Research Article

Solving Logistics Distribution Center Location with Improved Cuckoo Search Algorithm

Juan Li¹,²,³, Yuan-Hua Yang¹, Hong Lei², Gai-Ge Wang⁴,⁵,*

¹School of Computer and Information Engineering, Hubei Normal University, Huangshi 435002, China
²School of Artificial Intelligence, Wuchang University of Technology, Wuhan 430223, China
³Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, 130012, China
⁴Department of Computer Science and Technology, Ocean University of China, 266100 Qingdao, China
⁵Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China

ABSTRACT

As a novel swarm intelligence optimization algorithm, cuckoo search (CS), has been successfully applied to solve various optimization problems. Despite its simplicity and efficiency, the CS is easy to suffer from the premature convergence and fall into local optimum. Although a lot of research has been done on the shortage of CS, learning mechanism has not been used to achieve the balance between exploitation and exploration. Based on this, a differential CS extension with balanced learning namely Cuckoo search algorithm with balanced-learning (O-BLM-CS) is proposed. Two sets, the better fitness set (FSL) and the better diversity set (DSL), are produced in the iterative process. Two excellent individuals are selected from two sets to participate in search process. The search ability is improved by learning their beneficial behaviors. The FSL and DSL learning factors are adaptively adjusted according to the individual at each generation, which improve the global search ability and search accuracy of the algorithm and effectively balance the contradiction between exploitation and exploration. The performance of O-BLM-CS algorithm is evaluated through eighteen benchmark functions with different characteristics and the logistics distribution center location problem. The results show that O-BLM-CS algorithm can achieve better balance between exploitation and exploration than other improved CS algorithms. It has strong competitiveness in solving both continuous and discrete optimization problems.

© 2021 The Authors. Published by Atlantis Press B.V.

This is an open access article distributed under the CC BY-NC 4.0 license (http://creativecommons.org/licenses/by-nc/4.0/).

1. INTRODUCTION

Lots of real-world problems can be converted to optimization problems, such as economic load dispatch, multi-robot path planning, wireless sensor networks, image segmentation, and radar applications. Optimization algorithms [1–3] are based on nature-inspired ideas with selecting the best alternative in a given objective function. In general, the optimization algorithms can be either a heuristic or a metaheuristic approach.

Rapid growth of the size and complexity of optimization problems implies a vital need for alternative optimization methods to the traditional mathematical optimization approaches [4]. Metaheuristic algorithms [5] have proved to be a viable solution to this challenge. Some of the well-known methods in this arena are genetic algorithms (GAs) [6–8], particle swarm optimization (PSO) [9–12], differential evolution (DE) [13–15], monarch butterfly optimization (MBO) [16–20], artificial bee colony (ABC) [21], earthworm optimization algorithm (EWA) [22], ant colony optimization (ACO) [23], chicken swarm optimization (CSO) [24], krill herd (KH) [25–27], firefly algorithm (FA) [28–33], simulated annealing (SA) [34], intelligent water drop (IWD) [35], water cycle algorithm (WCA) [36], moth search (MS) [37], monkey algorithm (MA) [38],
The CS algorithm has been applied successfully to diverse fields since it was proposed by Yang and Deb [66]. A number of CS variants have been developed to improve the performance of the CS algorithm. These variants can be generally divided into five categories, which are population topology and multi-swarm techniques, parameter control, local search operator, hybrid methods with other algorithms, and novel learning schemes.

Yang and Deb [77] proposed a modified CS to solve practical engineering problems. Li et al. [78] enhanced the exploitation ability of the CS algorithm by using knowledge learning strategy. Gandomi et al. [79] developed a new CS algorithm to solve truss optimization problems. Kamoona et al. [80] proposed a novel enhanced cuckoo search (ECS) algorithm for image contrast enhancement, which proposed a new range of search space for the parameters of the local/global enhancement (LGE) transformation that need to be optimized. Yang et al. [81] proposed a novel modified CS algorithm named as NMCSA to solve optimal placement of actuators for active vibration control, which minimized control spillover effect and maximized the control force applied to the desired modes. Majumder et al. [82] proposed a hybrid discrete cuckoo search (HDCS) algorithm to minimize makespan for this scheduling problem. In HDCS, a modified Lévy flight was proposed to transform a continuous position into a discrete schedule for generating a new solution. Ma et al. [83] proposed an improved dynamic self-adaptation CS algorithm based on collaboration between subpopulations.

Although much effort has been made to enhance the performance of CS, many of the variants CS fail to improve the performance of CS algorithm on complicated problems. For example, some CS variants still cannot solve the global optimum for difficult problems involving many local optima. Meanwhile, some CS variants are able to increase population diversity, but they may face problems like slow convergence speed.

In this paper, we proposed an improved CS algorithm namely O-BLM-CS that adopts balanced-learning strategies. Although a lot of research has been done on the shortage of CS, learning mechanism has not been used to achieve the balance between exploitation and exploration. Based on this, in O-BLM-CS, the better fitness set (FSL) and the better diversity set (DSL), are produced in the iterative process. Two excellent individuals are selected from two sets to participate in search processes. The search ability is improved by learning their beneficial behaviors. The FSL and DSL learning factors are adaptively adjusted according to the individual at each generation, which improve the global search ability and search accuracy of the algorithm and effectively balance the contradiction between exploitation and exploration. To verify the effectiveness of O-BLM-CS, we conducted comprehensive experiments on eighteen test functions and the logistics distribution center location problem. The experimental results show that O-BLM-CS performed better than other evolutionary algorithms in terms of the quality of the solution and convergence rate.

The main contributions of this study can be summarized as follows: (1) Fitness sorting learning mechanism (FSL) is introduced into individual updates, which improve the performance of algorithm exploitation. (2) Diversity sorting learning mechanism (DSL) is introduced into individual updates, which improve the performance of algorithm exploration. (3) Initialization with opposition-based learning model increases the chance for finding an individual close to the global best solution.

The remainder of this paper is organized as follows: Section 2 reviews the basic characteristics of CS, and then Section 3 describes balanced-learning model and O-BLM-CS algorithm steps. Benchmark problems and corresponding experimental results are given in Section 4. Finally, Section 5 concludes this paper and points out some future research directions.

2. CUCKOO SEARCH

The CS algorithm [66] is a nature-inspired evolutionary algorithm, which inspired by parasitism behavior of cuckoo species that lay eggs in other host birds. The algorithm is based on the obligate brood parasitic behavior found in some cuckoo nests by combining a model of this behavior with the principles of Lévy flights, which is a type of random walk with a heavy tail. CS is based on three idealized rules:

1. Each cuckoo lays one egg at a time, and places it in a randomly chosen nest.
2. The best nests with the highest quality eggs (solutions) will be carried over to the next generations.
3. The number of available host nests is fixed, and a host can discover an alien egg with the probability $P_a$. If the alien egg is discovered, the nest is abandoned and a new nest is built in a new location.

The position of the number $i$ nest are indicated by using $D$-dimensional vector, the offspring are produced by using Lévy flights (based on random walks). Lévy flight is performed as follows:

$$X_{i}^{t+1} = x_i^t + a \odot \text{levy}(\lambda) \quad (i = 1, 2, ..., n),$$

$$a = a_0 \odot (x_i^t - x_j^t)$$

where $x_i^t$ and $x_j^t$ are different solutions selected randomly. Product shows element by element multiplications, and $\lambda$ is the information about step size that is used to control the range of the random search. Information about step size that is more useful can be computed by using Eq. (2). After partial solutions are discarded, a new solution with the same number of cuckoos is generated by using Eq. (3).

$$X_{i}^{t+1} = x_i^t + r \left( X_m^t - X_n^t \right)$$

where $r$ generates a random number between $-1$ and $1$, $X_m^t$ and $X_n^t$ are random solutions for the $t$-th generation.

$$a = a_0 + (a_1 - a_0) \cdot d_i$$
\[ d_i = \frac{||x_i - x_{\text{best}}||}{d_{\text{max}}} \]  

where \( x_i \) represents the \( i \)-th nest position, \( x_{\text{best}} \) represents the optimal nest position, \( d_{\text{max}} \) is the maximum distance between the optimal nest and all other nests. \( a_0 \) and \( a_1 \) represent the minimum and maximum step size, respectively. The structure of CS algorithm is described in Algorithm 1.

**Algorithm 1: CS algorithm**

1. randomly initialize population of \( n \) host nests
2. calculate fitness value for each solution in each nest
3. while (stopping criterion is not meet do)
4. for \( i = 1 \) to \( n \)
5. generate new solution by using Levy flights;
6. choose candidate solution;
7. if \( f(x_i^t) > f(x_{i}^{t+1}) \)
8. replace with new solution;
9. end if
10. end for
11. throw out a fraction \( (p_a) \) of worst nests
12. for each abandoned nest \( k \in c \) do
13. for each \( i \in n \) do
14. generate solution \( k_i^{t+1} \) using Eq. (3)
15. if \( f(x_i^t) > f(x_{i}^{t+1}) \)
16. replace with new solution;
17. end if
18. end for
19. end for
20. rank the solution and find the current best
21. end while

3. IMPROVED CS ALGORITHM

3.1. Initialization with Opposition-Based Learning Model

Tizhoosh [84] proposed an affective technique namely opposition-based learning (OBL) for enhancing various algorithms. The OBL transforms candidates from current search space to a new search space which add the opposition solutions of individual. The OBL increases the chance for finding an individual close to the global best solution by evaluating a solution and its opposition solution. When a solution \( x \) is evaluated, the opposite solution \( \bar{x} \) will be evaluated simultaneously. The generalized opposite-based learning is defined as follows: Let \( X = (x_1, x_2, ..., x_n) \) be the point in \( D \)-dimensional space. The generalized opposite-based learning point \( \bar{X} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_D) \) is definite as follows:

\[ \bar{x}_i = k(a_i + b_i) - x_i \]  

where \( k \) is the random number between 0 and 1, \( a_i \) and \( b_i \) are boundary of the search space, respectively.

In order to improve the search performance of CS, the OBL idea is introduced the CS to initialize the population in this paper. We split the population into two subpopulations (P1 and P2). The subpopulation P1 is generated by a random distribution. The subpopulation P2 is initialized in terms of OBL strategy. The two subpopulations are composed of one population after updating the solutions in the population, which can make the population size unchanged in the optimization process. Furthermore, the population is sorted by their fitness and located the best individual.

3.2. Individual Update with Balanced-Learning Model

The balance between exploitation and exploration is an important goal for optimization algorithm. In this second, two learning models, FSL and DSL, are introduced into the CS algorithm to balance the performance of CS algorithm in terms of exploitation and exploration.

1. Fitness sorting learning mechanism

In order to improve the performance of algorithm exploitation, FSL is introduced into individual updates. According to the fitness value of individuals, population is sorted in descending order in FSL. The individual with the smallest fitness value is the best. The new population after sorted by fitness is shown in the Figure 1, the first individual is the worst, and the \( n \)-th individual is the best. \( X_i \) is the current \( i \)-th individual. \( \{X_{i+1}, ..., X_n\} \) contains all individuals whose fitness is better than the current individual fitness. An individual is randomly selected from \( \{X_{i+1}, ..., X_n\} \) and used to update process of the current individual.

2. Diversity sorting learning mechanism

In order to improve the performance of algorithm exploration, DSL is also introduced into individual updates. In DSL, according to the diversity of individuals, the individuals are sorted in descending order, in which the diversity \( d_{i,j} \) is evaluated by the Euclidean distance as shown in Eq. (7). The diversity matrix \( d \) can be calculated by Eq. (8). The diversity of each individual is denoted by the mean diversity \( \bar{d}(X_i) \). The distance-based diversity and fitness-based diversity are investigated in some researches. Yu et al. [85] make the first attempt to use both distance-based diversity \( D_d \) and fitness-based diversity \( D_f \) to control the mutation process to the best of our knowledge. It should be noticed that the DSL is introduced only into individual updates in this paper. Meanwhile, the two diversity measures are different.

\[ d_{i,j} = d(X_i, X_j) = \sqrt{\sum_{k=1}^{D} (X_{i,k} - X_{j,k})^2} \]  

\[ d = \begin{bmatrix} d_{1,1}, & d_{1,2}, & \ldots, & d_{1,n} \\ d_{2,1}, & d_{2,2}, & \ldots, & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1}, & d_{n,2}, & \ldots, & d_{n,n} \end{bmatrix} \]  

\[ \bar{d}(X_i) = \sum_{j=1,j\neq i}^{D} d_{ij}/(n-1) \]  

The individual with the biggest diversity value is the best. In the new population after diversity sorting, as shown in the Figure 2, the first individual is the worst, and the \( n \)-th individual is the best. \( X_0 \)
is the current $i$-th individual. $\{X_{q+1}, \ldots, X_n\}$ contains all individuals whose diversity is better than the current individual diversity. An individual is randomly selected from $\{X_{q+1}, \ldots, X_n\}$ and used for the updating process of the current individual.

$$x_{g+1,i} = x_{g,i} + a \oplus \text{levy}(\hat{\beta}) + R_1(x_{\text{FSL},g} - x_{g,i}) + R_2(x_{\text{DSL},g} - x_{g,i})$$

where the $x_{\text{FSL},g}$ is randomly chosen from $\{X_{q+1}, \ldots, X_n\}$ with better fitness than the current individual $x_{g,i}$. The $x_{\text{DSL},g}$ is randomly chosen from $\{X_{q+1}, \ldots, X_n\}$ with better diversity than the current individual $x_{i,g}$. $R_1$ is learning factor from $x_{\text{FSL},g}$ fitness learns behaviors. $R_3$ is learning factor from $x_{\text{DSL},g}$ diversity learns behaviors. $R_1$ and $R_2$ can be computed by using Eqs. (11) and (12), respectively.

$$R_1 = \left( f(X_i) - f_{\text{min}} \right) / \left( f_{\text{mean}} - f_{\text{min}} \right)$$

where $f(X_i)$ is the solution of the current, $f_{\text{min}}$ is the optimal solution at current generation, and $f_{\text{mean}}$ is the mean solution.

In the early stage of search, the algorithm tends to global search and needs a larger learning factor to improve the global search ability; in the later stage of search, the algorithm tends to local search and needs a smaller learning factor to improve the local search ability. The need for diversity changes as the number of iterations increases. Therefore, we adjust the learning factor $R_2$ according to the evolution generations.

$$R_2 = (T/t)^2$$

### 3.3. The Procedure Pseudo Code of O-BLM-CS Algorithm

FSL and DSL effectively balance the performance of CS algorithm in terms of exploitation and exploration. The structure of the CS algorithm with balance-Learning model algorithm (O-BLM-CS) can be described in Algorithm 2.

**Algorithm 2: O-BLM-CS algorithm**

1. initialize population in terms of opposition-based learning model
2. calculate fitness value for each solution in each nest.
3. for $i = 1$ to $T$ do
4. calculate the fitness values of individuals and sort according to fitness values.
5. calculate the diversity of individuals and sort according to diversity.
6. update the learning factor $R_2$ of DSL according to Eq. (12).
7. for $i = 1$ to $NP$ do
8. update the learning factor $R_1$ of FSL according to Eq. (12).
9. randomly choose the better individual from the FSL set.
10. randomly choose the better individual from the DSL set.
11. generate $x_{i+1}^t$ as new solution by using Eq. (12).
12. choose candidate solution.
13. if $f(x_i^t) > f(x_{i+1}^t)$
14. replace $x_i^t$ with new solution $x_{i+1}^t$;
15. end if
16. end for
17. throw out a fraction $(p_a)$ of worst nests.
18. for each abandoned nest $k \in c$ do
19. for each $i \in n$ do
20. generate solution $k_{i+1}$ using Eq. (3).
3.4. Analysis of Algorithm Complexity

The computational complexity of the O-BLM-CS algorithm is analyzed according to the steps in Algorithm 2. Let the population size and dimension are NP and D, respectively. Obviously, O-BLM-CS algorithm is just seven more steps, step (4) – (10), than the standard CS algorithm. In Algorithm 2, sorting by fitness in steps (4) has time complexity O(NP × D), sorting by diversity in steps (5) has time complexity O(NP × D). Calculating the learning factors R₁ and R₂ in steps (4) and (8) have time complexity O (1). Randomly choosing the better individual from the FSL set and DSL set in steps (9)–(10) have time complexity O(NP × D). Each iteration calculation time from step (3) to step (27) is O (NP × D). From the above results, after omitting the low-order terms, the total time complexity of the O-BLM-CS algorithm is O(T × NP × D), which is only related to T, NP, and D. The O-BLM-CS algorithm has the same computational complexity as standard CS algorithm. Consequently, the improvement of the algorithm does not increase the computational overhead.

4. RESULTS

4.1. Optimization of Functions and Parameter Settings

In order to verify the performance of O-BLM-CS algorithm, eighteen different global optimization problems were tested. F1–F5 are unimodal functions, F6–F11 are multimodal functions with many local minima, F12–F14 are shifted unimodal functions, and F15–F18 are shifted multimodal functions. A brief description of these benchmark problems is described in Table 1. The experiments were carried out on a P4 Dual-core platform with a 1.75 GHz processor and 4 GB memory, running under the Windows 7.0 operating system. The algorithms were developed using MATLAB R2017a. The maximum number of iterations, population size, and the times of running were set to 30,000, 30, and 30, respectively. The probability that foreign eggs were found was Pₐ = 0.25.

4.2. Comparison with Other CS Variants and Rank-Based Analysis

This section focuses on some of the recent developments of CS algorithms that are directly related to our study. We compared O-BLM-CS with standard CS [66] and four improved CS variants: Chaos cuckoo search algorithm (CCS) [86], Gaussian disturbance cuckoo search algorithm (GCS) [87], Combination of cuckoo search and particle swarm optimization (CSPSO) [88], Orthogonal learning cuckoo search algorithm (OLCS) [67]. CCS [86] proposed a modified Chaos enhanced CS algorithm, which enhances initialized host nest location. GCS [87] is a cuckoo algorithm with Gaussian disturbance. CSPSO [88] is a kind of algorithm combining CS with PSO. OLCS [67] used a new search strategy with orthogonal learning to enhance the exploitation ability of CS algorithm. The parameter configurations of these algorithms are shown in Table 2 according to corresponding references. Eighteen benchmark functions on 30-dimensional and 50-dimensional are tested. The same parameters are set for all algorithms. Population size NP = 30, FES = 10⁵ × D. The detailed results that O-BLM-CS compares five CS variants for D = 30 and D = 50 are summarized in Tables 3 and 4. Statistical results are shown in Tables 5 and 6. All experiments run 30 times. The best results in this table are bolded.

The average (Mean) and standard deviation (STD) with 30-dimensional and 50-dimensional are reported in Tables 3 and 4. Wilcoxon signed-rank test between O-BLM-CS and five algorithms (CS, CCS, GCS, OLCS, and CSPSO) at 30-dimensional and 50-dimensional was conducted in Tables 5 and 6 in which signs “+,” “-,” and “≈” indicate that the performance of O-BLM-CS is better than, less than and similar to other competitor.

The optimization results for 30-dimensional (30-D): From Table 3, O-BLM-CS can get global optima on functions F1, F7, F8, and F17 with 100% robustness. OLCS can get global optima on functions F3, F7, F8, and F17. For unimodal functions F1–F5, O-BLM-CS achieves higher accuracy than other algorithms on functions F1 and F4. OLCS achieves higher accuracy than other algorithms on functions F2, F3, and F5. O-BLM-CS is only inferior to OLCS on F2 and F3. For multimodal problems F6–F11, O-BLM-CS was significantly better than other algorithms on all functions. OLCS can get global optima on functions F7 and F8. For the shifted unimodal functions, O-BLM-CS achieves higher accuracy than other algorithms on F13 and F14. For the shifted multimodal functions F16 and F17, for F15, CCS is the best, for F16, OLCS is the best. Therefore, these statistical tests confirmed that O-BLM-CS algorithm with balanced-learning have better overall performance than other tested competitors. The ranking results of five algorithms are summed in Table 5.

The optimization results for 50-dimensional (50-D): From Table 3, for F7 and F8, only O-BLM-CS and OLCS can get global optima. For F3, F14, and F17, OLCS can get global optima. For unimodal functions F1–F5, O-BLM-CS is the best on F1, F2, and F5. OLCS achieves higher accuracy than other algorithms on F13 and F14. For the shifted multimodal functions F16 and F17, for F15, CCS is the best, for F16, OLCS is the best. Therefore, these tests confirmed that O-BLM-CS algorithm with balanced-learning have better overall performance than other tested competitors. The Wilcoxon signed-rank test results are summed in Table 6. We can see that the O-BLM-CS optimization algorithms explore a larger search space than other algorithms. Altogether, the obtained results on 30-dimensional and 50-dimensional reveal that O-BLM-CS provides appropriate level of exploration and exploitation trade-off over the considered problems.

The performance ranking of six algorithms is listed in Tables 7–10 based on the Wilcoxon test. When their performances are same,
Table 1 | Brief description of eighteen functions.

| Type            | Function | Name                  | Search Range      | Error Threshold | Global Optimum |
|-----------------|----------|-----------------------|-------------------|-----------------|----------------|
| Unimodal        | F1       | Sphere                | [-100, 100]       | 10^-6           | 0              |
|                 | F2       | Rosenbrock            | [-30, 30]         | 10^-6           | 0              |
|                 | F3       | Step                  | [-100, 100]       | 10^-6           | 0              |
|                 | F4       | Elliptic              | [-100, 100]       | 10^-6           | 0              |
|                 | F5       | Schwefel 2.22         | [-10, 10]         | 10^-6           | 0              |
|                 | F6       | Ackley                | [-32, 32]         | 10^-6           | 0              |
|                 | F7       | Rastrigin             | [-5.12, 5.12]     | 10^-6           | 0              |
| Multimodal      | F8       | Griewank              | [-600, 600]       | 10^-6           | 0              |
|                 | F9       | Schwefel 2.26         | [-500, 500]       | 10^-6           | 0              |
|                 | F10      | Generalized Penalized 1| [-50, 50]        | 10^-6           | 0              |
|                 | F11      | Generalized Penalized 2| [-50, 50]        | 10^-6           | 0              |
| Shifted unimodal| F12      | Shifted Sphere        | [-100, 100]       | 10^-6           | -450           |
|                 | F13      | Shifted Schwefels problem 1.2| [-100, 100] | 10^-6           | -450           |
|                 | F14      | Shifted rotated high conditioned elliptic function | [-100, 100] | 10^-6           | -450           |
| Shifted multimodal| F15    | Shifted Rosenbrock    | [-100, 100]       | 10^-2           | 390            |
|                 | F16      | Shifted rotated Ackleys| [-32, 32]        | 10^-2           | -140           |
|                 | F17      | Shifted rotated Griewanks| [-600, 600]   | 10^-2           | 0              |
|                 | F18      | Shifted rotated Rastrigin| [-5.12, 5.12]  | 10^-2           | -330           |

Table 2 | The personal parameters of different algorithms.

| Algorithms            | Parameter Configurations                       |
|-----------------------|-----------------------------------------------|
| CS [66]               | P_a = 0.25                                    |
| CCS [86]              | P_a = 0.2, a = 0.5, b = 0.2, xi = (0, 1)       |
| GCS [87]              | a = 1/3, P_a = 0.25                            |
| CSPSO [88]            | P_a = 0.25, a = 0.1, W = 0.9 ~ 0.4, c1 = c2 = 2.0 |
| OLCS [67]             | P_a = 0.2.a = 0.5,K = 9,Q = 3                  |
| O-BLM-CS              | P_a = 0.25, M = 3, y = 0.5                     |

Figure 3 | The convergence graphs of different algorithms on unimodal and multimodal functions (D = 30).

4.3. Application in the Problem of Logistics Distribution Center Location

4.3.1. Problem description

The distribution center is the most important hardware facility for logistics distribution center in the logistics system. All the logistics activities are almost entirely carried out with the distribution center. The positioning of the distribution center almost determines the cost required for the distribution business, which is a very
The logistics distribution center location selects a certain number of locations in a number of known sites, which minimize the total cost of forming the logistics network. This type of problem with the nature of NP-hard problems is a nonlinear model with more complex constraints and non-smooth characteristics. The problem can be described as: m cargo distribution center are search in n demand point, so the distance between m searched distribution centers and other n cargo demand points is the shortest. The constraint conditions are as follows:

1. The supply of goods in the distribution center can meet the requirements of the cargo demand point;
2. The goods required for a cargo demand point can only be provided by one distribution center;

3. The cost of transporting the goods to the distribution center is not considered.

According to the above assumptions, the mathematical model of the problem for logistics distribution center location can be described as:

\[
\min (\cos t) = \sum_{i=1}^{m} \sum_{j=1}^{n} (\text{need}_j \cdot \text{dist}_{ij} \cdot \mu_{ij}) \\
s.t. \sum_{i=1}^{m} \mu_{ij} = 1, i \in M, j \in N \\
\mu_{ij} \leq h_j, i \in M, j \in N \\
\sum_{i=1}^{m} h_i \leq p, i \in M
\]
Table 10 | Rank table for the mean values of 50-dimensional cases on F10–F18.

| Algorithm | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| CCS       | 6   | 6   | 6   | 6   | 6   | 6   | 6   | 2   |     |
| CCS       | 5   | 4   | 5   | 3   | 3   | 3   | 1   | 3   | 3   |
| GCs       | 3   | 3   | 4   | 5   | 4   | 5   | 4   | 5   | 5   |
| CSPSO     | 5   | 4   | 5   | 1   | 5   | 4   | 5   | 4   | 6   |
| OLCS      | 2   | 2   | 2   | 5   | 1   | 2   | 3   | 1   | 4   |
| O-BLM-CS  | 1   | 1   | 1   | 2   | 1   | 2   | 1   |     |     |

It can be seen from Figure 5a, the optimal convergence curve of CS in 6 distribution centers for 100 iterations can converge at 20 iterations, but the average convergence curve is not converged. The optimal distribution, average distribution, and the worst distribution cost obtained in 6 distribution centers for 100 iterations are 4.7345E04, 5.9312E04, and 6.8017E04, respectively. From Figure 6a, the optimal convergence curve of CS in 4 distribution centers for 50 iterations can converge at 100 iterations, the average convergence curve can converge at 250 iterations. The optimal distribution, average distribution, and the worst distribution cost obtained in 4 distribution centers for 500 iterations are 6.5268E04, 7.4188E04, and 7.7992E04, respectively. From Figure 7a, the optimal convergence curve of CS in 6 distribution centers for 500 iterations can converge at 50 iterations, the average convergence curve can converge at 200 iterations. The optimal distribution, average distribution, and the worst distribution cost obtained in 6 distribution centers for 500 iterations are 4.7128E04, 5.3999E04, and 5.6927E04, respectively. From Figure 8, the optimal convergence curve of CS in 10 distribution centers for 500 iterations can converge at 300 iterations, the average convergence curve is not converged. The optimal distribution, average distribution, and the worst distribution cost obtained in 10 distribution centers for 500 iterations are 3.0265E04, 3.4765E04, and 3.8255E04, respectively.

Table 11 | Total rank and final rank on F1–F18.

| Dim | Rank | Algorithms |
|-----|------|------------|
|     | Total rank | CS | CCS | GCs | CSPSO | OLCS | O-BLM-CS |
| 30  | 107  | 61  | 64  | 80  | 39.5  | 29.5  |          |
|     | Final rank | 6   | 3   | 4   | 5   | 2     | 1     |          |
| 50  | 105  | 66  | 71  | 78  | 37    | 27    |          |
|     | Final rank | 6   | 3   | 4   | 5   | 2     | 1     |          |

where Eq. (13) represents the objective function, \( m \) indicates the number of logistics distribution center, \( n \) is the number of goods demand point, cost is the transportation cost, \( \text{nest} \) determines the demand quantity of demand point \( j \), \( \text{dist}_{ij} \) indicates the distance between distribution center \( i \) and goods demand point \( j \). Eqs. (14–19) are the constraints. Eq. (14) indicates that a demand point of goods can only be distributed by a distribution center, Eq. (15) defines that each demand point of goods must have a distribution center to distribute goods, and Eq. (16) indicates the number of goods demand points for a distribution center.

4.3.2. Analysis of experimental results

In this section, there is a logistics network with 40 demand points. The geographical position coordinates and demands were shown in Table 12. The maximum number of iterations \( T = 30,000 \), population size \( NP = 15 \), and the times of running is 30.

For the first set of experiments, the effectiveness of the O-BLM-CS is verified by comparing CS algorithms, 4, 6, and 10 points were selected as the address of the distribution center to minimize the sum of all costs in this experiment. When the number of iterations \( T = 100 \), for running 20, 30, and 50 times in 6 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by CS are shown in Figure 5. When the number of iterations \( T = 500 \), for running 20, 30, and 50 times in 6 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by CS are shown in Figure 6. When the number of iterations \( T = 500 \), for running 20, 30, and 50 times in 4 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by CS are shown in Figure 7. When the number of iterations \( T = 500 \), for running 20, 30, and 50 times in 10 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by CS are shown in Figure 8.

For the second set of experiments, the O-BLM-CS algorithm is run 20, 30, and 50 times independently in 40 cities 4, 6, and 10 distribution centers. When the number of iterations \( T = 100 \), for 6 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by O-BLM-CS are shown in Figure 9. When the number of iterations \( T = 500 \), for 6 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by O-BLM-CS are shown in Figure 10. When the number of iterations \( T = 500 \), for 10 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by O-BLM-CS are shown in Figure 11. When the number of iterations \( T = 500 \), for 10 distribution centers, the average convergence curve, optimal convergence curve, and the optimal route found by O-BLM-CS are shown in Figure 12. Table 17 shows distribution ranges 100 iterations for 4, 6, and 10 distribution centers, respectively.
Table 12 | The geographical position coordinates and demands.

| No | Coordinates | Demand | No | Coordinates | Demand | No | Coordinates | Demand |
|----|-------------|--------|----|-------------|--------|----|-------------|--------|
|    | x  | y  |    | x  | y  |    | x  | y  |    |
| 1  | 97 | 28 | 94 | 11 | 91 | 96 | 85 | 21 | 111 |
| 2  | 100| 56 | 11 | 12 | 39 | 90 | 54 | 22 | 63 |
| 3  | 45 | 67 | 50 | 13 | 50 | 101| 25 | 23 | 67 |
| 4  | 150| 197| 50 | 17 | 169| 95 | 48 | 27 | 143 |
| 5  | 105| 48 | 80 | 15 | 157| 54 | 66 | 25 | 100 |
| 6  | 24 | 158| 29 | 16 | 104| 35 | 82 | 26 | 35 |
| 7  | 88 | 61 | 93 | 17 | 169| 95 | 48 | 27 | 143 |
| 8  | 55 | 105| 10 | 18 | 48 | 39 | 78 | 28 | 94 |
| 9  | 120| 120| 18 | 19 | 115| 61 | 16 | 29 | 57 |
| 10 | 43 | 105| 38 | 20 | 154| 174| 49 | 30 | 25 |

(a)  

Figure 5 | Convergence curves and optimal distribution centers scheme for the cuckoo search (CS) algorithm in 6 distribution centers ($T = 100$).

(b)  

Figure 6 | Convergence curves and optimal distribution centers scheme for the cuckoo search (CS) algorithm in 4 distribution centers ($T = 500$).

From Figure 10a, the optimal convergence curve of O-BLM-CS in 4 distribution centers for 500 iterations can converge at 20 iterations, the average convergence curve can converge at 15 iterations. The optimal distribution, average distribution, and the worst distribution cost obtained in 4 distribution centers for 500 iterations are 6.3813E04, 6.4194E04, and 6.4231E04, respectively. From Figure 11a, the optimal convergence curve of O-BLM-CS in 6 distribution centers for 500 iterations can converge at 5 iterations, the average convergence curve can converge at 50 iterations. The optimal distribution, average distribution, and the worst distribution cost obtained in 6 distribution centers for 500 iterations are 4.5021E04, 4.5181E04, and 4.6023E04, respectively. From Figure 12a, the optimal convergence curve of O-BLM-CS in 10 distribution centers for 500 iterations can converge at 10 iterations,
Table 13 | The distribution scheme for the cuckoo search (CS) algorithm in 6 distribution center \((T = 100)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 10                  | 33, 6, 30, 12, 13, 8, 23 |
| 25                  | 11, 21, 9 |
| 27                  | 4, 38, 20, 24, 39 |
| 22                  | 14, 29, 36, 26, 18, 37 |
| 5                   | 7, 28, 2, 1, 25, 16, 19, 34, 35 |
| 15                  | 31, 17, 32, 40 |

Table 14 | The distribution scheme for the cuckoo search (CS) algorithm in 4 distribution center \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 8                   | 33, 6, 30, 10, 12, 13, 23, 25, 21, 11 |
| 22                  | 26, 36, 3, 29, 14, 18, 37, 7, 28, 2, 34, 5, 16, 1, 35 |
| 27                  | 4, 9, 20, 24, 38, 39 |
| 15                  | 19, 31, 17, 32, 40 |

Table 15 | The distribution scheme for the cuckoo search (CS) algorithm in 6 distribution centers \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 30                  | 33, 6 |
| 23                  | 11, 25, 9, 21, 10, 13, 12, 8, 29 |
| 20                  | 4, 27, 38, 24, 39 |
| 18                  | 14, 3, 36, 26, 37, 22 |
| 1                   | 28, 7, 34, 2, 5, 3, 16, 5 |
| 15                  | 31, 17, 32, 40 |

Table 16 | The distribution scheme for the cuckoo search (CS) algorithm in 10 distribution centers \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 11                  | - |
| 21                  | 25, 9 |
| 20                  | 4, 27, 38, 24, 39 |
| 14                  | 3, 29 |
| 8                   | 12, 10, 13, 23 |
| 30                  | 6, 33 |
| 5                   | 7, 28, 2, 19, 31 |
| 22                  | 36, 26, 18, 37, 7 |
| 1                   | 34, 35 |
| 15                  | 17, 32, 40 |

Table 17 | The distribution scheme for the O-BLM-CS algorithm in 6 distribution centers \((T = 100)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 10                  | 33, 6, 30, 12, 13, 8, 23 |
| 11                  | 25, 21, 9 |
| 20                  | 4, 38, 27, 24, 39 |
| 22                  | 14, 29, 3, 36, 26, 18, 37, 7 |
| 1                   | 28, 2, 5, 16, 34, 35, 19 |
| 15                  | 31, 17, 32, 40 |

Table 18 | The distribution scheme for the O-BLM-CS algorithm in 4 distribution centers \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 23                  | 33, 6, 30, 10, 12, 13, 8, 25, 21, 11, 9 |
| 22                  | 26, 36, 3, 29, 14, 18, 37, 7, 28, 2, 34, 5, 16, 1, 35 |
| 27                  | 4, 9, 20, 24, 38, 39 |
| 15                  | 19, 31, 17, 32, 40 |

Table 19 | The distribution scheme for the O-BLM-CS algorithm in 6 distribution centers \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 13                  | 33, 6, 30, 12, 10, 8, 23, 29 |
| 21                  | 11, 25, 9 |
| 20                  | 4, 27, 38, 24, 39 |
| 22                  | 14, 3, 36, 26, 37, 18 |
| 16                  | 28, 7, 34, 2, 5, 19, 35, 1 |
| 15                  | 31, 17, 32, 40 |

Table 20 | The distribution scheme for the O-BLM-CS algorithm in 10 distribution centers \((T = 500)\).

| Distribution Center | Distribution Scope |
|---------------------|---------------------|
| 11                  | - |
| 21                  | 25, 9 |
| 20                  | 4, 27, 38, 24, 39 |
| 14                  | 3, 29 |
| 8                   | 12, 10, 13, 23 |
| 30                  | 6, 33 |
| 28                  | 7, 5, 2, 19, 31, 34 |
| 22                  | 36, 26, 18, 37, 7 |
| 1                   | 35, 16 |
| 15                  | 17, 32, 40 |

The average convergence curve can converge at 10 iterations. The optimal distribution, average distribution and the worst distribution cost obtained in 10 distribution centers for 500 iterations iterations are 2.8234E04, 3.0618E04, and 3.1886E04, respectively.

Table 17 shows distribution ranges 100 iterations for 6 distribution centers in 40 cities, and Tables 18–20 show distribution ranges 500 iterations for 4, 6, and 10 distribution centers, respectively. From Tables 17–20, the optimal distribution center points found by O-BLM-CS algorithm in 6 distribution centers for 100 iterations is (10, 11, 20, 22, 1, 5). The optimal distribution center points in 4, 6, and 10 distribution centers for 500 iterations are (23, 22, 27, 15), (12, 21, 20, 22, 16, 15), and (11, 21, 20, 14, 8, 30, 28, 22, 1, 15), respectively.

It can be seen from Figures 5 and 9, logistics distribution location strategy of O-BLM-CS in 6 distribution centers at 100 iterations for 10 distribution centers is better than CS in both the optimal convergence curve and the average convergence curve. The average convergence curve of O-BLM-CS can converge at 15 iterations, but CS does not converge to the optimal solution. It can be seen from Figures 6a and 10a, the average convergence curve of O-BLM-CS in 4 distribution centers at 500 iterations can converge at 15 iterations. the average convergence curve of CS can converge at 250 iterations, which in include O-BLM-CS is far superior to CS for terms of convergence speed. Although the CS algorithm can converge, it has a lot of noise for the average convergence curve. For 500 iterations and 10 distributions, CS converges only at 200 iterations, while O-BLM-CS converges to the optimal solution at 50 iterations. It is worth mentioning that CS does not converge to the optimal solution at 100 iterations for 10 distributions. O-BLM-CS converges to the optimal solution at just 10 iterations. It indicates that O-BLM-CS has fast speed and high solution accuracy, which effectively reduces the cost for 10 distribution centers.
Figure 7 | Convergence curves and optimal distribution centers scheme for the (CS) algorithm in 6 distribution centers ($T = 500$).

Figure 8 | Convergence curves and optimal distribution centers scheme for the cuckoo search (CS) algorithm in 10 distribution centers ($T = 500$).

Figure 9 | Convergence curves and optimal distribution centers scheme for the O-BLM-CS algorithm in 6 distribution centers ($T = 100$).
Figure 10 | Convergence curves and optimal distribution centers scheme for the O-BLM-CS algorithm in 4 distribution centers ($T = 500$).

Figure 11 | Convergence curves and optimal distribution centers scheme for the O-BLM-CS algorithm in 6 distribution centers ($T = 500$).

Figure 12 | Convergence curves and optimal distribution centers scheme for the O-BLM-CS algorithm in 10 distribution centers ($T = 500$).
of logistics distribution. In addition, the results of STD indicate that the O-BLM-CS has a better robustness than the other algorithms.

In this section, O-BLM-CS is compare with CS about optimization accuracy. Table 21 shows the comparison results with the best fitness value (Best), average fitness value (Mean), the worst fitness value (Worst), STD, and running time (Time). It can conclude that the average distribution cost of O-BLM-CS for 100 iterations in 6 distribution centers is 4.6082E4 which is 13230 lower than CS. The average distribution cost for 500 iterations in 4 distribution centers in 4.6082E4 which is 13230 lower than CS. The average distribution cost for 500 iterations in 6 distribution centers is 4.7492E+04 which is 8810 lower than CS. The average distribution cost for 500 iterations in 10 distribution centers is 3.0618E+04 which is 4147 lower than CS.

Based on the above analysis, it can be known that O-BLM-CS found the optimal route compared with CS in 4, 6, and 10 distribution centers. The results of O-BLM-CS are better than CS in terms of optimal value, average value, worst value, or running time. The reason may be that the balanced-learning strategy with diversity and adaptability improve the global search ability and search accuracy of the algorithm and effectively balance the contradiction between exploration and exploitation. The opposition learning operator accelerate the convergence speed of the algorithm. Meanwhile, the running time of O-BLM-CS is significantly lower than CS, and the number of iterations is significantly reduced. O-BLM-CS algorithm can select the address of logistics distribution center more quickly and accurately compared with CS algorithm. Finally, we can say that the O-BLM-CS outperforms CS in terms of convergence rate and robustness.

5. CONCLUSIONS

In this paper, an improved CS algorithm with balanced-learning scheme namely BLM-CS has been proposed. Two sets, the better adaptive set (FSL) and the DSL, are produced in the iterative process. Two excellent individuals are selected from two sets to participate in search process. The search ability is improved by learning their beneficial behaviors. The FSL and DSL learning factors are adaptively adjusted according to the individual at each generation, which improve the global search ability and search accuracy of the algorithm and effectively balance the contradiction between exploration and exploitation. The performance of BLM-CS algorithm is evaluated through fifteen benchmark functions with different characteristics. The results show that BLM-CS algorithm can achieve better balance between explore and exploit than other improved CS algorithms. It has strong competitiveness in solving the continuous optimization problems. In order to verify the performance of O-BLM-CS, this algorithm is applied to solve the problem of logistics distribution center location. The effectiveness of the proposed method is verified by comparing with other algorithms in both 6 distribution centers and 10 distribution centers.

In the future, the O-BLM-CS algorithm combined with other optimization algorithms will be the focus for us. We will determine how to generalize our work to handle combinatorial optimization problems and to extend O-BLM-CS optimization algorithms to the realistic engineering areas. The further studying during the next few years are shown as follow.

1. Employing the O-BLM-CS to solve unsolved optimization problems, especially multi-objective optimization problems [89,90], will be a challenge in future research.

2. Hybridizing the O-BLM-CS with other algorithm components such as DE and hill climbing is also a challenge in future research work [91-93].

3. O-BLM-CS has achieved some notable accomplishments in solving discrete and continuous optimization problems. Therefore, expanding the application scope of O-BLM-CS and designing suitable optimization operators will be a challenge in future research.

4. Expanding O-BLM-CS for more constrained optimization applications is also an important challenge in future research work [94,95].

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

AUTHORS’ CONTRIBUTIONS

Conceptualization, J.L.; methodology, H.L.; software, G.-g.W.; validation, J.L.; writing—original draft preparation, J.L. and Y.-h.Y.; writing—review and editing, G.-g.W.; All authors have read and agreed to the published version of the manuscript.

Funding Statement

This work was supported by the fundamental research funds for the central universities (93K172020K08).
ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and the editor for their careful reviews and constructive suggestions to help us improve the quality of this paper.

REFERENCES

[1] J. Li, H. Lei, A.H. Alavi, G.-G. Wang, Elephant herding optimization: variants, hybrids, and applications, Math. Basel. 8 (2020), 1415.

[2] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, M. Zhou, Chaotic local search-based differential evolution algorithms for optimization, IEEE Trans. Syst. Man Cybern. Syst. (2019), 1–14.

[3] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, J. Wang, Bi-objective elite differential evolution algorithm for multivalued logic networks, IEEE Trans. Cybern. 50 (2020), 233–246.

[4] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, J. Wang, Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019), 601–614.

[5] G.-G. Wang, Y. Tan, Improving metaheuristic algorithms with information feedback models, IEEE Trans. Cybern. 49 (2019), 542–555.

[6] K. Deb, An introduction to genetic algorithms. Sadhan. 24(4-5): (1999), 293–315.

[7] H. Garg, A hybrid gsa-ga algorithm for constrained optimization problems, Inf. Sci. 478 (2019), 499–523.

[8] H. Garg, A hybrid pso-ga algorithm for constrained optimization problems, Appl. Math. Comput. 274 (2016), 292–305.

[9] J. Kennedy, R. Eberhart, Particle swarm optimization, in Proceeding of the IEEE International Conference on Neural Networks, IEEE, Perth, Australia, 1995, pp. 1942–1948.

[10] G.-G. Wang, A.H. Gandomi, X.-S. Yang, A.H. Alavi, A novel improved accelerated particle swarm optimization algorithm for global numerical optimization, Eng. Comput. 31 (2014), 1198–1220.

[11] J. Sun, B. Feng, W. Xu, Particle swarm optimization with particles having quantum behavior, in Proceedings of Congress on Evolutionary Computation (CEC 2004), Portland, OR, USA, 2004, pp 325–331.

[12] A.O. Adewumi, M.A. Arasomwan, On the performance of particle swarm optimisation with(out) some control parameters for global optimisation, Int. J. Bio-Inspir. Comput. 8 (2016), 14–32.

[13] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997), 341–359.

[14] Z. Xu, A. Unveren, A. Acan, Probability collectives hybridised with differential evolution for global optimisation, Int. J. Bio-Inspir. Comput. 8 (2016), 133–153.

[15] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, M. Shao, Path planning for uninhabited combat aerial vehicle using hybrid metaheuristic de/bbo algorithm, Adv. Sci. Eng. Med. 4 (2012), 550–564.

[16] G.-G. Wang, X. Zhao, S. Deb, A novel monarch butterfly optimization with greedy strategy and self-adaptive crossover operator, in 2015 2nd International Conference on Soft Computing & Machine Intelligence (ISCMI 2015), IEEE, Hong Kong, 2015, pp. 45–50.

[17] G.-G. Wang, S. Deb, X. Zhao, Z. Cui, A new monarch butterfly optimization with an improved crossover operator, Oper. Res. Int. J. 18 (2018), 731–755.

[18] Y. Feng, G.-G. Wang, W. Li, N. Li, Multi-strategy monarch butterfly optimization algorithm for discounted [0-1] knapsack problem, Neural Comput. Appl. 30 (2018), 3019–3036.

[19] G.-G. Wang, S. Deb, Z. Cui, Monarch butterfly optimization, Neural Comput. Appl. 31 (2019), 1995–2014.

[20] Y. Feng, G.-G. Wang, S. Deb, M. Lu, X. Zhao, Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization, Neural Comput. Appl. 28 (2017), 1619–1634.

[21] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm, J. Global Optim. 39 (2007), 459–471.

[22] G.-G. Wang, S. Deb, L.d.S Coelho, Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, Int. J. Bio-Inspir. Comput. 12 (2018), 1–22.

[23] M. Dorigo, T. Stützel, Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.

[24] X. Meng, Y. Liu, X. Gao, H. Zhang, A new bio-inspired algorithm: chicken swarm optimization, in: Y. Tan, Y. Shi, C.C. Coello (Eds.), Advances in Swarm Intelligence, vol. 8794, Springer International Publishing, Hefei, China, 2014, pp. 86–94.

[25] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimisation algorithm, Commun. Nonlinear Sci. 17 (2012), 4031–4045.

[26] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic krill herd algorithm, Inf. Sci. 274 (2014), 17–34.

[27] G.-G. Wang, A.H. Gandomi, A.H. Alavi, D. Gong, A comprehensive review of krill herd algorithm: variants, hybrids and applications, Artif. Intell. Rev. 51 (2019), 119–148.

[28] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Mixed variable structural optimization using firefly algorithm, Comput. Struct. 89 (2011), 2325–2336.

[29] X.S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspir. Comput. 2 (2010), 78–84.

[30] G.-G. Wang, L. Guo, H. Duan, H. Wang, A new improved firefly algorithm for global numerical optimization, J. Comput. Theor. Nanosci. 11 (2014), 477–485.

[31] A. Galvez, A. Iglesias, New memetic self-adaptive firefly algorithm for continuous optimisation, Int. J. Bio-Inspir. Comput. 8 (2016), 300–317.

[32] B. Nasiri, M.R. Meybodi, History-driven firefly algorithm for optimisation in dynamic and uncertain environments, Int. J. Bio-Inspir. Comput. 8 (2016), 326–339.

[33] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A modified firefly algorithm for uav path planning, Int. J. Hybrid Inf. Technol. 5 (2012), 123–144.

[34] S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, Optimization by simulated annealing, Science. 220 (1983), 671–680.

[35] H. Shah-Hosseini, The intelligent water drops algorithm: a bio-inspired water swarm optimization algorithm, Int. J. Bio-Inspir. Comput. 2 (2010), 78–84.

[36] J. Kennedy, R. Eberhart, Particle swarm optimization, in Proceeding of the IEEE International Conference on Neural Networks, IEEE, Perth, Australia, 1995, pp. 1942–1948.

[37] G.-G. Wang, Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, Memetic Comput. 10 (2018), 151–164.
[38] K. Gupta, K. Deep, J.C. Bansal, Spider monkey optimization algorithm for constrained optimization problems. 21(23): (2017), 6933–6962.

[39] H. Beyer, The Theory of Evolution Strategies, Springer, New York, NY, USA, 2001.

[40] K. Penev, G. Littlefair, Free search—a comparative analysis, Inf. Sci. 172 (2005), 173–193.

[41] S. Baluja, Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[42] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008), 702–713.

[43] S. Mirjalili, S.M. Mirjalili, A. Lewis, Let a biogeography-based optimizer train your multi-layer perceptron, Inf. Sci. 269 (2014), 188–209.

[44] H. Duan, W. Zhao, G. Wang, X. Feng, Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm ts/bbo, Math. Probl. Eng. 2012 (2012), 1–22.

[45] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm, J. Sensor Actuator Netw. 1 (2012), 86–96.

[46] H. Garg, An efficient biogeography based optimization algorithm for solving reliability optimization problems, Swarm Evol. Comput. 24 (2015), 1–10.

[47] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Comput. Appl. 27 (2016), 1053–1073.

[48] A.H. Gandomi, Interior Search Algorithm (ISA): a novel approach for global optimization, ISA Trans. 53 (2014), 1168–1183.

[49] Y. Shi, An optimization algorithm based on brainstroming process, Int. J. Swarm Intell. Res. 2 (2011), 35–62.

[50] Y. Shi, J. Xue, Y. Wu, Multi-objective optimization based on brain storm optimization algorithm, Int. J. Swarm Intell. Res. 4 (2013), 1–21.

[51] A.H. Gandomi, X.-S. Yang, A.H. Alavi, S. Talatahari, Bat algorithm for constrained optimization tasks, Neural Comput. Appl. 22 (2013), 1239–1255.

[52] X.S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng. Comput. 29 (2012), 464–483.

[53] S. Mirjalili, S.M. Mirjalili, X.-S. Yang, Binary bat algorithm, Neur. Comput. Appl. 25 (2013), 663–681.

[54] G.-G. Wang, H.E. Chu, S. Mirjalili, Three-dimensional path planning for uav using an improved bat algorithm, Aerosp. Sci. Technol. 49 (2016), 231–238.

[55] X. Cai, X.-z. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, Int. J. Bio-Inspir. Comput. 8 (2016), 205–214.

[56] G. Wang, L. Guo, A novel hybrid bat algorithm with harmony search for global numerical optimization, J. Appl. Math. 2013 (2013), 1–21.

[57] G.-G. Wang, B. Chang, Z. Zhang, A multi-swarm bat algorithm for global optimization, in 2015 IEEE Congress on Evolutionary Computation (CEC 2015), IEEE, Sendai, Japan, 2015, pp. 480–485.

[58] G.-G. Wang, M. Lu, X.-J. Zhao, An improved bat algorithm with variable neighborhood search for global optimization, in 2016 IEEE Congress on Evolutionary Computation (IEEE CEC 2016), IEEE, Vancouver, Canada, 2016, pp. 1773–1778.

[59] X.-Y. Yang, Nature-Inspired Metaheuristic Algorithms, 2nd ed., Luniver Press, Frome, UK, 2010.

[60] W. Khattab, P. Fleming, The stud ga: a mini revolution?, in: A. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN v, vol. 1498, Springer, London, UK, 1998, pp. 683–691.

[61] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation. 76 (2001), 60–68.

[62] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, Hybridizing harmony search with biogeography based optimization for global numerical optimization, J. Comput. Theor. Nanosci. 10 (2013), 2318–2328.

[63] T. Niknam, A.K. Fard, Optimal energy management of smart renewable micro-girds in the reconfigurable systems using adaptive harmony search algorithm, Int. J. Bio-Inspir. Comput. 8 (2016), 184–194.

[64] A. Rezoug, D. Boughaci, A self-adaptive harmony search combined with a stochastic local search for the 0-1 multidimensional knapsack problem, Int. J. Bio-Inspir. Comput. 8 (2016), 234–239.

[65] Y. Tan, Fireworks Algorithm—a Novel Swarm Intelligience Optimization Method, Springer-Verlag, Berlin, Heidelberg, Germany, 2015.

[66] X.-S. Yang, S. Deb, Cuckoo search via levy flights, in: A. Abraham, A. Carvalho, F. Herrera, V. Pai (Eds.), Proceeding of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), IEEE Publications, Coimbatore, India, 2009, pp. 210–214.

[67] X. Li, J. Wang, M. Yin, Enhancing the performance of cuckoo search algorithm using orthogonal learning method, Neural Comput. Appl. 24 (2013), 1233–1247.

[68] X. Li, M. Yin, Modified cuckoo search algorithm with self adaptive parameter method, Inf. Sci. 298 (2015), 80–97.

[69] G.-G. Wang, A.H. Gandomi, X. Zhao, H.E. Chu, Hybridizing harmony search algorithm with cuckoo search for global numerical optimization, Soft Comput. 20 (2016), 273–285.

[70] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. 20 (2016), 3349–3362.

[71] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, J. Wang, A hybrid meta-heuristic de/cs algorithm for uav path planning, J. Inform. Comput. Sci. 9 (2012), 4811–4818.

[72] J. Li, Y.-x. Li, S.-s. Tian, J. Zou, Dynamic cuckoo search algorithm based on taguchi opposition-based search, Int. J. Bio-Inspir. Comput. 13 (2019), 59–69.

[73] J. Li, D.-d. Xiao, H. Lei, T. Zhang, T. Tian, Using cuckoo search algorithm with q-learning and genetic operation to solve the problem of logistics distribution center location, Math. Basel. 8 (2020), 149.

[74] J. Li, Xiao, D.-d. T. Zhang, C. Liu, Y.-x. Li, G.-g. Wang, Multi-swarm cuckoo search algorithm with q-learning model, Comput. J. (2020).

[75] H. Garg, An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm, Beni-Suef Univ. J. Basic Appl. Sci. 4 (2015), 14–25.

[76] H. Garg, Multi-objective optimization problem of system reliability under intuitionistic fuzzy set environment using cuckoo search algorithm, J. Intell. Fuzzy Syst. 29 (2015), 1653–1669.

[77] X.S. Yang, S. Deb, Engineering optimisation by cuckoo search, Int. J. Math. Modell. Numer. Optim. 1 (2010), 330–343.
[78] J. Li, Y.-x. Li, S.-s. Tian, J.-l. Xia, An improved cuckoo search algorithm with self-adaptive knowledge learning, Neural Comput. Appl. 32 (2020), 11967–11997.

[79] A.H. Gandomi, S. Talatahari, X.-S. Yang, S. Deb, Design optimization of truss structures using cuckoo search algorithm, Struct. Design Tall. Special Build. 22 (2013), 1330–1349.

[80] A.M. Kamoona, J.C. Patra, A novel enhanced cuckoo search algorithm for contrast enhancement of gray scale images, Appl. Soft Comput. 85 (2019), 105749.

[81] B. Yang, J. Miao, Z. Fan, J. Long, X. Liu, Modified cuckoo search algorithm for the optimal placement of actuators problem, Appl. Soft Comput. 67 (2018), 48–60.

[82] A. Majumder, D. Laha, P.N. Suganthan, A hybrid cuckoo search algorithm in parallel batch processing machines with unequal job ready times, Comput. Ind. Eng. 124 (2018), 65–76.

[83] H.-s. Ma, S.-x. Li, S.-f. Li, Z.-n. Lv, J.-s. Wang, An improved dynamic self-adaption cuckoo search algorithm based on collaboration between subpopulations, Neural Comput. Appl. 31 (2018), 1375–1389.

[84] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06), Vienna, Austria, 2005, pp. 695–701.

[85] Y. Yu, S. Gao, Y. Wang, Z. Lei, J. Cheng, Y. Todo, A multiple diversity-driven brain storm optimization algorithm with adaptive parameters, IEEE Access. 7 (2019), 126871–126888.

[86] L. Huang, S. Ding, S. Yu, J. Wang, K. Lu, Chaos-enhanced cuckoo search optimization algorithms for global optimization, Appl. Math. Model. 40 (2016), 3860–3875.

[87] F. Wang, X.-s. He, Y. Wang, The cuckoo search algorithm based on gaussian disturbance, J. Xi’an Polytechnic Univ. 25 (2011), 566–569.

[88] F.L.L. Wang, X.-s. He, Y. Wang, Hybrid optimization algorithm of pso and cuckoo search, in 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), IEEE, Dengleng, China, 2011.

[89] R.M. Rizk-Allah, R.A. El-Sehiemy, S. Deb, G.-G. Wang, A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor, J. Supercomput. 73 (2017), 1235–1256.

[90] J.-H. Yi, L.-N. Xing, G.-G. Wang, J. Dong, A.V. Vasilakos, A.H. Alavi, L. Wang, Behavior of crossover operators in nsga-iii for large-scale optimization problems, Inf. Sci. 509 (2020), 470–487.

[91] D. Gao, G.-G. Wang, W. Pedrycz, Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism, IEEE Trans. Fuzzy Syst. 28 (2020), 3265–3275.

[92] W. Li, G.-G. Wang, A.H. Gandomi, A survey of learning-based intelligent optimization algorithms, Arch. Comput. Method E. (2021).

[93] Y. Feng, S. Deb, G.-G. Wang, A.H. Alavi, Monarch butterfly optimization: a comprehensive review, Expert Syst. Appl. 168 (2021), 114418.

[94] Y. Feng, G.-G. Wang, Binary moth search algorithm for discounted {0-1} knapsack problem, IEEE Access. 6 (2018), 10708–10719.

[95] K. Srikanth, L.K. Panwar, B.K. Panigrahi, E. Herrera-Viedma, A.K. Sangaiah, G.-G. Wang, Meta-heuristic framework: quantum inspired binary grey wolf optimizer for unit commitment problem, Comput. Electr. Eng. 70 (2018), 243–260.