The Journal of Supercomputing (2019) 75:2395–2422
https://doi.org/10.1007/s11227-018-2625-x

Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization

Mohammad Shehab[1] · Ahamad Tajudin Khader[1] · Makhlouf Laouchedi[2] · Osama Ahmad Alomari[1]

Published online: 6 October 2018

Abstract
The cuckoo search algorithm (CSA) is a promising metaheuristic algorithm for solving numerous problems in different fields. It adopts the Levy flight to guide the search process. Nonetheless, CSA has drawbacks, such as the utilization of global search; in certain cases, this technique may surround local optima. Moreover, the results cannot be guaranteed if the step size is considerably large, thereby leading to a slow convergence rate. In this study, we introduce a new method for improving the search capability of CSA by combining it with the bat algorithm (BA) to solve numerical optimization problems. The proposed algorithm, called CSBA, begins by establishing the population of host nests in standard CSA and then obtains a solution through particular part to identify a new solution in BA (i.e., further exploitation). Therefore, CSBA overcomes the slow convergence of the standard CSA and avoids being trapped in local optima. The performance of CSBA is validated by applying it on a set of benchmark functions that are divided into unimodal and multimodal functions. Results indicate that CSBA performs better than the standard CSA and existing methods in the literature, particularly in terms of local search functions.

Keywords Nature-inspired algorithms · Cuckoo search algorithm · Levy flight · Bat algorithm · Slow convergence · Local optima

1 Introduction

Optimization exists in various domains, such as computer science, engineering, economics, medicine, and energy. It is primarily concerned with finding the optimal

✉ Mohammad Shehab
moh.shehab12@gmail.com

[1] School of Computer Science, Universiti Sains Malaysia (USM), Gelugor, Pulau Pinang, Malaysia

[2] ParIMd, LRPE, USTHB: Universit des Sciences et de Technologies Houari Boumediene, Bab Ezzouar, Algeria

values for several decision variables to develop a solution for an optimization problem [14]. This solution is optimally considered when it satisfies the decision maker [25]. An optimization problem involves the minimization or maximization of a suitable decision-making algorithm that is typically adapted to approximation methods. The principle of decision making entails choosing among several alternatives [29]. The result of this choice is the selection of the best decision from all available choices.

Optimization algorithms are developed on the basis of nature-inspired ideas that deal with selecting the best alternative by considering a given objective function [13]. An optimization algorithm can either be a heuristic or a metaheuristic approach. Heuristic approaches are problem-designed approaches in which each optimization problem has its particular heuristic methods that are not applicable to other types of optimization problem [30]. A metaheuristic-based algorithm is also a general solver template that can be adopted for different types of optimization problem by appropriately modifying its operators and configuring its parameters. In particular, each optimization algorithm can be categorized into three classes: evolutionary algorithms (EAs), swarm-based algorithms, and trajectory-based algorithms. EAs mimic the evolutionary principle of survival of the fittest. It typically begins with a set of individuals (i.e., solutions) called a population. In each generation, EA algorithms recombine the desirable characteristics of the current population to derive a new population that is selected on the basis of the natural selection principle. Examples of EAs include genetic algorithms (GAs) [12], genetic programming [19], differential evolution [35], and the harmony search algorithm (HS) [9]. Swarm-based algorithms mimic the behavior of a group of organisms as they strive to survive. At each iteration, the solutions are typically constructed on the basis of historical information obtained from the previous generation [4]. Examples of swarm-based algorithms are artificial bee colony [17], the firefly algorithm [40], the cuckoo search algorithm (CSA) [41], and the bat algorithm (BA) [39]. Trajectory-based algorithms begin with a single provisional solution. At each iteration, this solution moves to its neighboring solution, which is located in the same search space region, using a specific neighborhood structure. Examples of trajectory-based algorithms include tabu search (TS) [10], simulated annealing (SA) [18], and hill climbing [20].

This study focuses on CSA, which was developed by Yang and Deb (2009) to emulate the brood parasitism behavior among cuckoo birds. The aggressive breeding behavior of cuckoos has inspired this optimization algorithm. Brood parasitism is a primary survival mechanism of cuckoos. Cuckoos lay eggs in host nests; these eggs carefully mimic the pattern and color of host eggs [5]. In case the host recognizes the cuckoo eggs, it will either throw the mimetic eggs out of its nest or simply leave its nest and build a new one. Therefore, cuckoos must accurately mimic their hosts' eggs, whereas host birds must improve their skills in identifying parasitic eggs. This relationship illustrates the struggle for survival. In the context of optimization, each egg in the nest represents a solution, and the cuckoo egg represents a new solution. The objective of this study is to offer new and potentially better solutions to replace the mediocre ones in a nest. CSA can be extended to complicated cases in which each nest has multiple eggs that represent a set of solutions.

CSA is a promising optimization technique and has been successfully developed to solve global optimization problems [33]. Nevertheless, it also has drawbacks, such as solutions being poorly exploited at certain times, e.g., in cases when large steps generate a new solution that is either too far from the last solution or out of the boundary. By contrast, the effect is imperceptible when the step is too small; moreover, CSA suffers from a slow convergence rate [26]. In addition, no information is shared among solutions in CSA [24].

Wang et al. [37] proposed a hybrid algorithm that combines CSA and HS (HS/CSA) to address continuous optimization problems. In HS/CSA, the pitch adjustment of HS is used to update the process of CSA, thereby increasing population diversity. However, HS exhibits inherent drawbacks, including weak local search ability [14]. Therefore, HS/CSA still suffers when conducting local search.

Kanagaraj et al. [16] proved the efficiency of hybrid CSA and GA (HCSAGA) in solving engineering design optimization problems. HCSAGA began by randomly generating an initial solution, such that it is distributed throughout the solution space. Then, the authors applied selection, crossover, and mutation to each generation. Thereafter, the best solutions were selected by applying a certain form of elitism. Subsequently, the Levy flight operation for the best solution space was performed. Finally, the solutions in the current population were replaced with better solutions based on genetic principles. On the basis of the previous steps in identifying the best solution, HCSAGA requires a long computation time, which may lead to undesirable real problems, such as those in the clinical field.

A new hybrid algorithm that is composed of CSA and particle swarm optimization (PSO) was introduced by Wang et al. [36]. These authors claimed that the search area of PSO can be extended by hybridizing CSA and PSO, thereby preventing the shortcoming of PSO of falling easily into the extremum point.

In the current study, a hybrid algorithm that combines the optimization capabilities of CSA and BA, called CSBA, is proposed. This hybrid algorithm is introduced to overcome the limitations of CSA by exploiting the abilities of BA. BA directly utilizes the most suitable global solution in a population to identify new positions for particles at each iteration (i.e., sharing information among solutions). In addition, CSBA focuses on exploitation, thereby improving convergence (i.e., avoiding slow convergence). The proposed CSBA is expected to exhibit fast convergence, high computational precision, and improved effectiveness in searching for the optimal objective function value. The performance of CSBA is judiciously evaluated on 13 benchmark functions selected from the literature. The results illustrate the effectiveness of the proposed algorithm compared with other methods on most of the benchmark cases.

The rest of this paper is organized as follows. Section 2 reviews the basic CSA and the basic BA. The CSBA approach is presented in Sect. 3. In Sect. 4, our method is evaluated by using benchmark problems and comparing the results with eight other methods. The last part concludes the paper and points out directions for future work.

2 Preliminaries

2.1 Cuckoo search algorithm

The use of CSA in the optimization context was proposed by Yang and Deb in 2009 [41]. To date, work on this algorithm has significantly increased, and the CSA has succeeded in having its rightful place among other optimization methodologies [43]. This algorithm is based on the obligate brood parasitic behavior found in some cuckoo species, in combination with the Levy flight behavior discovered in some birds and fruit flies. The CSA is an efficient metaheuristic swarm-based algorithm that efficiently strikes a balance between local nearby exploitation and global-wide exploration in the search space problem [31].

The cuckoo has a specific way of laying its eggs to distinguish it from the rest of the birds [42]. The following three idealized rules clarify and describe the standard cuckoo search:

- Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs will be carried over to the next generations.
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability$P\alpha \in (0, 1)$. In this case, the host bird can either get rid of the egg or simply abandon the nest and build a completely new nest. In addition, probability$P\alpha$ can be used by the n host nest to replace the new nests.

CSA is similar to other swarm-based algorithms, and the CSA starts with an initial population of n host nests. These initial host nests will be randomly attracted by the cuckoos with eggs and also by random Levy flights to lay the eggs. Thereafter, nest quality will be evaluated and compared with another random host nest. In case the host nest is better, it will replace the old host nests. This new solution has the egg laid by a cuckoo. If the host bird discovers the egg with a probability P α ∈ (0, 1), the host either throws out the egg or abandons it and builds a new nest. This step is done by replacing the abundant solutions with the new random solutions [32].

Yang and Deb used a certain and simple representation for the implementation, with each egg representing a solution. As the cuckoo lays only one egg, it also represents one solution. The purpose is to increase the diversity of new, and probably better, cuckoos (solutions) and replace them instead with the worst solutions. By contrast, the CSA can be more complicated by using multiple eggs in each nest to represent a set of solutions.

The CSA, as a bat algorithm [39] and an FA [38], uses a balance between exploration and exploitation. The CSA is equiponderant to the integration of a local random walk. The local random walk can be written as

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(P\alpha - \varepsilon) \otimes (x_j^t - x_k^t) \tag{1}$$

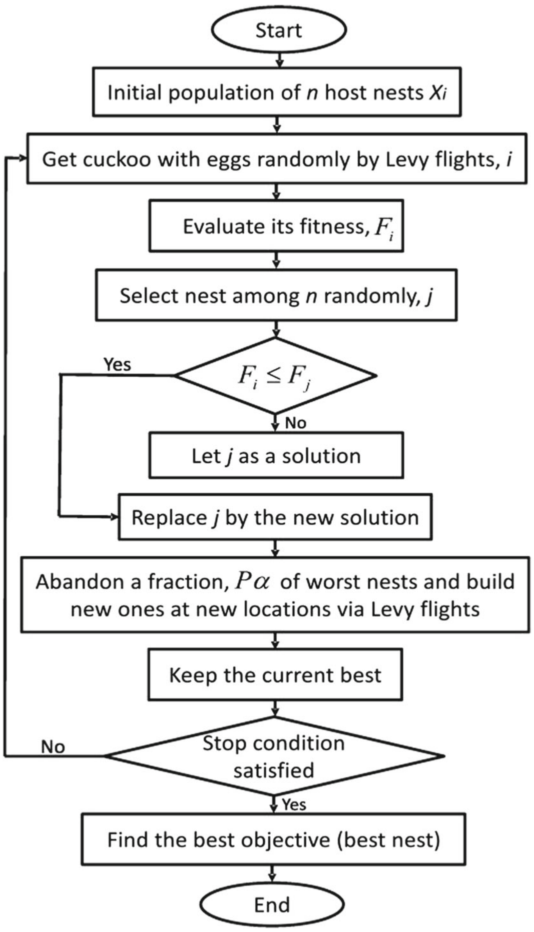
where x_j^t and x_k^t are two different solutions selected randomly by random permutation,$H(u)$ is a Heaviside function and generates a random number drawn from a uniform distribution between 0 and 1, and s is the step size. Furthermore, global explorative random walk by using Levy flights can be expressed as follows:

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda), \tag{2}$$

And

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\lambda \pi / 2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad s >> s_0 > 0 \tag{3}$$

Fig. 1 Flowchart of CSA


Start
Initial population of n host nests X_i
Get cuckoo with eggs randomly by Levy flights, i
Evaluate its fitness, F_i
Select nest among n randomly, j
$F_i \leq F_j$
Yes
No
Let j as a solution
Replace j by the new solution
Abandon a fraction, $P\alpha$ of worst nests and build new ones at new locations via Levy flights
Keep the current best
Stop condition satisfied
No
Yes
Find the best objective (best nest)
End

where L is the characteristic scale of the problem of interest, while $\alpha > 0$ is a factor of size scaling. The value can be calculated by $\alpha = O(L/10)$; for more affectivity, $\alpha = O(L/100)$ can be used. In addition, x^t in the above equation represents the current location, which is the only way to determine the next location x^{t+1}. This is called the random walk and the Markov chain. In the second term, $\alpha L(s, \lambda)$ represents the transition probability. However, to avoid premature convergence and increase diversity (not only confined in a local optimum), the new solutions should be generated on a remote sufficient distance from the current best solution and some random elements should be included. Figure 1 shows the representation of the CSA search process.

2.2 Bat algorithm

Bat-inspired algorithm (BA), introduced by Xin-She Yang in 2010 [39], emulates the echolocation behavior of bats. There are many kinds of bats in nature. All of them have similar behavior when navigating and hunting; however, they are different in terms of size and weight. Microbats extensively used echolocation feature, which assists them in seeking for prey and/or avoids obstacles in a complete darkness. The behavior of microbats can be formulated by the novel optimization technique.

In the BA, the artificial bat has position vector, velocity vector, and frequency vector which are updated during the course of iterations. The BA can explore the search space through position and velocity vectors (or updated position vectors).

Each bat has a position X_i, frequency F_i, and velocity V_i in a d-dimensional search space. The velocity, position, and frequency vectors are updated in Eqs. 4, 5, and 6.

$$V_i(t+1) = V_i(t) + (X_i(t) - Gbest) \times F_i \tag{4}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{5}$$

where Gbest is the best solution obtained so far and F_i represents the frequency of the ith bat which is updated in each course of iteration as follows:

$$F_i = F_{min} + (F_{max} - F_{min}) \times \beta \tag{6}$$

where β is a random number of uniform distribution in [0,1]. The BA employed a random walk to improve its capability in exploitation as given below.

$$x_{new} = x_{old} + \varepsilon A_t \tag{7}$$

where ε is a random number in [-1,1], and A is the loudness of emitted sound. At each iteration, the loudness and pulse emission (r) are updated as follows:

$$A_i(t+1) = \alpha A_i(t) \tag{8}$$

$$r_i(t+1) = r_i(0)(1 - e^{(-\gamma \times t)}) \tag{9}$$

where α and γ are constant parameters that lies between 0 and 1 and used to update loudness rate A_i and pulse rate (r_i). The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1 Basic bat-inspired algorithm

```
1: Initialize the bat population $X_i(i = 1, 2, .., n)$ and $V_i$
2: Define pulse frequency $F_i$
3: Initialize pulse rate $r_i$ and the loudness $A_i$
4: while $t < Maxnumberofiterations$ do
5:    Generate new solutions by adjusting frequency,
6:    Updating velocities and positions [equations (4) to (4)]
7:    if $rand > r_i$ then
8:       Select a solution among the best solutions randomly
9:       Generate a local solution around the selected best solution
10:   end if
11:   Generate a new solution by flying randomly
12:   if $rand < A_i$ and $f(X_i) < f(x^*)$ then
13:      Accept the new solutions
14:      Increase $r_i$ and reduce $A_i$
15:   end if
16:   Rank the bats and find the current Gbest
17: end while
```

3 CSBA

This section provides the details of merging the components of CSA with those of BA. The combined algorithm avoids solutions with poor fitness to increase the quality of solutions and focuses on exploitation to improve slow convergence, thereby improving search efficiency. The following paragraphs describe the details of CSBA.

In general, Levy flight is applied to optimization and optimal search; its efficiency has been proven by achieving favorable results that signify a promising beginning [27]. Therefore, CSA is balanced between exploration and exploitation [22, 23]. However, solutions are poorly exploited at certain times, such as in cases when large steps generate a new solution that is either too far from the last solution or out of the boundary. By contrast, the effect is unnoticeable when the step is too small. CSBA is proposed to overcome this drawback of CSA by utilizing the advantages of BA; BA can provide fast convergence in the initial stage by switching from exploration to exploitation [1, 3, 44]. Thus, the advantages of the CSBA are increasing the quality of the solutions, enhancing performance, and avoiding being trapped in local optima.

Figure 2 illustrates the CSBA flowchart, which is divided into three parts. The first part shows the initialization and comparison between the solutions of Levy flight and tournament selection to send it to the second part. The second part is surrounded by a red stripe, which represents the BA component. A new solution is generated in this part on the basis of solution *i* from the first part. Furthermore, pulse frequency F_i, loudness A_i, and pulse rate r_i are defined, as mentioned in Eqs. 6, 8, and 9, respectively. Velocities and positions (Eqs. 4 and 5) are updated, thereby allowing the search for all possible solutions around the best solution (i.e., increase in exploitation). A solution is then randomly selected among the best solutions, and a local solution is generated close to the selected best solution after checking r_i (i.e., intensive local search). The new solution is selected by comparing pulse frequency with the current solution. The second part provides the best solution with a high likelihood to exceed the last condition in the third part, which is called probability Pa (Sect. 2.1). Thus, the best solution of CSBA should be efficient. The first and third parts belong to CSA, whereas the second part belongs to BA.

4 Simulations

The parameter settings for the experiments are mentioned in Sect. 4.1. CSBA is compared with five optimization algorithms, i.e., the krill herd algorithm (KH), HS, GA, BA, and CSA, in Sect. 4.2 by using a set of global numerical optimization problems through various experiments performed on benchmark functions. These functions are implemented under the same conditions adopted in Ref. [37]. In the present work, the 13 benchmark functions are used to analyze CSBA. The following points show the information of all benchmark functions. The functions are classified into two groups: (1) unimodal optimization functions with only one local optimum and (2) multimodal optimization functions that frequently contain multiple global and local optima.

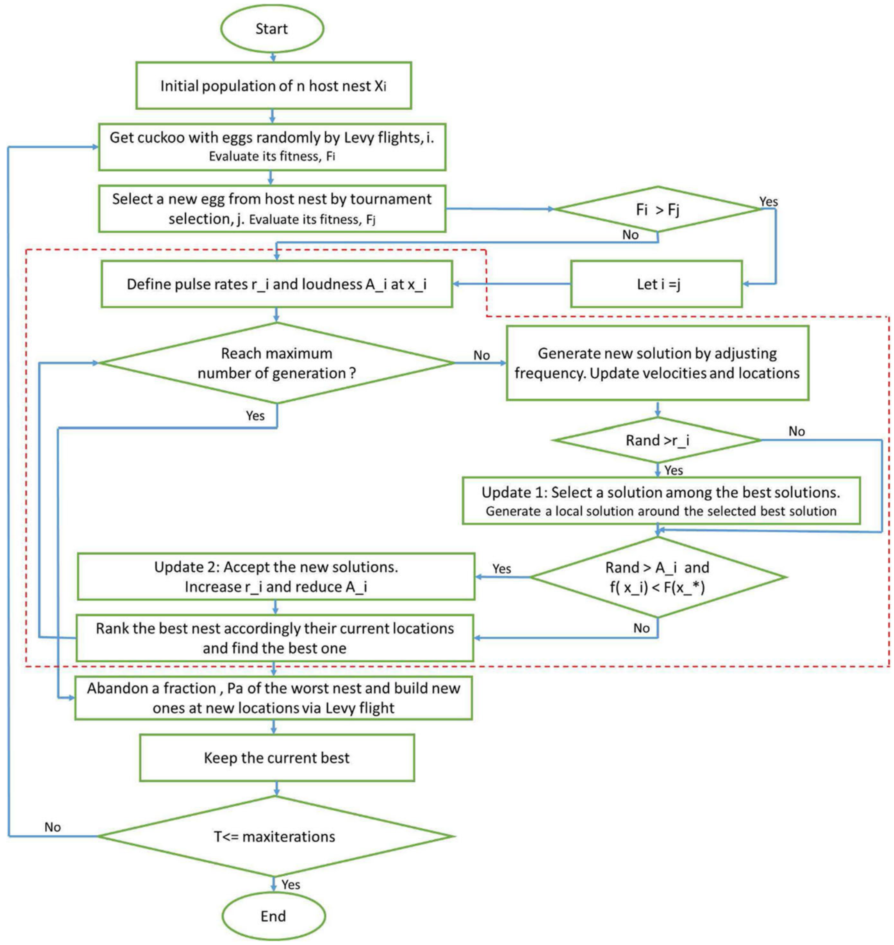

Start
Initial population of n host nest Xi
Get cuckoo with eggs randomly by Levy flights, i. Evaluate its fitness, Fi
Select a new egg from host nest by tournament selection, j. Evaluate its fitness, Fj
Fi > Fj
Yes
No
Let i =j
Define pulse rates r_i and loudness A_i at x_i
Reach maximum number of generation ?
No
Yes
Generate new solution by adjusting frequency. Update velocities and locations
Rand >r_i
No
Yes
Update 1: Select a solution among the best solutions. Generate a local solution around the selected best solution
Rand > A_i and f(x_i) < F(x_*)
Yes
No
Update 2: Accept the new solutions. Increase r_i and reduce A_i
Rank the best nest accordingly their current locations and find the best one
Abandon a fraction , Pa of the worst nest and build new ones at new locations via Levy flight
Keep the current best
T<= maxiterations
No
Yes
End

Fig. 2 Flowchart of CSBA

1. *F1: Ackley*

$$f(x) = 20 + e - 20.e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$

Ackley is a multimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-32.768, 32.768]$, with the global minimum at $\boldsymbol{x}_i = 0$ [3].

2. *F2: Griewank*

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Griewank is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-600, 600]$, with the global minimum at $\boldsymbol{x}_i = 0$ [11].

3. *F3: Generalized Penalized #1*

$$f(x) = \frac{\pi}{30}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\}$$
$$\sum_{i=1}^{n} u(x_i, 10, 100, 4)$$

Generalized Penalized #1 is a multimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-50, 50]$, with the global minimum at $\boldsymbol{x}_i$=0 [45], where $y_i = 1 + 0.25(x_i + 1)$, $u(x_i, a, k, l) = k(x_i - a)^l$

4. *F4: Generalized Penalized #2*

$$f(x) = 0.1\left\{\sin^2(3\pi x_1\right\} + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot \left[1 + \sin^2(3\pi x_{i+1})\right]$$
$$+ (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$$

Generalized Penalized #2 is a multimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-50, 50]$, with the global minimum at $\boldsymbol{x}_i$=0 [45], where $u(x_i, a, k, l) = k(x_i - a)^l$

5. *F5: Quartic with noise*

$$f(x)\sum_{i=1}^{n}\left(i.x_i^4 + U(0, 1)\right)$$

Quartic with noise is a multimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-1.28, 1.28]$, with the global minimum at $\boldsymbol{x}_i = 0$ [34].

6. *F6: Rastrigin*

$$f(x) = 10.n\sum_{i=1}^{n}\left(x_i^2 - 10.\cos(2\pi x_i)\right)$$

Rastrigin is a multimodal function, usually evaluated on the hypercube $\boldsymbol{xi} \in [-5.12, 5.12]$, with the global minimum at $\boldsymbol{x}_i = 0$ [6].

7. *F7: Rosenbrock*

$$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$$

Rosenbrock is a multimodal function, usually evaluated on the hypercube $\boldsymbol{xi} \in [-5, 10]$, with the global minimum at $\boldsymbol{x}_i$=0 [15].

8. *F8: Schwefel 2.26*

$$f(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin\left(|x_i|^{1/2}\right)$$

Schwefel 2.26 is a multimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-500, 500]$, with the global minimum at $\boldsymbol{x}_i$=420.9687 [21].

9. *F9: Schwefel 1.2*

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

Schwefel 1.2 is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-100, 100]$, with the global minimum at $\boldsymbol{xi} = 0$ [15].

10. *F10: Schwefel 2.22*

$$f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$

Schwefel 2.22 is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-100, 100]$, with the global minimum at $\boldsymbol{x}_i = 0$ [15].

11. *F11: Schwefel 2.21*

$$f(x) = \max_i |x_i|, \quad 1 < i < n$$

Schwefel 2.21 is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-100, 100]$, with the global minimum at $\boldsymbol{x}_i$=0 [28].

12. *F12: Sphere*

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Sphere is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-5.12, 5.12]$, with the global minimum at $\boldsymbol{x}_i$=0 [8].

13. *F13: Step*

$$f(x) = 6.n + \sum_{i=1}^{n} \lfloor x_i \rfloor$$

Step is a unimodal function, usually evaluated on the hypercube $\boldsymbol{x}_i \in [-5.12, 5.12]$, with the global minimum at $\boldsymbol{x}_i = 0$ [7].

All the experiments are run using a computer with processor Intel(R) Core (TM) i5-3210M CPU @ 2.66 GHz with 4 GB of RAM and 32 bit for operating system with Microsoft Windows 10 professional. The source code is implemented using the MATLAB (R2015a).

It is worth to mention that all the experimental results (Tables 1–10) show the best normalization (i.e., the best value for each method is determined and then normalized) and the mean normalization (i.e., the average value for each method is determined and then normalized) of each benchmark function for each algorithm. That is, normalization is the process of regularizing data with respect to the difference in values between samples. In the experiments, different benchmark functions are compared with one another. This procedure is difficult due to the wide gap between solutions. Therefore, normalization improves data integrity [9]. In this article, normalization is calculated based on the following equation:

$$z_i = \frac{x_i - \mu}{S} \tag{10}$$

where is $x = (x1, \ldots, x_n)$, n denotes the total number of data, z_i denotes the normalized data for element ith, μ is the mean, and S is the standard deviation. Finally, the minimum element of the data will be **1** in the normalization results.

4.1 Influence of control parameter

Parameter setting plays an important role in the performance of meta-heuristic methods when solving different problems. In this work, the parameters of BA (i.e., loudness (A), pulse rate (r)) and the parameters of CSA (i.e., discovery rate ($P\alpha$), population size (n)) are thoroughly studied with 100 implementations with 10,000 run times performed for each algorithm on each benchmark function to decrease the influence of randomness. The results show the best value of each parameters (see Tables 3–8) which are used later in Sect. 4.2.

4.1.1 Population size:*n*

The parameters setting is used in this section as follows: There are different values of$n$ (5, 10, 15, 20, 50, 100, 150, 250, and 500) with the fixed value for each of ($P\alpha$ = 0.25, A and r = 0.5). Tables 1 and 2 illustrate the results of CSBA with different values of n.

As shown in Table 1, all the results are close to one another except when $n = 20$ (i.e., by looking at the last row “Total,” the numbers are nearly similar except when n = 20; the number is 10). The results of the best normalization when $n = 20$ outperform the others in all benchmark functions except F5, F7, and F11. Furthermore, the results of the mean normalization in Table 2 refer to the convergence of the results for n = 20 by achieving 11 best results out of 13. Therefore, the value of n will be set to 20 in the experiments later.

Table 1 Best normalized optimization results for CSBA with different *n*

n									
	5	10	15	20	50	100	150	250	500
F1	5.41E+01	**1.00**	1.17E+02	**1.00**	1.49E+01	9.91E+02	5.94E+02	1.93E+02	3.79E+02
F2	5.23E+01	7.13E+00	**1.00**	**1.00**	2.56E+00	**1.00**	2.60E+01	5.16E+01	1.70E+01
F3	1.04E+00	**1.00**	1.04E+00	**1.00**	1.17E+00	1.04E+00	1.04E+00	1.04E+00	1.77E+00
F4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F5	2.41E+01	8.37E+00	4.45E+01	1.93E+00	2.19E+01	3.70E+00	1.12E+01	4.90E+01	**1.00**
F6	1.74E+01	6.89E+00	2.39E+01	**1.00**	2.36E+01	6.65E+01	2.16E+01	3.22E+01	1.82E+01
F7	1.23E+00	1.04E+00	1.09E+00	1.90E+00	**1.00**	1.04E+00	1.24E+00	1.14E+00	2.24E+00
F8	1.01E+01	1.20E+01	1.09E+00	**1.00**	2.68E+01	3.80E+00	5.96E+00	1.62E+01	3.41E+01
F9	1.04E+00	2.57E+00	2.27E+00	**1.00**	5.68E+00	1.24E+00	1.10E+00	2.55E+00	2.25E+00
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	1.33E+01	1.05E+01	1.86E+00	1.86E+00	6.39E+00	2.87E+01	**1.00**	2.58E+01	2.87E+01
F12	2.28E+00	7.58E+00	2.85E+00	**1.00**	2.53E+00	1.05E+00	5.00E+00	**1.00**	8.54E+00
F13	**1.00**	1.32E+00	8.20E+00	**1.00**	**1.00**	**1.00**	**1.00**	1.29E+00	2.56E+00
Total	3	4	3	10	4	4	4	3	3

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 2 Mean normalized optimization results for CSBA with different *n*

n									
	5	10	15	20	50	100	150	250	500
F1	2.83E+01	2.49E+01	1.83E+00	**1.00**	1.66E+01	9.32E+01	1.08E+02	1.09E+02	1.98E+02
F2	2.23E+01	3.87E+01	1.02E+01	**1.00**	1.49E+00	1.28E+00	1.10E+01	1.12E+01	1.60E+01
F3	1.58E+00	1.53E+00	1.23E+00	**1.00**	1.15E+00	1.16E+00	1.20E+00	1.17E+00	1.17E+00
F4	2.23E+00	1.60E+00	1.42E+00	**1.00**	1.11E+00	1.02E+00	**1.00**	1.18E+00	1.24E+00
F5	2.69E+00	2.32E+00	2.54E+00	**1.00**	1.93E+00	1.81E+00	1.87E+00	1.66E+00	1.40E+00
F6	1.02E+02	7.62E+01	**1.00**	**1.00**	7.62E+00	1.52E+01	2.62E+01	2.26E+01	2.71E+00
F7	1.01E+00	1.04E+00	**1.00**	1.06E+00	1.02E+00	1.01E+00	1.02E+00	1.02E+00	1.07E+00
F8	1.14E+00	1.29E+00	1.28E+00	**1.00**	**1.00**	1.29E+00	1.24E+00	1.21E+00	1.38E+00
F9	1.42E+02	1.15E+01	1.16E+01	**1.00**	3.43E+00	2.79E+00	2.01E+00	1.24E+01	1.55E+01
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	1.32E+00	1.32E+00	**1.00**	1.27+00	1.35E+00	1.43 E+00	1.22E+00	1.21E+00	1.21E+00
F12	1.11E+00	7.58E+00	1.25E+00	**1.00**	1.19E+00	**1.00**	1.10E+00	1.02E+00	1.33E+00
F13	**1.00**	1.02E+00	1.03E+00	**1.00**	**1.00**	1.02E+00	**1.00**	1.04E+00	1.01E+00
Total	2	1	4	11	3	2	3	1	1

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

4.1.2 Discovery rate: $P\alpha$

The effect of the elitism parameter is studied in the benchmark problems with the elitism parameter $P\alpha = 0$–1 (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0), both A and $r = 0.5$, and $n = 20$ (see Tables 3 and 4).

Table 3 shows that CSBA performs best when $P\alpha = 0.1$, 0.2, and 0.4; CSBA has a similar performance, especially for F3, F10, and F13; that is, the elitism parameter $P\alpha$ has little influence on the three benchmark functions. Furthermore, when $P\alpha = 0$, 0.3, and 0.5–0.9, the CSBA performance achieves almost the same results. However, the worst results are obtained when $P\alpha = 1$. Therefore, CSBA has the best performance when $P\alpha = 0.2$. On the basis of these results, $P\alpha$ is set to 0.2 in the present study.

4.1.3 Loudness: A

The influence of A is investigated through an array of simulations with A = 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0. n, Pα, and r are equal to 20, 0.2, and 0.5, respectively (see Tables 5 and 6). From Table 5, A = 0.7 for F1, F3 to F5, F7, F10, F12, and F13. By contrast, the CSBA performs the worst when A = 0 and 0.1, especially for F1, F9, F10, and F13. Therefore, the value of A will be set to 0.7 in the last experiment for r.

4.1.4 Pulse rate: r

As mentioned in previous paragraphs, the values of n, $P\alpha$, and A are set to 20, 0.2, and 0.7, respectively. The influence of pulse rate r is studied in the benchmark functions with r=0, 0.1, 0.2, 0.9, 1.0 (see Tables 7 and 8).

Table 7 shows that three sets of results were obtained. Beginning with r=0.4, which obtained the best CSBA performance for F4, F6–F8, F10, and F12–F13, followed by the second set, which includes r=0.1–0.2, 0.5–0.6, and 0.8–1.0, all the r values have the same amount of the best value total. The last set contains the remaining benchmark functions that obtained the worst results. Therefore, CSBA has the best performance when r is equal or close to 0.4. Based on the results, r is set to 0.4.

4.2 Comparisons with other methods

CSBA is initially compared with the global optimization problems of five optimization algorithms, namely KH, HS, GA, BA, and CSA. In our simulations, similar parameters for CSA, BA, and CSBA (as shown above) are set with population size $n = 20$, discovery rate $P\alpha = 0.2$, loudness $A = 0.7$, and pulse rate $r = 0.4$. The parameters used for KH, HS, and GA are the same as those used for each original algorithm. The other parameters, i.e., frequency minimum, frequency maximum, function dimension, and maximum generation, are set to 0, 2, 20, and 100,000, respectively. A total of 100 implementations are performed for each algorithm on each benchmark function to decrease the influence of randomness. Tables 9 and 10 show the different scales used to normalize the values for illustrating the differences of the six methods.

Table 3 Best normalized optimization results for CSBA with different $P\ \alpha$

$P\alpha$											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	6.89E+00	1.55E+01	**1.00**	1.73E+01	3.01E+01	1.88E+01	3.35E+00	5.18E+00	1.16E+01	2.82E+01	1.87E+01
F2	4.28E+01	**1.00**	8.58E+00	3.38E+01	1.42E+01	1.64E+01	8.59E+00	4.22E+01	6.91E+00	1.23E+01	5.94E+01
F3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.17E+00
F4	1.15E+00	1.02E+00	1.11E+00	1.08E+00	1.17E+00	1.06E+00	**1.00**	**1.00**	1.23E+00	1.14E+00	1.22E+0
F5	1.06E+00	4.23E+00	1.22E+01	3.31E+01	5.11E+00	7.01E+00	5.17E+00	5.39E+00	1.82E+01	**1.00**	1.81E+01
F6	6.36E+00	2.54E+00	**1.00**	6.99E+00	1.13E+01	2.55E+00	7.63E+00	5.63E+00	1.40E+00	1.94E+00	7.95E+0
F7	1.06E+02	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.05E+01	**1.00**
F8	5.84E+02	3.46E+00	2.18E+00	2.18E+00	5.15E+00	1.10E+01	6.24E+00	3.67E+01	1.26E+01	**1.00**	1.90E+01
F9	5.06E+02	4.85E+02	1.95E+02	1.95E+02	**1.00**	9.53E+02	6.84E+02	3.28E+02	1.16E+02	2.48E+01	2.55E+02
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.21E+00	**1.00**	**1.00**	**1.00**	1.01E+00	2.12E+01
F11	2.88E+01	**1.00**	3.97E+01	1.27+01	1.46E+01	2.93E+01	1.08E+01	4.47E+00	1.21E+01	6.82E+01	2.93E+01
F12	4.61E+01	6.03E+00	**1.00**	2.02E+00	6.82E+00	2.54E+00	1.13E+01	4.86E+00	7.52E+00	1.12E+00	1.48E+01
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	2.05E+01	1.22E+02	2.05E+02	1.02E+03	1.21E+03
Total	3	5	6	3	4	2	3	3	2	3	1

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 4 Mean normalized optimization results for CSBA with different $P\alpha$

$P\alpha$											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	1.20	1.26	1.03	**1.00**	1.14	1.02	1.35	1.38	1.32	1.26	1.55
F2	1.29E+00	**1.00**	1.32E+00	4.54E+07	5.25E+07	1.57E+08	8.78E+07	4.42E+08	9.21E+08	1.01E+09	5.18E+06
F3	**1.00**	1.10E+00	1.02E+00	1.07E+00	1.05E+00	1.14E+00	1.13E+00	1.25E+00	1.24E+00	1.16E+00	**1.00**
F4	1.11E+00	**1.00**	1.11E+00	1.32E+00	1.24E+00	1.20E+00	1.52E+00	1.38E+00	1.52E+00	1.78E+00	1.16E+00
F5	**1.00**	1.06	1.04	1.24	1.03	1.27	1.10	1.09	1.13	1.34	1.25
F6	1.10E+00	1.38E+07	1.38E+07	1.38E+07	**1.00**	1.22E+00	1.38E+07	5.53E+07	5.53E+07	2.76E+07	3.88E+07
F7	2.33	2.31	**1.00**	2.37	2.39	2.32	2.33	2.32	2.34	2.31	2.38
F8	4.44E+51	1.25E+00	1.04E+00	**1.00**	1.05E+00	1.19E+00	1.21E+00	1.28E+00	1.29E+00	1.25E+00	1.40E+161
F9	8.13E+00	**1.00**	1.48E+03	9.72E+04	2.40E+05	4.28E+05	3.77E+05	4.69E+05	4.93E+05	4.98E+05	5.48E+05
F10	1.02E+00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.02E+00	6.65E+06
F11	1.22	1.16	**1.00**	1.18	1.23	1.01	1.02	1.13	1.25	1.07	1.04
F12	1.47E+50	1.22E+00	1.16E+00	1.12E+00	**1.00**	1.14E+00	1.25E+00	1.37E+00	1.25E+00	1.19E+00	6.42E+160
F13	**1.00**	1.01E+00	**1.00**	**1.00**	1.04E+00	1.01E+00	4.49E+15	4.49E+15	4.49E+15	4.49E+15	4.35E+15
Total	3	4	4	4	3	1	1	1	1	0	1

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 5 Best normalized optimization results for CSBA with different loudness

A											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F2	1.71E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	1.80E − 01	**1.00**
F3	1.17E+00	1.17E+00	**1.00**	1.17E+00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F4	3.52E+03	3.01E+00	**1.00**	**1.00**	**1.00**	1.08E+00	1.41E+00	**1.00**	1.32E+00	1.09E+00	1.15E+00
F5	2.49E+01	2.74E+00	1.96E+00	5.31E+01	1.01E+01	3.90E+01	6.83E+00	**1.00**	1.97E+00	1.05E+01	1.61E+01
F6	2.59E+00	4.92E+01	3.08E+01	4.71E+00	5.80E+01	5.39E+01	1.65E+00	1.51E+02	1.38E+02	**1.00**	9.02E+01
F7	1.67	1.01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F8	4.00E+00	1.64E+01	4.97E+00	3.06E+00	1.37E+01	**1.00**	2.09E+00	4.74E+00	4.92E+01	7.19E+00	3.66E+00
F9	**1.00**	**1.00**	2.51E+04	1.74E+02	4.42E+01	6.99E+02	9.67E+05	9.09E+11	4.91E+16	1.97E+16	1.78E+24
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	38.30	5.73	12.59	5.73	18.02	22.80	4.05	2.98	10.66	12.60	**1.00**
F12	8.63E+01	4.22E+01	7.58E+00	3.94E+01	1.31E+01	5.24E+01	1.07E+01	**1.00**	1.02E+01	6.52E+01	7.76E+00
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Total	4	4	6	5	6	6	5	8	5	6	7

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 6 Mean normalized optimization results for CSBA with different loudness

A											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F2	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	**1.00**
F3	1.16E+00	**1.00**	1.12E+00	1.23E+00	1.14E+00	1.16E+00	1.08E+00	1.21E+00	1.15E+00	1.21E+00	1.20E+00
F4	1.74E+00	1.43E+00	1.34E+00	1.23E+00	**1.00**	1.04E+00	1.12E+00	1.17E+00	1.14E+00	1.08E+00	1.13E+00
F5	**1.00**	1.24E+00	1.34E+00	1.51E+00	1.73E+00	1.70E+00	1.74E+00	1.55E+00	1.57E+00	1.69E+00	1.89E+00
F6	**1.00**	3.83E+07	2.69E+00	3.83E+07	1.53E+08	7.65E+07	3.51E+00	7.65E+07	7.65E+07	7.65E+07	3.19E+00
F7	1.13	1.40	1.45	1.33	**1.00**	1.60	1.27	1.53	1.43	**1.00**	1.87
F8	1.19E+00	1.24E+00	1.16E+00	1.13E+00	1.16E+00	1.08E+00	1.11E+00	**1.00**	1.20E+00	1.27E+00	1.23E+00
F9	1.02E+05	1.54E+04	3.07E+04	1.00E+05	4.99E+24	1.17E+04	1.99E+02	**1.00**	9.61E+24	2.50E+25	6.89E+25
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	1.08	1.06	1.01	1.07	1.14	1.06	1.05	1.26	1.07	**1.00**	1.08
F12	1.14E+00	1.20E+00	**1.00**	1.30E+00	1.23E+00	1.10E+00	1.24E+00	1.10E+00	1.02E+00	1.33E+00	1.15E+00
F13	1.02E+00	1.01E+00	1.02E+00	1.03E+00	1.01E+00	1.01E+00	**1.00**	1.01E+00	1.01E+00	1.02E+00	1.01E+00
Total	4	3	3	2	4	2	3	4	2	4	3

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 7 Best normalized optimization results for CSBA with different pulse rate

r											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	5.41E+06	1.32E+08	1.17E+08	1.49E+07	1.25E+06	9.91E+07	5.94E+07	1.93E+07	3.79E+07	7.28E+06	**1.00**
F2	5.23E+09	7.13E+00	8.20E+00	2.65E+00	1.68E+01	**1.00**	2.60E+01	5.16E+00	1.70E+00	1.36E+00	6.54E+00
F3	1.044	**1.00**	1.049	1.171	1.021	1.049	1.042	1.046	1.177	1.041	1.043
F4	1.23E+00	1.04E+00	**1.00**	1.09E+00	**1.00**	1.04E+00	1.24E+00	1.14E+00	2.24E+00	2.36E+00	3.53E+03
F5	2.41E+02	8.37E+00	4.45E+01	2.19E+01	2.93E+00	3.70E+00	1.12E+01	4.90E+01	2.56E+00	**1.00**	5.91E+00
F6	1.74E+01	6.89E+00	2.39E+01	2.36E+01	**1.00**	6.65E+01	2.16E+01	3.22E+01	1.82E+01	1.47E+00	3.44E+01
F7	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F8	1.01E+01	1.20E+01	1.09E+00	2.68E+01	**1.00**	3.80E+00	5.96E+00	1.62E+01	3.41E+01	6.85E+00	2.74E+01
F9	1.04E+29	2.57E+26	2.27E+22	1.24E+13	5.68E+14	1.10E+12	2.55E+00	2.25E+03	**1.00**	1.67E+04	1.85E+06
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	1.86	1.86	13.35	6.39	10.53	28.27	**1.00**	25.81	2.62	3.41	14.40
F12	3.28E+00	7.58E+00	2.85E+00	2.53E+00	**1.00**	1.05E+00	5.00E+00	1.29E+00	8.54E+00	5.03E+00	7.04E+86
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Total	3	4	4	3	7	4	4	3	4	4	4

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 8 Mean normalized optimization results for CSBA with different pulse rate

r											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	2.83E+12	2.49E+12	1.83E+12	1.63E+12	**1.00**	9.32E+11	8.98E+11	1.09E+12	3.98E+11	1.99E+11	1.66E+12
F2	3.23E+09	3.87E+07	1.02E+07	2.02E+00	**1.00**	1.28E+00	1.10E+09	1.12E+00	1.10E+00	1.30E+00	1.49E+00
F3	1.58E+00	1.53E+00	1.23E+00	1.17E+00	**1.00**	1.16E+00	1.20E+00	1.17E+00	1.17E+00	1.06E+00	1.15E+00
F4	2.23E+00	1.60E+00	1.42E+00	1.06E+00	1.11E+00	1.02E+00	**1.00**	1.18E+00	1.24E+00	1.36E+00	1.77E+00
F5	2.69	2.32	2.54	2.08	**1.00**	1.81	1.87	1.66	1.40	1.37	1.93
F6	1.52E+08	7.62E+07	3.81E+07	2.98E+00	**1.00**	1.52E+08	7.62E+07	7.62E+07	2.71E+00	3.81E+07	7.62E+07
F7	1.01	1.04	**1.00**	1.06	1.02	1.01	1.02	1.02	1.07	1.06	1.14
F8	1.14E+00	1.29E+00	1.28E+00	1.51E+00	1.35E+00	1.29E+00	1.24E+00	1.21E+00	1.38E+00	1.23E+00	**1.00**
F9	7.42E+24	5.65E+24	3.76E+24	3.43E+24	2.79E+23	2.01E+23	2.24E+02	1.35E+04	**1.00**	1.00E+10	7.74E+09
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	1.32	1.32	1.17	1.27	1.35	1.43	1.22	1.21	1.21	**1.00**	1.25
F12	1.11E+00	1.14E+00	1.25E+00	1.03E+00	1.19E+00	**1.00**	1.10E+00	1.02E+00	1.33E+00	1.29E+00	9.16E+159
F13	**1.00**	1.02E+00	1.03E+00	**1.00**	1.01E+00	1.02E+00	**1.00**	1.04E+00	1.01E+00	1.02E+00	**1.00**
Total	2	1	2	2	6	2	3	1	2	2	3

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 9 Mean normalized optimization results

	KH	HS	GA	BA	CSA	CSBA
F1	1,372,770,963	1.23781E+13	15,907,669,921	8.35598E+15	3.4902E+15	**1.00**
F2	3.42E+09	6.01E+05	2.38E+04	3.41E+09	1.22E+00	**1.00**
F3	1.97E+00	**1.00**	**1.00**	1.31E+00	**1.00**	1.14E+00
F4	2.02E+05	1.90E+05	**1.00**	2.11E+05	2.01E+05	1.20E+00
F5	68.99349967	2.245813663	5.239815617	4.743698018	2.090284576	**1.00**
F6	6.97E+09	6.52E+06	8.22E+03	8.40E+07	4.17E+07	**1.00**
F7	337.7393272	357.805229	42.2297262	211.2094139	358.1659325	**1.00**
F8	1.52E+159	1.35E+160	8.90E+157	3.80E+158	**1.00**	1.18E+00
F9	3.97E+65	1.79E+67	6.16E+65	1.72E+70	1.28E+46	**1.00**
F10	4.01E+00	2.73E+02	1.02E+00	1.03E − 05	**1.00**	**1.00**
F11	16.07457828	15.32548372	11.00450684	2385.956441	14.50703826	**1.00**
F12	1.09E+160	1.31E+160	7.97E+157	3.45E+158	**1.00**	**1.00**
F13	4.50E+15	4.42E+15	4.50E+15	1.52E+17	1.01E+00	**1.00**
Total	0	1	2	0	4	10

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

Table 10 Best normalized optimization results

	KH	HS	GA	BA	CSA	CSBA
F1	2.19E+10	1.30E+13	1.25E+11	6.66E+12	2.13E+08	**1.00**
F2	2.10E+09	1.18E+06	7.63E+03	5.31E+03	1.48E+00	**1.00**
F3	1.32	1.26	1.26	1.04	1.26	**1.00**
F4	2.61E+07	1.74E+07	**1.00**	8.58E+03	2.61E+07	2.39E+02
F5	1.30E+04	1.67E+01	3.33E+01	1.95E+01	1.01E+01	**1.00**
F6	1.38E+13	1.00E+08	**1.00**	9.46E+06	4.48E+01	1.52E+01
F7	59684.14	113941.10	8.30	61981.75	114711.26	**1.00**
F8	5.87E+160	5.79E+160	1.48E+158	1.94E+158	1.17E+00	**1.00**
F9	2.82E+82	2.04E+82	3.85E+80	1.83E+85	9.10E+36	**1.00**
F10	1.20E+05	8.57E+04	5.12E+03	1.81E+02	**1.00**	**1.00**
F11	6726.62	6726.17	4784.35	1.1004E+306	4834.12	**1.00**
F12	3.14E+162	3.00E+160	9.77E+158	7.95E+159	7.68E+00	**1.00**
F13	4.50E+15	4.36E+15	4.50E+15	5.78E+13	**1.00**	**1.00**
Total	0	0	2	0	2	11

Bold refers to the summation of all number 1.00, which indicates the minimum normalization value (i.e., when the total is higher, the performance is better)

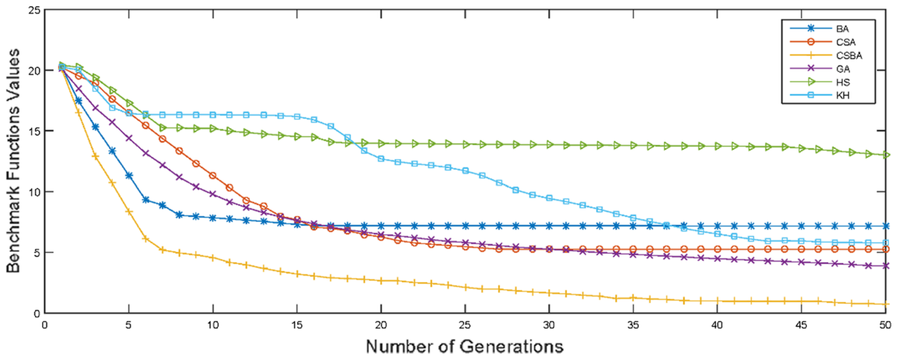


Fig. 3 Performance comparison for F1 Ackley function

Table 9 presents the average of the results. CSBA is the most effective in determining the minimum objective function on 10 of the 13 benchmarks, namely *F1–F2*, *F5–F7*, and *F9–F13*. CSA ranks second and performs best on *F3*, *F8*, *F10*, and *F12*. CSA is followed by GA, which performs best on *F3* and *F4*. HS performs best on F3. Table 10 shows that CSBA performs best on 11 of the 13 benchmarks, namely *F1–F3*, *F5*, and *F7–F13*. CSA and GA are the second most effective; they perform best on the benchmarks *F10*, *F13,* and *F4*, *F6*, respectively. Notably, the results of CSBA that did not achieve the optimal solutions (i.e., *F3*, *F4*, and *F9* in Table 9; *F4* and *F6* in Table 10) are under the multimodal functions, which focus on global and local optima. Furthermore, multimodal functions have complex equations. However, all the results of CSBA in the aforementioned functions are very close to the optimal solution.

The most representative convergent curves are illustrated in Figs. 3–10. The values in the figures are the mean function optima, which are the true values.

Figure 3 shows that CSBA is capable of finding better solutions compared with all the other methods. As shown in the figure, HS converges sharply during the initial search stage; however, as soon as HS is trapped in local minima, the global minimum slightly decreases. Furthermore, BA is close to CSBA during the initial stage, but their difference increases during the second stage. BA, CSA, GA, and KH initially move toward the best solutions, whereas GA converges toward the minimum later than the others. CSBA is the best among all the methods.

Figure 4 shows that CSBA is the fastest method for finding the best solution in the first part, with GA ranking second. By contrast, CSA and KH are the best performers in the second part. As shown in the figure, all the algorithms begin optimization at nearly the same point, whereas CSBA has a more stable convergent speed than the others. Several difficulties are encountered in finding the best results in the original CSA compared with those in the other methods. In this case, CSBA is the most efficient and fastest method for finding the best global function values among the six methods.

Figure 5 shows that the results are nearly similar to the results presented in Fig. 4. Therefore, the original CSA experiences more problems in determining the best global solution compared with the other methods. The results of the initial stage indicate that KH achieves better results than CSBA. However, the final results illustrate that CSBA

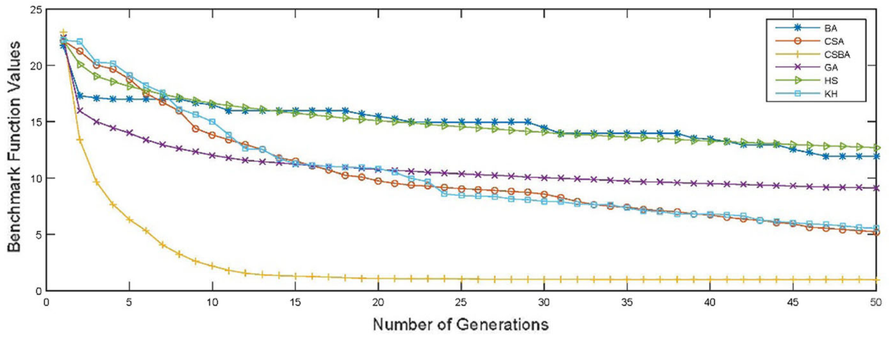


Fig. 4 Performance comparison for F3 Penalty 1 function

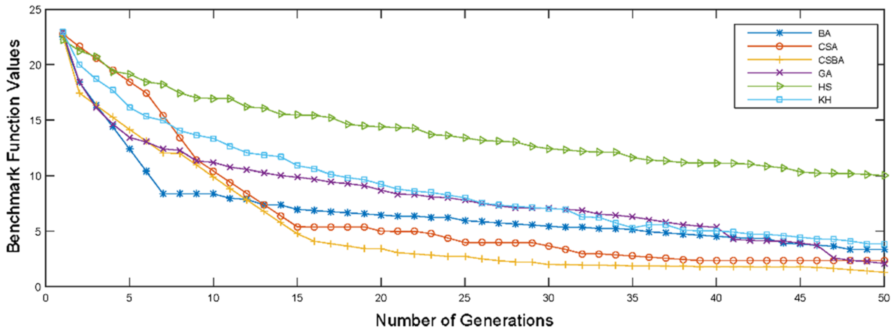


Fig. 5 Performance comparison for F4 Penalty 2 function

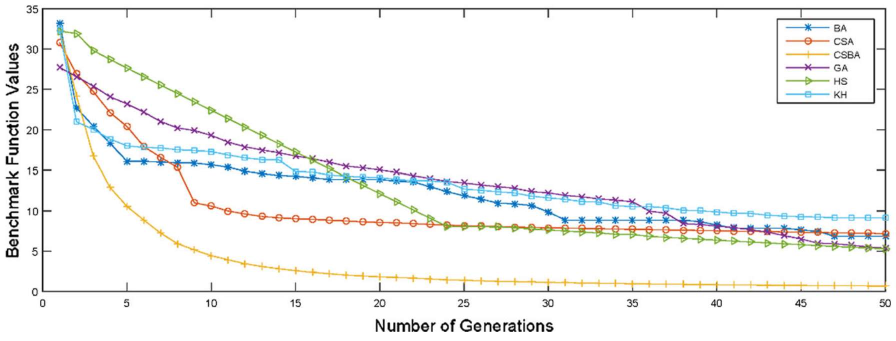


Fig. 6 Performance comparison for F6 Rastrigin function

obtains the best results, whereas KH obtains the worst. HS also has faster convergent speed than the other methods. Moreover, it achieves the closest results to CSBA.

Figure 6 shows that CSBA performs equally with all the other methods. Moreover, BA achieves the best results at the 11th generation. However, CSBA converges in a more stable state for this case compared with the other methods. The combination of CSA and BA demonstrates good performance.

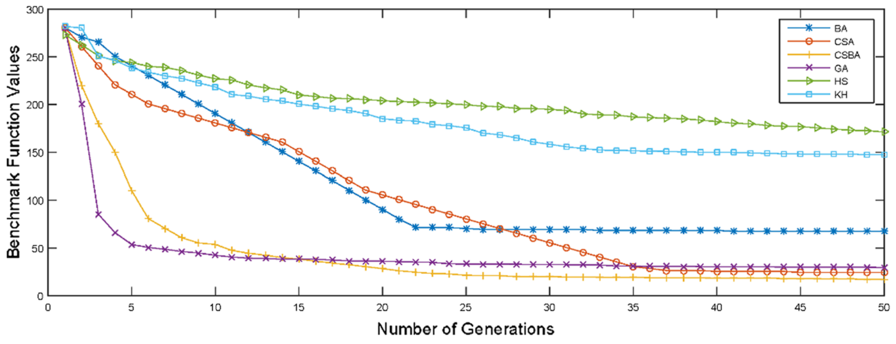


Fig. 7 Performance comparison for F7 Rosenbrock function

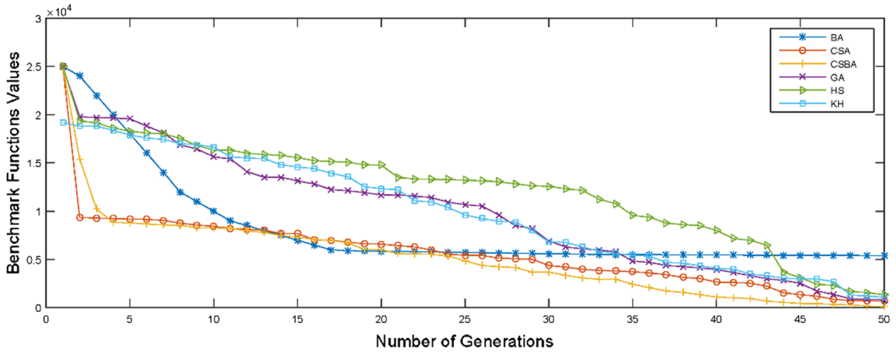


Fig. 8 Performance comparison for F9 Schwefel 1.2 function

Figure 7 shows that CSBA achieves the best performance among the six methods, followed by CSA. GA exhibits the third best performance with a relatively slow and stable convergence rate. Similarly, the results of all the methods are close to one another, with CSBA gaining a slight advantage, as shown in Fig. 8.

Figure 9 illustrates that CSBA performs better than the other methods in the unimodal case, particularly in the second part. In the beginning, GA outperforms CSBA until the 15th generation and then CSA until the end. Figure 10 indicates that CSBA achieves the best performance among the six methods in the optimization process, followed by GA, CSA, and KH.

An analysis of Figs. 3 to 10 determines that our proposed metaheuristic CSBA considerably outperforms the other methods.

5 Conclusion

In this work, CSA is improved by combining it with BA. The hybrid method, CSBA, is then evaluated on various benchmarks. In CSBA, the first population is optimized by CSA during iteration, and the best solution is randomly embedded into the second population evolved by BA. Thereafter, the result obtained by BA is rechecked through

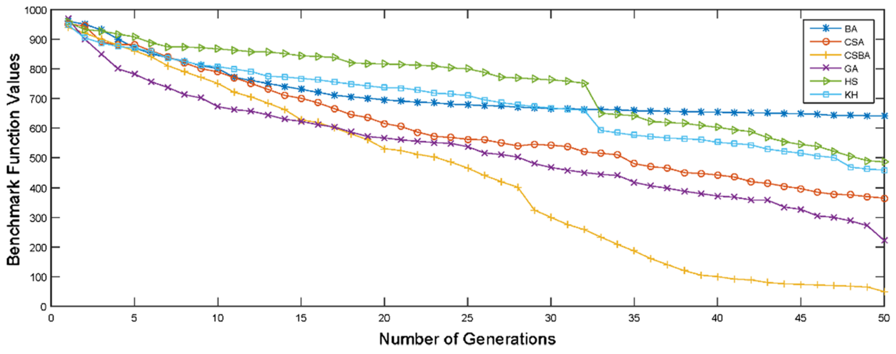


Fig. 9 Performance comparison for F10 Schwefel 2.22 function

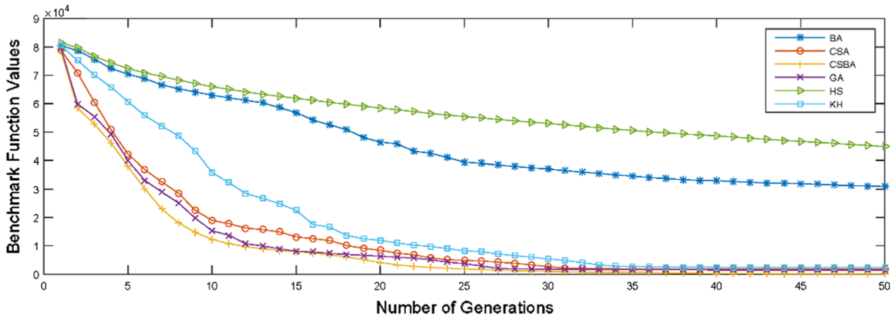


Fig. 10 Performance comparison for F12 Sphere function

the discovery rate in CSA. CSBA, as a combination of CSA and BA, is capable of exploiting the good features of the two basic algorithms and preventing all individuals from getting trapped in inferior local optimal regions. Furthermore, CSBA is investigated on 13 benchmark functions. The results show that CSBA exhibits improved efficiency and effectively compared with other search methods, such as the CSA, BA, GA, HS, and KH. CSBA can be applied to other benchmark functions, such as real-world optimization problems, for further examination.

Finally, future work should focus on local search algorithms, such as hill climbing, SA, and TS. Furthermore, different algorithms should be applied to compare them with one another. The performance of the proposed algorithm on other benchmarks and on real-life problems should also be investigated.

Acknowledgements This work has been supported by the grant, account number 1001/PKOMP/8014016 under the Universiti Sains Malaysia (USM).

References

1. Alomari OA, Khader AT, Al-Betar MA, Abualigah LM (2017) Gene selection for cancer classification by combining minimum redundancy maximum relevancy and bat-inspired algorithm. Int J Data Min Bioinform 19(1):32–51

2. Alomari OA, Khader AT, Al-Betar MA, Awadallah MA (2018) A novel gene selection method using modified MRMR and hybrid bat-inspired algorithm with β-hill climbing. Appl Intell 48(4):1–19
3. Back T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford
4. Bolaji AL, Al-Betar MA, Awadallah MA, Khader AT, Abualigah LM (2016) A comprehensive review: Krill herd algorithm (kh) and its applications. Appl Soft Comput 49:437–446
5. Dainson M, Mark M, Hossain M, Yoo B, Holford M, McNeil SE, Riehl C, Hauber ME (2018) How to make a mimic? Brood parasitic striped cuckoo eggs match host shell color but not pigment concentrations. J Chem Ecol 44(5):1–7
6. Dieterich JM, Hartke B (2012) Empirical review of standard benchmark functions using evolutionary global optimization. arXiv:1207.4318
7. Digalakis JG, Margaritis KG (2002) An experimental study of benchmarking functions for genetic algorithms. Int J Comput Math 79(4):403–416
8. Dixon L (1978) The global optimization problem. An introduction. Toward Glob Optim 2:1–15
9. Gagnebin Y, Tonoli D, Lescuyer P, Ponte B, de Seigneux S, Martin PY, Schappler J, Boccard J, Rudaz S (2017) Metabolomic analysis of urine samples by UHPLC-QTOF-MS: impact of normalization strategies. Analytica Chimica Acta 955:27–35
10. Glover F (1977) Heuristics for integer programming using surrogate constraints. Decis Sci 8(1):156–166
11. Griewank AO (1981) Generalized descent for global optimization. J Optim Theory Appl 34(1):11–39
12. Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor
13. Jafri R, Ali SA, Arabnia HR (2013) Computer vision-based object recognition for the visually impaired using visual tags. In: Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p 1
14. Jafri R, Arabnia HR (2008) Fusion of face and gait for automatic human recognition. In: ITNG 2008. Fifth International Conference on Information Technology: New Generations, 2008. IEEE, pp 167–173
15. Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimisation problems. Int J Math Model Numer Optim 4(2):150–194
16. Kanagaraj G, Ponnambalam S, Jawahar N, Nilakantan JM (2014) An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization. Eng Optim 46(10):1331–1351
17. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department
18. Kirkpatrick S, Gelatt CD, Vecchi MP et al (1983) Optimization by simulated annealing. Science 220(4598):671–680
19. Koza JR (1994) Genetic programming ii: automatic discovery of reusable subprograms. MIT Press, Cambridge
20. Koziel S, Yang XS (2011) Computational optimization, methods and algorithms, vol 356. Springer, Berlin
21. Laguna M, Martí R (2005) Experimental testing of advanced scatter search designs for global optimization of multimodal functions. J Global Optim 33(2):235–255
22. Layeb A (2011) A novel quantum inspired cuckoo search for knapsack problems. Int J Bio-Inspired Comput 3(5):297–305
23. Li X, Wang J, Yin M (2014) Enhancing the performance of cuckoo search algorithm using orthogonal learning method. Neural Comput Appl 24(6):1233–1247
24. Long W, Jiao J (2014) Hybrid cuckoo search algorithm based on powell search for constrained engineering design optimization. WSEAS Trans Math 13:431–440
25. Luper D, Cameron D, Miller J, Arabnia HR (2007) Spatial and temporal target association through semantic analysis and gps data mining. IKE 7:25–28
26. Mirjalili S, Gandomi AH (2017) Chaotic gravitational constants for the gravitational search algorithm. Appl Soft Comput 53:407–419
27. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
28. Schwefel HP (1981) Numerical optimization of computer models. Wiley, Hoboken
29. Shehab M, Khader A, Laouchedi M (2018) A hybrid method based on cuckoo search algorithm for global optimization problems. J ICT 17(3):469–491

30. Shehab M, Khader AT, Al-Betar M (2016) New selection schemes for particle swarm optimization. IEEJ Trans Electron Inf Syst 136(12):1706–1711. https://doi.org/10.1541/ieejeiss.136.1706
31. Shehab M, Khader AT, Al-Betar MA (2017) A survey on applications and variants of the cuckoo search algorithm. Appl Soft Comput 61:1041–1059
32. Shehab M, Khader AT, Al-Betar MA, Abualigah LM (2017) Hybridizing cuckoo search algorithm with hill climbing for numerical optimization problems. In: 2017 8th International Conference on Information Technology (ICIT). IEEE, pp 36–43
33. Shehab M, Khader AT, Laouchedi M (2017) Modified cuckoo search algorithm for solving global optimization problems. In: International Conference of Reliable Information and Communication Technology. Springer, pp 561–570
34. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
35. Storn R, Price KV (1996) Minimizing the real functions of the ICEC'96 contest by differential evolution. In: International Conference on Evolutionary Computation, pp 842–844
36. Wang F, Luo L, He XS, Wang Y (2011) Hybrid optimization algorithm of PSO and cuckoo search. In: 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIM- SEC). IEEE, pp 1172–1175
37. Wang GG, Gandomi AH, Zhao X, Chu HCE (2016) Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Comput 20(1):273–285
38. Yang XS (2010) Firefly algorithm. Eng Optim:221–230
39. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010). Studies in Computational Intelligence, vol 284. Springer, Berlin, Heidelberg
40. Yang XS (2008) NIM algorithms. Luniver Press, Beckington
41. Yang XS, Deb S (2009) Cuckoo search via l´evy flights. In: World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009. IEEE, pp 210–214
42. Yang XS, Deb S (2014) Cuckoo search: recent advances and applications. Neural Comput Appl 24(1):169–174
43. Yang XS, Deb S (2017) Cuckoo search: state-of-the-art and opportunities. In: 2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence (ISCMI). IEEE, pp 55–59
44. Yang XS, He X (2013) Bat algorithm: literature review and applications. Int J Bio-Inspired Comput 5(3):141–149
45. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102