state of nest location transitions from y_1 to y_2 in the iteration of ACS, which is denoted as $T_y(y_1) = y_2$.

Theorem 1. *The probability of transition from y_1 to y_2 in the iteration of ACS can be expressed as*

$$P(T_y(y_1) = y_2) = P(x_1 \to x_1')P(pb_1 \to pb_1')P(x_1' \to x_2)P(pb_1' \to pb_2), \tag{11}$$

where $P(x_1 \to x_1')$ represents the corresponding position transition probability of Step 5 in ACS; $P(pb_1 \to pb_1')$denotes the state transition probability of the historical optimal position in Step 5; $P(x_1' \to x_2)$ represents the corresponding position transition probability of Step 6 in ACS; and $P(pb_1' \to pb_2)$ denotes the state transition probability of the historical optimal position in Step 6.

Similar proofs are provided in Literature [32].

Definition 4 (population state transition of nest location). For $\forall q_i = (y_{i1}, y_{i2}, \ldots y_{in}) \in Q, \forall q_j = (y_{j1}, y_{j2}, \ldots, y_{jn}) \in Q$ in the iteration of ACS, the population state of nest location transitions from q_i to q_j, which is represented by $T_q(q_i) = q_j$.

Theorem 2. *The probability of transition from q_i to q_j in the iteration of ACS can be expressed as*

$$P(T_q(q_i) = q_j) = \prod_{k=1}^{n} P(T_y(y_{ik}) = y_{jk}). \tag{12}$$

Similar proofs are provided in Literature [32].

Theorem 3. *The population state sequence $\{q(t) |\ t \geq 0\}$ in the ACS algorithm is a finite second Markov chain.*

Proof. The search space of any optimization algorithm is finite, as are x and pb in any nest state $y = (x, pb)$; thus, the stage space of the nest location is a finite space. Furthermore, the population state of a nest location $q = (y_1, y_2 \ldots y_n)$ is composed of n nest locations, where n represents a finite positive integer. Therefore, the population state of nest location q is finite. Theorem 2 indicates that the state transition probability is $P(T_q(q(t-1)) = q(t))$in the population state of nest location $\{q(t)|t \geq 0\}$ for $\forall\, q(t-1) \in Q, \forall\, q(t) \in Q$. This probability is determined according to the transition probabilities of all nest locations in the population of the nest location. Formula (11) suggests that the transition probability of any nest location in the population of nest locations can be expressed as $P(T_y(y-1)) = y(t) = P(x(t-1) \to x'(t-1))P(pb(t-1) \to pb'(t-1))p(x'(t-1) \to x(t))P(pb'(t-1) \to pb(t))$, where $P(x(t-1) \to x'(t-1))$, $P(pb(t-1) \to pb'(t-1))$, $P(x'(t-1) \to x(t)$, and $P(pb'(t-1) \to pb(t))$ are all relevant to x and pb at $t-1$.

Thus, $P(T_q(q-1)) = q(t)$ is relevant only to the nest location state $y_i(t-1), 1 \leq i \leq n$ at $t-1$. The population state of nest location $\{q(t)|t \geq 0\}$ exhibits a Markov property. $P(x(t-1) \to x'(t-1))$, $P(pb(t-1) \to pb'(t-1))$, $P(x'(t-1) \to x(t))$, and $P(pb'(t-1) \to pb(t))$ are irrelevant to t; hence, $P(T_y(y-1)) = y(t)$ is irrelevant to t. $P(T_q(q(t-1)) = q(t))$ is also irrelevant to t. In other words, the population state sequence of the nest location is secondary. Therefore, the population state sequence of nest location $\{q(t)|t \geq 0\}$ is a finite secondary Markov chain.

5.4. Convergence analysis of ACS

Definition 5 [32]. Suppose the global optimal solution of the optimization problem $\langle A, f\rangle$ is gb, and the optimal state set of the nest location is defined as $R = \{y = (x, pb)|f(pb) = f(gb), y \in Y\}$.

Theorem 4. *In ACS, the state sequence of nest location is $\{y(t)|t \geq 0\}$ and the state set of optimal nest location R is a closed set of the state space of nest location Y. Similar proofs are provided in Literature* [32].

Definition 6 [32]. Suppose the global optimal solution of the optimization problem $\langle A, f\rangle$ is gb, and the population state set of optimal nest locations is defined as $H = \{q = (y_1, y_2, \ldots, y_n|\exists y_i \in R, 1 \leq i \leq n\}$.

Theorem 5. *In ACS, the population state sequence of the nest location is $\{q(t)|\ t \geq 0\}$ and the population state set of optimal nest location H is a closed set of population state space Q.*

Similar proofs are provided in Literature [32].

Theorem 6 [32]. *In the population state set of nest location Q, no non-empty closed set B is derived from H. Therefore, $B \cap H = \varnothing$.*

Theorem 7 [32]. *When the iteration of the nest location in the population tends toward infinity, the sequence of the population state definitely enters optimal state set H.*

Theorem 8 [32]. *ACS converges to the global optimal.*

Proof. ACS saves the optimal location in each iteration, thereby guaranteeing a non-increase in the population. Thus, it satisfies convergence Condition 1 of the random algorithm. Theorem 7 indicates that following numerous iterations of ACS, the population sequence of the nest location enters into the optimal state. Therefore, the possibility that no global optimal solution is obtained after continuous infinite searches is 0. Thus, convergence Condition 1 of the random algorithm is satisfied. In this way, ACS converges to the global optimal.

Acknowledgments

This work was supported by the Natural Science Foundation of Inner Mongolia Autonomous Region no. 2013MS0119 and the Scientific Research Project of the Higher Education Institutions of Inner Mongolia Autonomous Region no. NJZY12070.

References

[1] D.E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co. Inc, Boston, 1989.
[2] S. Rahnamayan, R. Tizhoosh, M.M.A. Salama, Opposition based differential evolution, IEEE Trans. Evolut. Comput. 12 (1) (2008) 64–79.
[3] J.J. Liang, A.K. Qin, P.N. Suganthan, et al., Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (3) (2006) 281–295.

[4] B. He, L.X. Che, C.S. Liu, Novel hybrid shuffled frog leaping and differential evolution algorithm, Comput. Eng. Appl. 47 (18) (2011) 4–8.
[5] X. Chen, Y.Q. Zhou, Hybrid optimization algorithm of Monkey algorithm and pure method, Comput. Sci. 40 (11) (2013) 248–254 (in Chinese).
[6] H. Ma, An analysis of the equilibrium of migration models for biogeography-based optimization, Inf. Sci. 180 (18) (2010) 3444–3464.
[7] Z.H. Cai, W.Y. Gong, C.X. Ling, Study on the evolutionary program-based new biogeography optimization algorithm, Sys. Eng. Theory Prac. 30 (6) (2010) 1106–1112 (in Chinese).
[8] Z. Ouyang, Y.Q. Zhou, Self-adaptive step glowworm swarm optimization algorithm, J. Comput. Appl. 7 (2011) 021.
[9] H.S. Wu, F.M. Zhang, L.S. Wu, A new swarm intelligence algorithm – Wolf pack algorithm, Sys. Eng. Electron. 36 (11) (2013) 2430–2438 (in Chinese).
[10] X.S. Yang, S. Deb, Cuckoo Search via Levy Flight, Proceedings of World Congress on Nature & Biologically Inspired Computing, IEEE, India, 2009, pp. 210–214.
[11] R.B. Payne, M.D. Sornson, K. KlitZe, The Cuckoos, Oxford University Press, 2005.
[12] C. Brown, L.S. Liebovitch, R. Glendon, Lévy flights in Dobe Ju hoansi foraging patterns, Hum. Ecol. 35 (2007) 129–138.
[13] X.X. Hu, An improved cuckoo search of solving function optimization problem, Comput. Eng. Des. 34 (10) (2013) 3639–3642 (in Chinese).
[14] H.Q. Zheng, Y. Zhou, A novel cuckoo search optimization algorithm based on Gauss distribution, J. Comput. Inf. Syst. 8 (2012) 4193–4200.
[15] C.W. Qu, Cuckoo algorithm based on disturbance factor and decision-maker, Comput. Appl. Software 31 (7) (2014) 290–293 (in Chinese).
[16] E. Valian, S. Mohanna, S. Tavakoli, Improved cuckoo search algorithm for global optimization, Int. J. Commun. Inf. Tech. 1 (1) (2011) 31–44.
[17] M.H. Su, Y.L. Liu, T.B. Wu, An improved cuckoo search by solving unconstrained optimization problem, Comput. Eng. 40 (5) (2014) 224–227 (in Chinese).
[18] Y. Zhou, H. Zheng, A novel complex valued cuckoo search algorithm, Scientific World J. 2013 (2013) 597803.
[19] M. Li, D. Cao, Hybrid optimization algorithm of Cuckoo Search and DE, Comput. Eng. Appl. 49 (9) (2013) 57–60.
[20] H.J. Zheng, Y.Q. Zhou, Self-adaption step-length cuckoo search, Comput. Eng. Appl. 49 (10) (2013) 68–71 (in Chinese).
[21] X.S. Yang, Nature-inspired Metaheuristic Algorithms, 2nd ed., Luniver Press, Frome, 2010.
[22] R.N. Mantegna, Fast, accurate algorithm for numerical simulation of levy stable stochastic processes, Phys. Rev. E. 49 (5) (1994) 4677–4683.
[23] X.S. Yang, S. Deb, Engineering optimization by cuckoo search, Int. J. Math. Model. Numer. Opt. 1 (4) (2010) 330–343.
[24] R.L. Haupt, S.E. Haupt, Practical Genetic Algorithms, John Wiley & Sons, New York, 2004.
[25] D.H. Liu, S.C. Yan, Y. Lan, X.J. Ma, Chaos mapped particle swarm optimization method, J. Xidian Univ. 37 (4) (2010) 765–768 (in Chinese).
[26] W.B. Yu, X.S. Yang, X.P. Wei, Planar random polyline mapping in unit area and chaos analysis of its cross-iteration, Appl. Res. Comput. 28 (10) (2011) 3837–3841 (in Chinese).
[27] H. Liu, P. Zhu, Equivalent charactertions of Devaney chaos, J. Southwest Univ. National. (Nat. Sci. Ed.) 1 (2009) 015.
[28] X.F. He, Locality Preserving Projections, Proceedings of the 16th Conference on Neural Information Processing Systems, MIT Press, Cambridge, USA, 2003, pp. 153–160.
[29] S.K.S Fan, Y.Y. Chiu, A decreasing inertia weight particle swarm optimize, Eng. Opt. 39 (2) (2007) 203–228.
[30] J. Li, H. Sun, X.L. Shi, Study on the combination of multiple particle swarm algorithm and mixed frog leaping algorithm, J. Chinese Comput. Sys. 34 (9) (2013) 2164–2168 (in Chinese).
[31] F. Solis, R. Wets, Minimization by random search techniques, Math. Oper. Res. 6 (1981) 19–30.
[32] F. Wang, X.S. He, Y. Wang, et al., Markov model and convergence analysis based on cuckoo search algorithm, Comput. Eng. 38 (11) (2012) 180–185.