CrossMark

Multi-species Cuckoo Search Algorithm for Global Optimization

Xin-She Yang[1] · Suash Deb[2,3] · Sudhanshu K. Mishra[4]

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Many optimization problems in science and engineering are highly nonlinear and thus require sophisticated optimization techniques to solve. Traditional techniques such as gradient-based algorithms are mostly local search methods and often struggle to cope with such challenging optimization problems. Recent trends tend to use nature-inspired optimization algorithms. The standard cuckoo search (CS) is an optimization algorithm based on a single cuckoo species and a single host species. This work extends the standard CS by using the successful features of the cuckoo-host co-evolution with multiple interacting species. The proposed multi-species cuckoo search (MSCS) intends to mimic the co-evolution among multiple cuckoo species that compete for the survival of the fittest. The solution vectors are encoded as position vectors. The proposed algorithm is then validated by 15 benchmark functions as well as five nonlinear, multimodal case studies in practical applications. Simulation results suggest that the proposed algorithm can be effective for finding optimal solutions and all optimal solutions are achievable in the tested cases. The results for the test benchmarks are also compared with those obtained by other methods such as the standard cuckoo search and genetic algorithm. The comparison has demonstrated the efficiency of the present algorithm. Based on numerical experiments and case studies, we can conclude that the proposed algorithm can be more efficient in most cases. Therefore, the proposed approach can be a very effective tool for solving nonlinear global optimization problems.

Keywords Cuckoo search · Nature-inspired algorithm · Multi-species cuckoo search · Optimization · Swarm intelligence

Introduction

Many applications involve nonlinear optimization and thus require sophisticated optimization algorithms to solve. Such applications can be very diverse, spanning many areas and disciplines from engineering designs and scheduling to data mining and machine learning [13, 33, 40, 41, 43]. One of

✉ Xin-She Yang
xy227@cam.ac.uk

1 School of Science and Technology, Department of Design Engineering and Mathematics, Middlesex University, London NW4 4BT, UK

2 Ranchi, India

3 Distinguished Professorial Associate, Decision Sciences and Modelling Program, Victoria University, Melbourne, Australia

4 Department of Economics, North-Eastern Hill University, Shillong, India

the current trends is to use metaheuristic algorithms inspired by the successful characteristics in nature. Among these new algorithms, cuckoo search (CS) has been shown to be powerful in solving many problems [41, 43]. This standard CS version was mainly designed for single objective optimization problems, which was later extended to multi-objective optimization [46].

Both standard cuckoo search and its multiobjective extension used simplified characteristics to represent the brood parasitism of a single cuckoo species and their interactions with a single host species. However, the reality is far more complicated in the cuckoo-host co-evolution systems [10]. The co-evolution often involves multiple cuckoo species that compete with each other and compete for the resources of host bird species, while the hosts can also have multiple species. Cuckoo species tend to evolve to lay eggs with the mimicry of the size, colours, and texture of the eggs of host birds, often with critical timing advantage. On the other hand, host birds can counter-act such parasitism by developing their defensive strategies to identify potential

intruding eggs and minimize the risk of hatching cuckoo eggs. This arms race is both co-evolutionary and ongoing [11, 27], and the co-evolution seems to promote species richness and subspecies variations as well as diversity in parasitic cuckoos [21]. The richness of characteristics in cuckoo-host co-evolution provides us an inspiration to potentially design better optimization algorithms.

One way to model such co-evolution has been proposed by Mishra [22] to use a host-parasite co-evolutionary approach where both parasites and hosts took random flights. The probability of detection or rejection of eggs was dynamic, depending on the accumulative success of the parasites such as cuckoos. This approach has been used to solve both function optimization problems and completing incomplete correlation matrix [22]. However, this approach only captured a very small part of the major characteristics of the cuckoo-host co-evolutionary systems.

In order to capture more detailed characteristics of this co-evolution system, in this paper, we extend the original cuckoo search to a new multi-species co-evolutionary cuckoo search algorithm. The key idea is to simulate the main co-evolutionary behavior of both cuckoo species and host species. The key novelty of this approach is to allow different species of cuckoos (via more than two population groups) to compete and co-evolve with a host species (via a host population). This multi-population system is an intrinsic multi-swarm system, which can increase the selection pressure to drive the system towards better strategies and solutions.

Therefore, the paper is organized as follows. The "The Original Cuckoo Search" section summarizes the original cuckoo search and its main equations. The "Multi-Species Cuckoo Search" section outlines the novel features of the new multi-species cuckoo search, followed by the numerical experiments on 15 different test benchmarks in the "Validation by Numerical Experiments" section. The "Practical Applications" section presents the results of five different case studies concerning engineering designs, inverse parameter estimation and data classification. The paper concludes with discussions about further research directions in the "Discussions" section.

The Original Cuckoo Search

Cuckoo search (CS) is a nature-inspired metaheuristic algorithm, developed in 2009 by Xin-She Yang and Suash Deb [41]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights [28], rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms [17, 42].

A relatively comprehensive review of the studies up to 2014 was carried out by Yang and Deb [43].

Cuckoo Search and its Algorithmic Equations

In the natural world, many cuckoo species (59 species among 141 cuckoo species) engage the so-called obligate reproduction parasitism strategy. There is an evolutionary arms race between such cuckoo species and their associated host species [10, 11, 27]. Based on such phenomena, Yang and Deb developed the standard cuckoo search in 2009 [41]. The standard CS uses three simplified rules:

1. Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest. An egg corresponds to a solution vector.
2. The best nests with high-quality eggs (solutions) will be carried over to the next generation.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest at a new location.

In the original cuckoo search, there is no distinction between an egg, a nest, or a cuckoo. In this simplified scenario, each nest corresponds to one egg which also represents one cuckoo, which makes it much easier to implement [42]. Mathematically speaking, cuckoo search uses a combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \beta s \otimes H(p_a - \epsilon) \otimes (\mathbf{x}_j^t - \mathbf{x}_k^t), \quad (1)$$

where $\mathbf{x}_j^t$ and $\mathbf{x}_k^t$ are two different solutions selected randomly by random permutation. Here, $H(u)$ is a Heaviside function, ϵ is a random number drawn from a uniform distribution, and s is the step size. In addition, $\otimes$ denotes an entry-wise multiplication operator, and β is the small scaling factor.

On the other hand, the global random walk is carried out by using Lévy flights

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \otimes L(s, \lambda), \quad (2)$$

where

$$L(s, \lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi \lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg 0), \quad (3)$$

where $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest. Here, "$\sim$" highlights the fact that the search steps in terms of random

numbers $L(s, \lambda)$ should be drawn from the Lévy distribution on the right-hand side of Eq. 3. It is worth pointing out that Eq. 3 is an approximation to the Lévy distribution by a power-law distribution with an exponent λ. The use of Lévy flights enables the algorithm with higher ergodicity and explorability, which makes the algorithm more likely to jump out of any local optima [28, 41, 43].

Cuckoo Search in Applications

Since the development of the cuckoo search algorithm in 2009, it has been applied in many areas, including optimization, engineering design, data mining, and computational intelligence with promising efficiency. From the case studies in engineering design applications, it has been shown that cuckoo search has superior performance to other algorithms for a range of continuous optimization problems [16, 17, 42, 43, 50]. A review by Yang and Deb covered the literature up to 2013 [43], while a review by Fister et al. covered the literature up to 2015. Another review by Mohamad et al. focused on applications up to 2014 [24]. The most recent literature review has been carried out by Shehab et al. [32], which covers some of the most recent literature up to 2017. These reviews have highlighted some of the diverse applications using cuckoo search and its variants.

There are many other applications, including vehicle component optimization [15], wireless sensor networks [12], training neural networks [35], runoff-erosion modelling [31], phase equilibrium in thermodynamic calculations [3], network optimization [25], and scheduling [7]. In addition, cuckoo search has also been applied to multilevel color image segmentation [26], biodiesel engine optimization [38], graphic objective feature extraction [39], fractional order PID control design [51], vulnerabilities mitigation [53], and others [45].

From the algorithm development perspective, different cuckoo search variants have been developed by introducing new components or hybridizing with other algorithms. For example, a binary cuckoo search for feature selection has been developed by Pereira et al. [29]. Walton et al. [36] developed a modified cuckoo search for solving complex mesh generation in engineering simulation, while Zheng and Zhou [52] provided a variant of cuckoo search using Gaussian process. Mlakar et al. developed a self-adaptive cuckoo search [23], while Wang et al. enhanced cuckoo search with chaotic maps [37].

As a further extension, Yang and Deb [46] developed a multiobjective cuckoo search (MOCS) algorithm for engineering design applications. Recent studies have demonstrated that cuckoo search can perform significantly better than other algorithms in many applications [17, 45, 50]. However, none of the above variants of cuckoo search use the characteristics of multi-species co-evolution, and one of the main aims of this paper is to explore this area so as to further enhance and improve the CS algorithm.

Multi-species Cuckoo Search

In the original cuckoo search and its many variants, there is only one cuckoo species interacting with one species of host birds. In the standard cuckoo search, a cuckoo is allowed to lay a single egg and each nest contains only a single egg. This is a very simplified scenario. In the real cuckoo-host systems, it is observed that multiple cuckoo species co-evolve with one or more host species to compete for survival of the fittest by brood parasitism [10, 27]. Loosely speaking, cuckoos can evolve to subspecies with speciation, and they can be subdivided into different gentes targeting at different host species [11, 21]. The interaction dynamics can be very complex, forming an on-going, co-evolutionary arms race between cuckoo subspecies and host species as well as different cuckoo species. Field studies have shown that such co-evolution may promote species richness in parasitic cuckoos with the enhanced speciation and extinction rates [21]. Strictly speaking, different species, subspecies and gentes are used in the biological literature and their meaning can be different [21, 27]; however, we will simply use species here for simplicity.

Based on the above characteristics of cuckoo-host co-evolution, we can extend the original cuckoo search to capture more realistic cuckoo-host co-evolution to develop a new multi-species co-evolution cuckoo search, or multi-species cuckoo search (MSCS) for short. In order to describe the MSCS in detail, we use the following idealized rules/assumptions:

1. There are multiple cuckoo species (m species) that compete and co-evolve with a host species. The arms race between cuckoo species and the host species obeys the survival of the fittest. Both the best cuckoos and the best hosts will pass onto the next generation.
2. Each cuckoo of any cuckoo species can lay r eggs inside a randomly selected host nest. Each cuckoo egg has a probability p_a to be discovered (and then abandoned) by the host (thus the survival probability is $1 - p_a$).
3. Each nest contains q eggs. If the fraction/ratio of cuckoo eggs is higher than $1 - p_a$, the host bird can abandon the nest and then fly away to build a new nest in a new location.

Based on these rules for MSCS, we can represent them more mathematically. Each solution vector $\mathbf{x}_i$ to an optimization problem in a D-dimensional space is represented by an egg. Therefore, an egg is equivalent to a

solution vector, and a nest represents a group of q solutions.

In general, there are $m \geq 1$ species with a total population of n, each species has n_i ($i = 1, ..., m$) cuckoos such that

$$\sum_{i=1}^{m} n_i = n. \tag{4}$$

Each cuckoo lays $r \geq 1$ eggs. There are w host nests, and each nest can have $q \geq 1$ eggs on average. So the total number of eggs in the host nests are $N_h = wq$, which should be greater than n. That is, $N_h \geq n$. In nature, it is estimated that approximately 1/4 to 1/2 of eggs in the host nests are cuckoo eggs. Thus, for simplicity, we can set $n = N_h/2$ in this study.

Competition can occur at three different levels: intraspecies competition, inter-species competition, and cuckoo-host competition. Even for a single species, cuckoos within the same species can compete for host nests, which is intraspecies competition. For multiple species, one species of cuckoos can compete with cuckoos from other species by their egg-replacing strategy. The most significant competition is the cuckoo-host competition. All these three kinds of competition interact and co-evolve to form a complex system, leading to potential self-organizing intelligent behavior.

In the co-evolutionary system, different cuckoo species compete for the survival of the fittest, and cuckoos from one species can take over other cuckoos' territory or replace eggs laid by other cuckoo species. This can simply be realized mathematically by random swapping its location vector with another in a dimension-by-dimension manner. This binary random swapping operator can be achieved by the following two equations:

$$\mathbf{x}_a^{(\text{new})} = \mathbf{x}_a \otimes (1 - \mathbf{Q}) + \mathbf{x}_b \otimes \mathbf{Q}, \tag{5}$$

and

$$\mathbf{x}_b^{(\text{new})} = \mathbf{x}_a \otimes \mathbf{Q} + \mathbf{x}_b \otimes (1 - \mathbf{Q}), \tag{6}$$

where $\mathbf{x}_a$ is randomly selected from cuckoo species a, while $\mathbf{x}_b$ is selected from cuckoo species b. Here, Q is a random binary vector with the same length of $\mathbf{x}_a$ and each of its components is either 1 or 0 [i.e. $Q_k \in \{0, 1\}$ ($k = 1, 2, ..., D$)]. For example, $\mathbf{Q} = [1, 0, 0, 1, 0, 1, 1]$ is a binary vector in a seven-dimensional space. Again, $\otimes$ means that the operation is a component-wise or dimension-wise operation.

The main steps for implementing and simulating the above idealized characteristics are as follows:

1. There are two population sets: a total population of n cuckoos for m cuckoo species (each has its own

population n_j ($j = 1, 2, ..., m$)), and a population of N_h host nests. Thus, there are two initial best solutions: $\mathbf{g}_{cs}^*$ to denote the best cuckoo among all m cuckoo species (each species has its own best cuckoo $\mathbf{g}_j^*$) and $\mathbf{g}_h^*$ to denote the best host in terms of objective fitness.

2. For each generation of evolution, each cuckoo (say, cuckoo i) from one cuckoo species (say, species j) can lay r eggs in a randomly selected host nest (say, nest k). The newly laid eggs will replace randomly selected eggs in the nest so that the total number of eggs ($q \geq 1$) in the nest remains constant. The main equation for this action can be carried out by Eq. 1.

3. For any new egg laid by a cuckoo, there is a probability of p_a to be discovered. Among q eggs, if the fraction of cuckoo eggs exceeds $1 - p_a$, the host can abandon the nest completely and fly away to build a new nest at a new location via Eq. 2.

4. Different species of cuckoos compete for their survival and territories, thus they can lay eggs in the nests that other cuckoos just laid. This competition is equivalent to replacing or swapping its own eggs with another cuckoo's eggs from different species. Thus, it can be achieved by randomly swapping their components dimension by dimension via Eqs. 5 and 6.

5. Random mixing is carried out in terms of egg-laying and nest choices among different cuckoo species and the host species.

6. Both the best cuckoos and host nests (in terms of their fitness) should pass onto the next generation.

These key steps can be schematically represented as the pseudocode in Algorithm 1. To illustrate the main ideas, for two species of cuckoos with a total population of $n = 40$, we have $m = 2$ and $n = 40$. If two species have the same population size, we have $n_1 = n_2 = n/2 = 20$. For simplicity, if all nests have the same number of four eggs in 20 nests, we have $q = 4$ and $w = 20$, thus there $N_h = 20 * 4 = 80$ eggs in all the nests. In addition, if each cuckoo lays one egg at a time (i.e., $r = 1$), this means that there are $n \times r = 40$ cuckoo eggs in the cuckoo-host system. Thus, in this case, there are exactly 50% of the eggs belong to cuckoos in the combined population of cuckoo species and host nests.

Obviously, the number of cuckoo eggs in a particular nest can be randomly distributed from 1 to $q = 4$. For $p_a = 0.25$, if there are 3 or 4 eggs in a nest, one egg may typically belong to a cuckoo. If the number of alien eggs is higher, this nest can be abandoned by its host, and thus a new replacement nest with q new eggs (or randomly generated solutions) will be built in a new location. This new location is typically far enough from the original location to minimize the risk of being close to the same cuckoos.

Algorithm 1 Multi-species co-evolutionary cuckoo search

> Initialize parameters, cuckoos and host populations;
> Find the best cuckoo $\mathbf{g}_{cs}^*$ and best host $\mathbf{g}_h^*$;
> **while** *(stopping criterion is not met)* **do**
>> Choose a cuckoo species randomly (say, species j);
>> Select a cuckoo in the species randomly (say, i);
>> Generate a random number ϵ from [0,1];
>> **if** $\epsilon < p_a$ *(discovery probability)* **then**
>>> Generate a new solution by Eq. 1;
>> **else**
>>> Perform a Lévy flight by Eq. 2;
>> **end**
>> Put into a host nest randomly (say, nest k);
>> Generate a random binary vector $\mathbf{Q}$ of D dimensions ;
>> Swap two randomly selected eggs from two different species by Eqs. 5 and 6;
>> Evaluate all new fitness/objectives;
>> **if** *New solution is better* **then**
>>> Replace the worst egg in nest k by the new solution;
>> **end**
>> Update the best solution $\mathbf{g}_{cs}^*$ among all cuckoos;
>> **if** *cuckoo eggs ratio in nest k is higher than $1 - p_a$* **then**
>>> Perform a Lévy flight and build a new nest at a new location via Eq. 2;
>> **end**
>> Update the current best host $\mathbf{g}_h^*$ among all nests;
>> Pass the best cuckoos in each species and host nests to next generation;
>> Find and record the overall best solution $\mathbf{x}_*$;
>> Update iteration counter t;
> **end**

It is worth pointing out that there seems to have some similarity between multi-species cuckoo interactions and the multi-swarm optimization in the literature [5]. However, there are two key differences here: the multi-species cuckoo search (MSCS) mimics the co-evolution between parasite cuckoo species and host species, while multi-swarms mainly split a population of the same kind into subgroups or subswarms. In addition, the share of information in MSCS is among the same cuckoo species and the same host species, not directly shared among competing species. Such information-sharing structure can potentially enable more extensive exploitation of local information as well as global information. On the other hand, multi-swarms tend to share information among all subswarms. Furthermore, different cuckoo species compete for their own survival, while the

multi-swarms do not compete. These differences mean that MSCS is not a simple multi-swarm system, rather it is an interacting co-evolving multi-swarm system. Therefore, different algorithmic characteristics and performance can be expected.

Validation by Numerical Experiments

All new algorithms have to be extensively tested by a diverse range of benchmarks and case studies. As a preliminary test, we will use a subset of 15 function benchmarks and 5 case studies.

Benchmarks

For this purpose, we have selected 15 benchmark functions with different modalities and landscapes. These benchmarks are part of traditional optimization functions, the CEC2005 test suite and most recent CEC2015 test functions. The chosen set of benchmarks have diverse properties so that we can test the proposed algorithm more thoroughly.

The first function is the shifted sphere function f_1 from the CEC2005 benchmark suite [34]. This function has the global minimum $f_{1,\min} = -450$ in the domain $-100 \le x_i \le 100$.

The second function is Ackley function

$$f_2(\mathbf{x}) = -20 e^{-\frac{1}{5}(\frac{1}{D}\sum_{i=1}^{D} x_i^2)^{1/2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)} + 20 + e, \quad (7)$$

which has its global minimum $f_* = 0$ at $(0, 0, ..., 0)$. This function is highly nonlinear and multimodal.

The third function is Xin-She Yang's forest-like function

$$f_3(\mathbf{x}) = \left(\sum_{i=1}^{D} |x_i|\right) \exp\left[-\sum_{i=1}^{d}\sin(x_i^2)\right], \quad (8)$$

which has the global minimum $f_* = 0$ at $(0, 0, ..., 0)$ in the domain of $-2\pi \le x_i \le 2\pi$. This function is highly nonlinear and multimodal, and its first derivatives do not exist at the optimal point due to the modulus $|x_i|$ factor.

The fourth function is the shifted Schwefel's problem with noise in fitness as given in CEC2005 benchmark suite [34], which has the mean global minimum $f_{4,\min} = -450$ in the domain $-100 \le x_i \le 100$.

The fifth function is Schwefel's Problem 2.22 [49]

$$f_5(\mathbf{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \quad (9)$$

which has the global minimum $f_{5,\min} = 0$ at $\mathbf{x} = (0, 0, ..., 0)$ in the domain $-10 \le x_i \le 10$. This function is unimodal.

The sixth function is the shifted Rosenbrock function f_6 of CEC2005 benchmark suite with the minimum $f_{6,min} = 390$ in $-100 \leq x_i \leq 100$.

The seventh function is the shifted and rotated Griewank function with the minimum $f_{7,min} = -180$ in the domain of $0 \leq x_i \leq 600$.

The eighth function is Function 23 of the CEC2005 benchmarks [34], which is a non-continuous rotated hybrid composition function with the minimum $f_{8,min} = 360$ in $-5 \leq x_i \leq 5$.

The ninth function is Function 24 of the CEC2005 benchmarks [34], which is a rotated hybrid composition function with $f_{9,min} = 260$ in $-5 \leq x_i \leq 5$.

The tenth function is Function 25 of the CEC2005 benchmark suite [34], which is a rotated hybrid composition function without bounds with the minimum of $f_{10,min} = 260$ in $[2, 5]^D$.

The next five functions are taken from the CEC2015 benchmark suite [8, 30]. The 11th function is the rotated bent cigar function with the minimum $f_{11,min} = 100$, while the 12th function is the rotated Discus function with the minimum of $f_{12,min} = 200$. The 13th function is the shifted and rotated Weierstrass function with $f_{13,min} = 300$, and the 14th function is the shifted and rotated Schwefel's function with $f_{14,min} = 400$. Finally, the 15th function is the shifted and rotated Katsuura function with $f_{15,min} = 500$. All these functions have variables in the domain of $[-100,100]^D$ where D is the dimensionality of the functions.

Parameter Settings

In our implementations, we have used $n_1 = n_2 = 20$, $n = 40$, $r = 1$, $m = 2$, $N_h = 80$, $q = 4$ and $w = 20$ for the two sets of populations. For the parameters in the algorithmic equations, we have used $\alpha = \beta = 0.01$, $\lambda = 1.5$, $p_a = 0.25$ and a fixed number of iterations $t_{max} = 1000$ as the stopping criterion. These parameter settings are based on a preliminary parametric study in our simulations and the suggestions in the literature [41, 43].

In addition, $D = 10$ is used for all the test functions in the first experiment. Then, $D = 50$ is used for the second numerical experiment with all other parameter values remaining the same. For both the standard cuckoo search (CS) and the proposed MSCS, we have used $n_{cs} = 80$ so that the total numbers of function evaluations remain the same for all algorithms. Thus, the fairness of the comparison in terms of function evaluations is ensured.

Results

Each algorithm has been run for 100 trials so as to calculate meaningful statistics such as the best (minimum) objective values and the means of the obtained solutions. The error is

defined as the absolute value of the difference between the best $f(\mathbf{x}_*)$ found by the algorithm and the true minimum $f_{min}(true)$. That is

$$E_f = |f(\mathbf{x}_*) - f_{min}(true)|. \tag{10}$$

The numerical results are summarized in Table 1 where the best corresponds to the minimum of E_f and the mean corresponds to the average value of E_f.

From Table 1, we can see clearly that MSCS can obtain better results in all the benchmarks. The diversity among the cuckoo-host populations in MSCS is higher than those in CS, and the MSCS can be potentially more robust. This will in general promote the exploration ability of the search process.

Another way of looking at the simulation results is to analyze and compare the convergence behavior. In fact, MSCS converges faster than CS for all the test functions by tracing both the minimum objective values found during iterations. For example, for function f_5, its convergence plot is shown in Fig. 1 where we can see that MSCS converges faster even from the very early iterations. Other benchmarks show similar characteristics.

In order to test the proposed algorithm for solving higher-dimensional problems, we have also tested the same set of function benchmarks for $D = 50$. In most studies, researchers tend to use higher numbers of iterations for higher values of D, typically $t = 1000D$. But we have used the same settings of the parameters as before; that is, $t_{max} = 1000$. The results are summarized in Table 2.

As we can see from Table 2 that the MSCS obtained better results for almost all functions, except for the shifted and rotated Weierstrass function f_{13}. Even for this function

Table 1 Errors $|f(\mathbf{x}_*) - f_{min}(true)|$ for $D = 10$

| Function	CS		MSCS	
	Best	Mean	Best	Mean
f_1	2.97E−09	1.71E−06	2.21E−11	3.25E−08
f_2	2.12E−09	1.69E−08	1.41E−11	5.79E−09
f_3	7.02E−07	5.86E−06	3.68E−10	2.41E−09
f_4	3.56E−07	2.23E−04	8.17E−08	7.91E−05
f_5	4.11E−07	5.39E−06	1.01E−09	5.11E−08
f_6	1.25E−09	2.77E−08	7.23E−10	5.98E−09
f_7	2.17E−08	5.25E−08	2.49E−09	5.14E−09
f_8	4.51E+01	7.26E+02	1.37E+01	9.40E+01
f_9	2.65E+02	7.01E+02	3.86E+01	2.08E+01
f_{10}	7.92E+02	7.89E+02	9.32E+01	2.65E+01
f_{11}	8.14E+02	9.01E+02	1.27E+02	6.32E+02
f_{12}	2.59E+02	6.87E+02	3.92E+01	7.41E+01
f_{13}	2.76E+03	8.23E+03	4.62E+02	6.98E+03
f_{14}	5.89E+03	7.54E+03	2.51E+03	3.42E+03
f_{15}	2.83E+03	2.57E+03	1.49E+03	4.07E+03
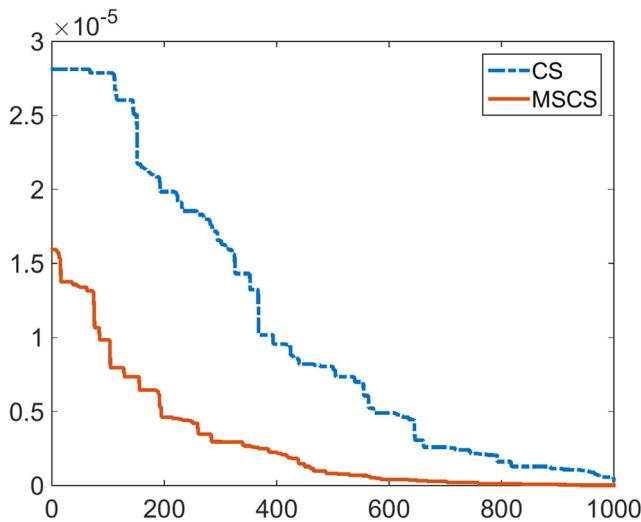
Fig. 1 Convergence plot for f_5 during iterations

(f_{13}), the two algorithms obtained the same orders of results, but the variation of MSCS is smaller. This means that the MSCS can obtain optimal solutions with sufficient robustness.

Practical Applications

To test the proposed MSCS algorithm further, we now use five test problems in real-world applications with diverse properties and nonlinearity. Three case studies are engineering design problems and they are mostly mixed integer programming problems. The fourth case study is the parameter

Table 2 Errors $|f(\mathbf{x}_*) - f_{\min}(\text{true})|$ for $D = 50$

Function	CS		MSCS	
	Best	Mean	Best	Mean
f_1	2.21E−06	7.43E−07	5.04E−09	6.37E−07
f_2	2.83E−08	9.74E−07	1.42E−09	1.75E−07
f_3	3.35E−06	6.21E−05	1.91E−07	3.22E−08
f_4	1.94E−05	7.73E−01	3.89E−04	8.79E−01
f_5	4.51E−05	6.12E−03	4.81E−06	7.33E−06
f_6	3.31E−07	9.17E−06	2.92E−07	9.05E−06
f_7	5.27E−06	7.51E−06	3.88E−07	4.71E−06
f_8	2.83E+03	8.12E+03	0.91E+03	1.38E+03
f_9	8.62E+03	8.90E+03	2.01E+03	8.92E+03
f_{10}	5.22E+04	9.27E+04	1.59E+03	8.33E+03
f_{11}	6.73E+03	7.93E+03	2.25E+02	7.16E+03
f_{12}	3.12E+03	8.89E+03	1.37E+03	4.70E+03
f_{13}	2.24E+04	7.67E+05	2.31E+04	5.91E+04
f_{14}	7.36E+04	4.93E+05	4.98E+04	9.87E+04
f_{15}	5.05E+04	9.28E+04	2.05E+03	6.69E+04

estimation problem or an inverse problem, which requires to solve a second-order differential equation to calculate the values of objective function. The final problem is the data clustering using the well-known Fisher's iris flower data set.

It is worth pointing out that these case studies are seemingly simple, but they are quite hard to solve due to high nonlinearity, multimodality and irregular search domains. They are nonlinear global optimization problems. In addition, the pressure vessel problem is also a mixed integer programming problem, which is much harder to solve, compared its continuous counterpart.

Design of a Spring

Let us start with a simple but nonlinear problem about the design of a spring under tension or compression from a metal wire [2, 9]. There are three design variables: the wire diameter (r), the mean coil diameter (d), and the number (N) of turns/coils. The objective is to minimize the overall weight of the spring

$$\text{minimize} \quad f(\mathbf{x}) = (2 + N)r^2 d, \tag{11}$$

subject to nonlinear constraints:

$$g_1(\mathbf{x}) = 1 - \frac{Nd^3}{71785r^4} \leq 0,$$
$$g_2(\mathbf{x}) = \frac{d(4d - r)}{12566r^3(d - r)} + \frac{1}{5108r^2} - 1 \leq 0, \tag{12}$$

$$g_3(\mathbf{x}) = 1 - \frac{140.45r}{d^2 N} \leq 0, \quad g_4(\mathbf{x}) = (d + r) - 1.5 \leq 0. \tag{13}$$

Some simple bounds or limits for the design variables are

$$0.05 \leq r \leq 2.0, \quad 0.25 \leq d \leq 1.3, \quad 2.0 \leq N \leq 15.0. \tag{14}$$

Using the proposed MSCS with the same parameter settings given in the "Parameter Settings" section, the results of 20 different runs are summarized in Table 3 where comparison is also made. As we can see, MSCS can obtain the best or the same results as the best results in the literature.

Table 3 Comparison of optimal solutions for spring design

Author	Optimal solution	Best objective
Arora [2]	(0.053396, 0.399180, 9.185400)	0.01273
Coello [9]	(0.051480, 0.351661, 11.632201)	0.01271
Yang and Deb [41]	(0.051690, 0.356750, 11.28716)	0.012665
Present	(0.051690, 0.356750, 11.28716)	0.012665

Table 4 Comparison of optimal solutions for pressure vessel design

Author	Optimal solution	Best objective
Cagnina et al. [6]	(0.8125, 0.4375, 42.0984, 176.6366)	6059.714
Coello [9]	(0.8125, 0.4375, 42.3239, 200.0)	6288.7445
Yang et al. [47]	(0.8125, 0.4375, 42.0984456, 176.6365959)	6059.714
Present	(0.8125, 0.4375, 42.0984456, 176.6366)	6059.714

Pressure Vessel Design

A well-known design benchmark is the pressure vessel design problem that has been used by many researchers. This design is to minimize the overall cost of a cylindrical vessel subject to stress and volume requirements. There are four design variables: the thickness d_1 and d_2 for the head and body, respectively, the inner radius r, and the length W of the cylindrical section [6, 9]. The main objective is

$$\min f(\mathbf{x}) = 06224rWd_1 + 1.7781r^2d_2 + 19.64rd_1^2 \\ + 3.1661Wd_1^2, \qquad (15)$$

subject to four constraints:

$$g_1(\mathbf{x}) = -d_1 + 0.0193r \le 0, \quad g_2(\mathbf{x}) = -d_2 + 0.00954r \le 0, \qquad (16)$$

$$g_3(\mathbf{x}) = -\frac{4\pi r^3}{3} - \pi r^2 W - 1296000 \le 0, \\ g_4(\mathbf{x}) = W - 240 \le 0. \qquad (17)$$

The inner radius and length are limited to $10.0 \le r, W \le 200.0$. However, the thickness d_1 and d_2 can only be the integer multiples of a basic thickness of 0.0625 in. Thus, the simple bounds for thickness are

$$1 \times 0.0625 \le d_1, d_2 \le 99 \times 0.0625. \qquad (18)$$

With these additional constraints, this optimization problem becomes a mixed integer programming because two variables are discrete and the other two variables are continuous.

Using the same parameter settings as before, the results of 20 independent runs are summarized and compared in Table 4. In fact, all these algorithms can find the global optimal solution as found by Yang et al. [47].

Speed Reducer Design

The speed reducer design is an engineering design benchmark with seven design variables. These design variables include the face width of the gear, number of teeth, and diameter of the shaft and others [18]. All these variables can take continuous values, except for x_3 which is an integer.

The objective to minimize the cost function

$$f(\mathbf{x}) = 0.7854 \left[x_1 x_2^2 (3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ + (x_4 x_6^2 + x_5 x_7^2) \right] \\ - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3), \qquad (19)$$

subject to 11 constraints:

$$g_1(\mathbf{x}) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0, \quad g_2(\mathbf{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0, \qquad (20)$$

$$g_3(\mathbf{x}) = \frac{1.93x_4^3}{x_2 x_3 x_6^4} - 1 \le 0, \quad g_4(\mathbf{x}) = \frac{1.93x_5^3}{x_2 x_3 x_7^4} - 1 \le 0, \qquad (21)$$

$$g_5(\mathbf{x}) = \frac{1.0}{110x_6^3} \sqrt{(\frac{745.0x_4}{x_2 x_3})^2 + 16.9 \times 10^6} - 1 \le 0, \qquad (22)$$

$$g_6(\mathbf{x}) = \frac{1.0}{85x_7^3} \sqrt{(\frac{745.0x_5}{x_2 x_3})^2 + 157.5 \times 10^6} - 1 \le 0, \qquad (23)$$

$$g_7(\mathbf{x}) = x_2 x_3 - 40 \le 0, \quad g_8(\mathbf{x}) = 5x_2 - x_1 \le 0, \qquad (24)$$

$$g_9(\mathbf{x}) = x_1 - 12x_2 \le 0, \quad g_{10}(\mathbf{x}) = (1.5x_6 + 1.9) - x_4 \le 0, \qquad (25)$$

$$g_{11}(\mathbf{x}) = (1.1x_7 + 1.9) - x_5 \le 0. \qquad (26)$$

In addition, the simple bounds for the variables are $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$ (integers only), $7.3 \le x_4 \le 8.3$, $7.8 \le x_5 \le 8.4$, $2.9 \le x_6 \le 3.9$, and $5.0 \le x_7 \le 5.5$.

Table 5 Comparison of optimal solutions for the speed reducer problem

Author	Optimal solution	Best objective
Akhtar et al. [1]	(3.5061, 0.7, 17, 7.549, 7.8593, 3.3656,5.28977)	3008.08
Cagnita et al. [6]	(3.5, 0.7, 17, 7.3, 7.8, 3.350214, 5.286683)	2996.348165
Yang and Gandomi [44]	(3.5, 0.7, 17, 7.3, 7.71532, 3.35021, 5.2875	2994.467
Present	(3.5, 0.7, 17, 7.3, 7.8, 3.34336449, 5.285351)	2993.749589

Table 6 Measured response of a simple vibration system.

t	0.00	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00
$y_d(t)$	0.00	0.59	1.62	2.21	1.89	0.69	-0.99	-2.53	-3.36	-3.15	-1.92

The results of 20 independent runs are summarized in Table 5 where comparison has also been made. As we can see, MSCS obtained the best result. Since there is no literature about the analysis of this problem, no one knows what the global best solution should be. Thus, we can only say that 2993.749589 is the best result achieved so far.

Parameter Identification of Vibrations

For a simple vibration problem, two unknown parameters μ and ν are estimated from the measurements of vibration amplitudes. The governing equation is

$$\frac{d^2 y(t)}{dt^2} + \mu \frac{dy(t)}{dt} + \nu y(t) = 40\cos(3t). \tag{27}$$

This is a damped harmonic motion problem and its general solution can be quite complex [48]. However, for fixed $\mu = 4$ and $\nu = 5$ with initial values of $y(0) = 0$ and $y'(0) = 0$, its analytical solution can be simplified as

$$y(t) = e^{-2t}[\cos(t) - 7\sin(t)] + 3\sin(3t) - \cos(3t). \tag{28}$$

For a real system with a forcing term $40\cos(3t)$, we do not know the parameters, but its vibrations can be measured. For example, in an experiment, there are $N = 11$ measurements as shown in Table 6.

The task is to estimate the values of the two parameters. However, one of the challenges is that the calculation of the objective function that is defined as the sum of errors squared. That is

$$f(\mathbf{x}) = \sum_{i=1}^{N}(y_{i,\text{predicted}} - y_{i,d})^2, \tag{29}$$

where the predicted $y(t)$ has to be obtained by solving the second-order ordinary differential (27) numerically and iteratively for every given set of μ and ν. This becomes a series of optimization problems.

Using the MSCS with a population of 40 cuckoos and the same parameter setting given in the "Parameter Settings" section. We have run the algorithm for 20 times so as to

obtain meaningful statistics. The mean values of the two parameters obtained from the measured data are $\mu_* = 4.025$ and $\nu_* = 4.981$, which are very close to the true values of $\mu = 4.000$ and $\nu = 5.000$.

Iris Classification

To test the MSCS algorithm even further, we use it to solve the classification problem of the well-known Fisher's Iris flower data set. This data set has 150 data points or instances with 4 attributes and 3 distinct classes [14]. We use the data from the UCI Machine Learning Repository.[1]

We have encoded the centers of clusters as solution vectors so as to minimize the overall intra-clustering distances. Although the parameter settings are the same as before, the number of iterations is limited to 100 so as to be comparable with the results from the literature [4]. The best values obtained are compared with the results obtained by other methods such as the hybrid k-means and PSO approach [19], multiple kernel-based fuzzy c-means with cuckoo search [4], and k-means with improved feature based cluster center initialization algorithm (CCIA) [20].

The optimization results are summarized in Table 7. As we can clearly see, MSCS obtained the best result that is an improvement over the best results obtained by multiple kernel fuzzy c-means based cuckoo search approach (MKF-cuckoo) [4].

The results and simulation we have obtained so far are indeed encouraging. Obviously, we will carry out more thorough evaluations of the proposed approach in the future work. So let us summarize the work we have achieved so far in this paper.

Discussions

The original cuckoo search has been extended to capture more realistic characteristics of cuckoo-host co-evolution systems. We have developed a multi-species cuckoo search for solving global optimization problems, based on the main characteristics of multiple cuckoo species competing and co-evolving with host species. Then, we have tested the proposed approach using a set of 15 function benchmarks to show that the proposed algorithm can indeed work well. Preliminary results suggest that MSCS can have a higher convergence rate and obtain better results in general. In addition, we have used five different case studies

Table 7 Accuracy comparison for Iris data set

Method	Author	Accuracy (%)
K-means and PSO	Kao et al. [19]	89.3
MKF-Cuckoo	Binu et al. [4]	95.0
K-means	Khan and Ahmad [20]	76.4
K-means with CCIA	Khan and Ahmad [20]	88.7
MSCS	this paper	97.1

---

[1] http://archive.ics.uci.edu/ml/datasets/Iris

from engineering designs, parameter estimation and data clustering to further test the proposed algorithm. Our simulation results and subsequent comparison have shown that the MSCS can indeed find the optimal solutions that are either better and comparable with the results obtained by other methods.

The essence of multi-species co-evolution is to use more than one species so as to see how different species interact and compete. In the present studies, we have just used two cuckoo species in the cuckoo-host co-evolution. Future work will focus on more than two cuckoo species and more detailed parametric studies with varied population sizes. In addition, it would be useful to gain more insight by tuning the key parameters in the algorithm to see how they may affect the overall performance of the algorithm.

Furthermore, in the real-world cuckoo-host co-evolution systems, there are multiple cuckoo species interacting with multiple host species, which can have much more complex behaviour and characteristics. For example, the common cuckoos can lay eggs in many different host species including garden warblers and reed warblers [27]. In reality, the number of eggs laid by cuckoos and inside nests can be random. The current approach with the preliminary tests consists of only a single host species with multiple cuckoo species. Therefore, a possible extension can include multiple host bird species compete and co-evolve with multiple cuckoo species. In addition, it can also be useful to carry out further tests of this algorithm using a more extensive set of benchmarks and real-world case studies.

Compliance with Ethical Standards This research does not involve any use of animals and the research is in compliance with the ethical standards of the Journal.

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Akhtar S, Tai K, Tay T. A socio-behavioural simulation model for engineering design optimization. Eng Optim. 2002;34(4):341–454.
2. Arora JS. Introduction to optimum design. New York: McGraw-Hill; 1989.
3. Bhargava V, Fateen SEK, Bonilla-Petriciolet A. Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations. Fluid Phase Equilib. 2013;337:191–200.
4. Binu D, Selvi M, Aloysius G. MKF-Cuckoo: hyrbidization of cuckoo search and multiple kernel-based fuzzy c-means algorithm. AASRI Procedia. 2013;4:243–9.
5. Blackwell T, Branke J. Multi-swarm optimization in dynamic environments. In: Applications of evolutionary computing, evoworkshops 2004, lecture notes in computer science. Berlin: Springer; 2004. p. 489–500.
6. Cagnina LC, Esquivel SC, Coello Coello CA. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica. 2008;32:319–26.
7. Chandrasekaran K, Simon SP. Multi-objective scheduling problem: hybrid appraoch using fuzzy assisted cuckoo search algorithm. Swarm and Evolutionary Comput. 2012;5(1):1–16.
8. Chen Q, Liu B, Zhangx Q, Suganthan PN, Qu BY. Problem definition and evaluation criteria for CEC2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization, Technical Report, Commputational Intelligence Laboratory, Zhengzhou University, China and Technical Report. Singapore: Nanyang Technology Univesity; 2014.
9. Coello Coello CA. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind. 2000;41:113–27.
10. Davies NB, Brooke ML. Co-evolution of the cuckoo and its hosts. Sci Am. 1991;264(1):92–8.
11. Davies NB. Cuckoo adaptations: trickery and tuning. J Zool. 2011;284(1):1–14.
12. Dhivya M, Sundarambal M. Cuckoo search for data gathering in wireless sensor networks. Int J Mobile Commun. 2011;9(4):642–56.
13. Dubey HM, Pandit M, Panigrahi BK. A biologically inspired modified flower pollination algorithm for solving dispatch problems in modern power systems. Cogn Comput. 2015;7(5):594–608.
14. Duda RO, Hart PE. Pattern classification and scene analysis. New York: Wiley; 1973.
15. Durgun I, Yildiz AR. Structural design optimization of vehicle components using cuckoo search algorithm. Mater Test. 2012;3(3):185–8.
16. Fister IJr, Fister D, Fister I. A comprehensie review of cuckoo search: variants and hybrids. Int J Math Numer Optim. 2013;4(4):387–409.
17. Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput. 2013;29(1):17–35.
18. Golinski J. An adaptive optimization system applied to machine synthesis. Mech Mach Theory. 1973;8(4):419–36.
19. Kao Y-T, Zahara E, Kao I-W. A hybridized approach to data clustering. Expert Syst Appl. 2008;34(3):1754–62.
20. Khan SS, Ahmad A. Cluster center initialization algorithm for k-means clustering. Pattern Recogn Lett. 2004;25(11):1393–1302.
21. Krüger O, Sorenson MD, Davies NB. Does co-evolution promote species richness in parasitic cuckoos. Proc Roy Soc B. 2009;276(1674):3871–9.
22. Mishra SK. Global optimization of some difficult benchmark functions by host-parasite co-evolutionary algorithm. Econ Bull. 2013;33(1):1–18.
23. Mlakar U, Fister IJr, Fister I. Hybrid self-adaptie cuckoo search for global optimization. Swarm Evol Comput. 2016;29:47–72.
24. Mohamad AB, Zain AM, Bazin NEN. Cuckoo search algorithm for optimization problems—a literature review and its applications. Appl Artif Intell. 2014;28(5):419–48.
25. Moravej Z, Akhlaghi A. A novel approach based on cuckoo search for DG allocation in distribution network. Electr Power Energy Syst. 2013;44(1):672–9.
26. Pare S, Kumar A, Bajaj V, Singh GK. A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve. Appl Soft Comput. 2016;47:76–102.
27. Payne RB. The cuckoos. Oxford: Oxford University Press; 2005.
28. Pavlyukevich I. Lévy flights, non-local search and simulated annealing. J Comput Phys. 2007;226(2):1830–44.

29. Pereira LAM, Rodrigues D, Almeida TNS, Ramos CCO, Souza AN, Yang XS, Papa JP. A binary cuckoo search and its application for feature selection. In: Cuckoo Search and Firefly Algorithm. Studies in Computational Intelligence; 2013. p. 141–154.

30. Qu BY, Liang JJ, Wang ZY, Chen Q, Suganthan PN. Novel benchmark functions for continuous multimodal optimization with comparative results. Swarm Evol Comput. 2016;26(1):23–34.

31. Santos CAG, Freire PKMM, Mishra SK. Cuckoo search via lévy fligths for optimization of a physically-based runoff-erosion model. J Urban Environ Eng. 2012;6(2):123–31.

32. Shehab M, Khader AT, Al-Betar MA. A survey on applications and variants of the cuckoo search algorithm. Appl Soft Comput. 2017. https://doi.org/10.1016/j.asoc.2017.02.034.

33. Siddique N, Adeli H. Nature-inspired chemical reaction optimisation algorithms. Cogn Comput. 2017;9:411–22.

34. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, Technical Report of Nanyang Technological University, Singapore and kanGAL Report, IIT Kanpur, India. 2005.

35. Valian E, Mohanna S, Tavakoli S. Improved cuckoo search algorithm for feedforward neural network training. Int J Articial Intell Appl. 2011;2(3):36–43.

36. Walton S, Hassan O, Morgan K, Brown MR. Modified cuckoo search: a new gradient free optimization algorithm. Chaos, Solitons Fractals. 2011;44(9):710–8.

37. Wang GG, Deb S, Gandomi AH, Zhang ZJ, Alavi AH. Chaotic cuckoo search. Soft Comput. 2016;20(9):3349–62.

38. Wong PK, Wong KI, Vong CM, Cheung CS. Modeling and optimization of biodiesel energy performance using kernel-based extreme learning machine and cuckoo search. Renew Energy. 2015;74:640–7.

39. Woźniak M, Polap D, Napoli C, Tramontana E. Graphic object feature extraction system based on cuckoo search algorithm. Expert Syst Appl. 2016;66:20–31.

40. Wu TQ, Yao M, Yang JH. Dophin swarm extreme learning machine. Cogn Comput. 2017;9(2):275–84.

41. Yang XS, Deb S. Cuckoo search via lévy flights. In: Proceedings of world congress on nature & biologically inspired computing (NaBic 2009), India. USA: IEEE Publications; 2009. p. 210–214.

42. Yang XS, Deb S. Engineering optimization by cuckoo search. Int J Math Model Num Optim. 2010;1(4):330–43.

43. Yang XS, Deb S. Cuckoo search: recent advances and applications. Neural Comput Appl. 2014;24(1):169–74.

44. Yang XS, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Eng Comput. 2012;29(5):464–83.

45. Yang XS. Cuckoo search and firefly algorithm: theory and applications. Studies in computational intelligence, vol. 516. Berlin: Springer; 2014.

46. Yang XS, Deb S. Multiobjective cuckoo search for design optimization. Comput Oper Res. 2013;40(6):1616–24.

47. Yang XS, Huyck C, Karamanoglu M, Khan N. True global optimality of the pressure vessel design problem: a benchmark for bio-inspired optimisation algorithms. Int J Bio-Inspired Comput. 2013;5(6):329–35.

48. Yang XS. Engineering mathematics with examples and applications. London: Academic Press; 2017.

49. Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput. 1999;3(2):82–102.

50. Yildiz AR. Cuckoo search algorithm for the selection of optimal machine parameters in milling operations. Int J Adv Manuf Technol. 2013;64(1):55–61.

51. Zamani AA, Tavakoli S, Etedali S. Fractional order PID control design for semi-active control of smart base-isolated structures: a multi-objective cuckoo search approach. ISA Tractions. 2017;67:222–32.

52. Zheng HQ, Zhou Y. A novel cuckoo search optimization algorithm based on Gauss distribution. J Comput Inform Syst. 2012;8(10):4193–200.

53. Zineddube M. Vulnerabilities and mitigation techniques toning in the cloud: a cost and vulnerablities coverage optimization approach using cuckoo search algorithm with lévy flights. Comput Secur. 2015;48:1–18.