Applied Soft Computing 61 (2017) 149–173

Contents lists available at ScienceDirect

Applied Soft Computing

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

Review article

Bio-inspired computation: Recent development on the modifications of the cuckoo search algorithm

Haruna Chiroma[a,*], Tutut Herawan[b], Iztok Fister Jr.[c], Iztok Fister[c], Sameem Abdulkareem[b], Liyana Shuib[b], Mukhtar Fatihu Hamza[d], Younes Saadi[b], Adamu Abubakar[e]

[a] Federal College of Education (Technical), Department of Computer Science, Gombe, Nigeria
[b] University of Malaya, Faculty of Computer Science and IT, Kuala Lumpur, Malaysia
[c] University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova, Maribor, Slovenia
[d] Bayero University Kano, Faculty of Engineering, Kano, Nigeria
[e] International Islamic University, Malaysia

ARTICLE INFO

Article history:
Received 11 February 2017
Received in revised form 3 June 2017
Accepted 24 July 2017
Available online 5 August 2017

Keywords:
Cuckoo search algorithm
Convergence
Swarm intelligence
Nature-inspired algorithms
Optimization metaheuristics

ABSTRACT

Presently, the Cuckoo Search algorithm is attracting unprecedented attention from the research community and applications of the algorithm are expected to increase in number rapidly in the future. The purpose of this study is to assist potential developers in selecting the most suitable cuckoo search variant, provide proper guidance in future modifications and ease the selection of the optimal cuckoo search parameters. Several researchers have attempted to apply several modifications to the original cuckoo search algorithm in order to advance its effectiveness. This paper reviews the recent advances of these modifications made to the original cuckoo search by analyzing recent published papers tackling this subject. Additionally, the influences of various parameter settings regarding cuckoo search are taken into account in order to provide their optimal settings for specific problem classes. In order to estimate the qualities of the modifications, the percentage improvements made by the modified cuckoo search over the original cuckoo search for some selected reviews studies are computed. It is found that the population reduction and usage of biased random walk are the most frequently used modifications. This study can be used by both expert and novice researchers for outlining directions for future development, and to find the best modifications, together with the corresponding optimal setting of parameters for specific problems. The review can also serve as a benchmark for further modifications of the original cuckoo search.

© 2017 Elsevier B.V. All rights reserved.

Contents

1. Introduction 150
2. Rudimentary of cuckoo search algorithm 151
   2.1. Origin of the cuckoo search algorithm 151
   2.2. The cuckoo search algorithm 151
3. Modified cuckoo search algorithm 152
4. Methodology 152
   4.1. Keywords search 152
   4.2. Search strategies 153
   4.3. Searched databases 153
   4.4. Paper selection criteria 153

* Corresponding author.
E-mail addresses: freedonchi@yahoo.com, freedonchi@gmail.com (H. Chiroma), tutut@um.edu.my (T. Herawan), iztok.fister2@uni-mb.si (I. Fister Jr.), iztok.fister@uni-mb.si (I. Fister), sameem@um.edu.my (S. Abdulkareem), liyanashuib@um.edu.my (L. Shuib), younessaadi@gmail.com (Y. Saadi).

http://dx.doi.org/10.1016/j.asoc.2017.07.053
1568-4946/© 2017 Elsevier B.V. All rights reserved.

4.5. Extraction of data ... 153
5. Modifications made to the cuckoo search algorithm ... 154
5.1. Continuous cuckoo search algorithm ... 154
5.1.1. Parameter control ... 154
5.1.2. Hybridization in cuckoo search algorithm ... 156
5.1.3. Multi-objectives cuckoo search algorithm ... 158
5.2. Binary cuckoo search algorithm ... 159
5.3. The most frequently used modified cuckoo search algorithm ... 160
5.4. Applications of the cuckoo search ... 160
6. Performance analysis of CS ... 161
6.1. The benchmark function suites ... 161
6.2. The most widely used parameter control ... 161
6.3. The most widely used performance measures ... 163
6.4. Performance study ... 165
6.5. Trends in the modifications to the cuckoo search algorithm ... 167
7. Open research Problems/Challenges ... 167
8. Future research directions ... 169
9. Conclusions ... 169
Acknowledgement ... 170
References ... 170

1. Introduction

Nature-inspired metaheuristic algorithms have been accepted widely throughout optimization, machine learning, data mining, computational intelligence and artificial intelligence over the last two decades [1,2]. The Cuckoo Search algorithm (CS) is one of the more promising nature-inspired algorithms [3], developed by Yang and Deb [4,5]. The CS presently attracts unprecedented attention from researchers using current metaheuristic algorithms for solving optimization problems. Studies on CS within the literature are growing quickly with applications covering various areas [6]. More recently, CS has been applied in several areas with promising performances [3], e.g., forecasting [7], classifications [8,9], power systems [10], feature selection [11], heat exchangers [12], etc. Recent studies have proven the effectiveness of CS over other metaheuristic algorithms [13,14]. The success recorded by the CS is attributed to its striking balance between local and global searching, in addition to the fewer number of parameters required by the CS for execution. The global search of CS uses Lévy flights non-standard random distribution with infinite mean and variance. Therefore, exploration of a search space by CS is closer to the nature, where resources are distributed non-uniformly. Hence, a global convergence can be better guaranteed [3]. Yang and Deb [6] stated that *"There is no doubt more applications of Cuckoo Search will be seen in the expanding literature in the coming years"*.

There are several reviews of CS applications in solving optimization problems. For example, Xing et al. [15] and Fister et al. [16,17] reviewed the applications of CS within engineering, software implementation, software process improvement, milling operations, structural optimization, scheduling, learning algorithm improvement, and multi-objective unit commitment. An overview of the CS applications, as well as detailed descriptions of CS with Lévy flight, was covered excellently by Yang and Deb [6] that captured domains as follows: Training Artificial Neural Networks (ANN), web services, embedded systems, optimization of machine parameters, designing of steel frames, wireless sensor networks and structural optimization. Walton et al. [18] presented a review of attempts made by the research community to improve the performances of CS (only two studies were considered) and hybridization of CS with other optimization algorithms. The review included some selected applications of CS in machine learning and design. An additional review of the applications of CS can be found in [3]. A review of the optimization of ANN weights, parameters of the Support Vector Machine (SVM) and radial basis function, job scheduling, finding cluster heads within wireless sensor networks, locating of the shortest path and clustering was presented by Kamat and Karegowda in [19].

However, searching for the optimal values of CS parameters has not been considered in previous reviews, although these settings play a crucial role in the performance of the CS [6]. Comparing performances among different CS modifications are worth studying [18], in view of the fact that the CS has witnessed a significant improvement from its inception, as shown by Fister et al. [16,17]. Unfortunately, comparisons between the performances of several CS modifications have been scarce in previous reviews. Therefore, this paper seeks answers to the following questions:

a Can performance improvement made on CS modification in different studies be compared?
b What are the modifications performed on CS since inception?
c What are the optimal parameter settings for the application of CS?
d What are the open questions that require answers for improving CS?

Answers to these questions could be beneficial to the research community during further modifications of CS for performance improvement. In fact, these answers could assist expert researchers by developing a new modification of CS and represent a starting point for novice researchers in searching for the proper modification of CS. In this paper. the abbreviations CS and MoCS represent the original cuckoo search algorithm before modification and the modified version of the cuckoo search algorithm, respectively.

The remainder of this paper is organized as follows: Section 2 presents the basic theory of CS. Section 3 discusses a modified CS. Section 4 presents the methods used for collecting the literature. Section 5 provides a review of the modifications of CS. Section 6 discusses the performance analysis of the studies and trends of CS modifications. Section 7 discusses the problem/challenges of the CS modification. Section 8 presents a promising future research direction before the concluding remarks in Section 9.

The summary of the contributions of the review paper are as follows:

i The comprehensive recent progress on the modifications of the cuckoo search algorithms are summarized and presented.
ii The estimated qualities of the modifications and the percentage improvements made by the modified cuckoo search over

the original cuckoo search for some selected reviews studies are computed.

iii It is found that the population reduction and usage of biased random walk are the most frequently used modifications.

iv The challenges of the cuckoo search algorithms are uncovered in the paper for easy identification by researchers.

v Future research opportunities for the enhancement of the cuckoo search algorithm are unveiled.

2. Rudimentary of cuckoo search algorithm

The origin and basic theoretical background of CS are described in this section, specifically, the origin of the algorithm, common characteristics, representation of solutions, the generation of new solutions, evaluation function, replacement and termination condition.

2.1. Origin of the cuckoo search algorithm

The beautiful sounds and aggressive reproduction strategy of a cuckoo makes it a fascinating bird. Within a broader view, the cuckoo family of birds are divided in three major species, i.e., Musophagidae, Cuculidae and Opisthocomidae. Cuckoos can be found across the globe. There are over 100 different species worldwide, whereas only Cuculidae can be found in Europe. Typically, cuckoos feed on caterpillars. Interestingly, the Ani and Guira cuckoo species can lay eggs in a communal nest. In order to increase their hatching probability, these two cuckoo species remove eggs from other nests. Some other cuckoo species engage the obligate brood parasitism by laying their eggs in the nests of other host birds. It has been estimated that half of the cuckoos don't hatch their own eggs but rather resort to parasitism. The aggressive reproduction strategies of the cuckoo's breeding parasitism are summarized as follows: Typically, a female cuckoo lays between 6 and 22 eggs with color matching the host eggs. In nature, different species produce different colored eggs. The size of the eggs varies within 16.3 mm–21.9 mm. A female cuckoo hides close to an appropriate host nest in order to use the opportunity by dumping her eggs in the host nest. Typically, the new host birds are evicted aggressively by the new born cuckoos [20,21]. The three types of breeding cuckoo parasitism strategies are as follows: (1) Intraspecific breeding parasitism, (2) Cooperative breeding, and (3) Nest takeover. Direct conflict between the host birds and cuckoos is however possible. The host birds either abandon the nest or throw the alien eggs out of the nest to produce new eggs.

2.2. The cuckoo search algorithm

The behavior of the cuckoo, as described in Subsection 2.1, is simulated during CS optimization to provide better performances than other distribution-based random walks when exploring large scale search spaces. The three major ideas of a cuckoo's behavior are proposed by Yang and Deb [4] for rules governing the CS optimization algorithm as follows: (1) Each cuckoo lays one egg at a time and dumps it into a randomly-chosen nest. (2) Nests with the eggs of optimal quality will move to the next generation. (3) The available host nest is fixed and the egg laid by a cuckoo is discovered by the host bird with the probability of the worst nests being abandoned ($p_a \in [0,1]$). In the CS algorithm, the quality or fitness of a solution is modeled proportional to the objective function value. Without difficulties, getting an optimized solution to a complete problem using CS does not require a comprehensive search. Each individual solution corresponding the host *nest* represents the position of the cuckoo's egg in the CS algorithm. Generally, the position is defined as:

$$\mathbf{x}_i^{(t)} = \left\{ x_{i,j}^{(t)} \right\}, \quad \text{for } i = 1 \ldots n \quad \text{and} \quad J = 1 \ldots D, \tag{1}$$

where t denotes the current generation, n the number of host nests within the population and D the dimensionality of the problem. In the CS algorithm, the initial value of the *j-th* attributes in the *i-th* particle $x_{i,j}^{(0)}$ is determined typically as follows:

$$x_{i,j}^{(0)} = U(0,1).(u_j - l_j) + l_j, \tag{2}$$

where $U(0, 1)$ determines the random value drawn from the uniform distribution within the interval [0, 1], u_j and l_j are the upper and lower boundaries of the *j-th* attributes, respectively. In each generation, the CS algorithm controls these boundaries as follows: If the value of the attributes underflows, this value is updated with the corresponding lower bound, while, if this value overflows, it is updated with the corresponding upper bound. The general form of the CS algorithm, which is based on a local random-walk appropriate for exploitation of the search space through Lévy flights, is described as follows:

$$\mathrm{x}_i^{(t+1)} \quad \mathrm{x}_i^{(t)} + \alpha \oplus \mathrm{Levy}(s, \lambda), \quad s >> s_o > 0, \tag{3}$$

where $t \in [1, MaxGeneration]$ is the number of the current generation, the *MaxGeneration* is the predetermined number of maximum generations, α is the *step size scale factor*, s the *step size*, λ the *standard deviation*, and Lévy flight is expressed as follows:

$$\mathrm{Levy}(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}. \tag{4}$$

Gamma function $\Gamma(\lambda)$ in Eq. (4) is defined as $\Gamma(\lambda) = \int_0^\infty t^{\lambda-1} e^{-t} dt$, and represents an extension of the factorial function. In a special case, when λ is a positive integer, it holds $\Gamma(\lambda) = (\lambda - 1)!$. In nature, the flight movements of many animals and insects are recognized as being random. Commonly, the foraging path of animals has a characteristic that their next move is based on the current state and the variation probability of the next state. Anyway, it chooses the path to be determined indirectly on a probability which can be modeled mathematically. Lévy flight is a random distribution that enables a series of straight jumps chosen from a heavy-tailed probability density function [22]. In statistical terms, when the distribution is used within a stochastic algorithm for global optimization, like CS, this corresponding algorithm is capable of finding the global optimum [23]. The Lévy flight process step size can be calculated as follows:

$$s = \frac{u}{|v|^{\frac{1}{\beta}}}, \tag{5}$$

where u and v are two random values drawn from normal distribution with mean zero and their standard deviations σ_u^2 and σ_v^2 as follows:

$$u \sim N\left(0, \sigma_u{}^2\right), v \sim N\left(0, \sigma_v{}^2\right), \quad \text{and} \tag{6}$$

$$\sigma_u = \left\{ \frac{\Gamma\left(1+\beta\right) \sin\left(\tau\beta/2\right)}{\Gamma\left[\left(1+\beta\right)/2\right] \beta 2 \left(\beta - 1\right)^2} \right\}^{\frac{1}{\beta}}, \sigma_v = 1. \tag{7}$$

The original CS is capable of interleaving the local search (according to Eq. (2)) with the global search (according to Eq. (3)). This interleaving is controlled by a switching parameter p_a which determines a fraction of the worst nests needed to be abandoned and new ones to be built in order to discovering new, probably more promising regions of the search space.

In summary, all the mentioned features of the CS are assembled within a pseudo-code of the original CS algorithm which is illustrated in Fig. 1 [4].

```
begin
    Objective function f(x_i), population of nests x_i = (x_{i,1},...,x_{i,D})^T;
    Initialize nest (x_i : for i = 1,...,n);
    while (Termination condition not met) or
        for i=1 to NP do
            u_i = Generate new solution (x_i);
            f_trial = Evaluate the new solution (u_i);
            j = ⌊U(0,1)·n+1⌋;
            if f_trial < f_j then
                x_j = u_i;  f_j = f_trial;
            endif
            if U(0,1) < p_a then
                Initialize nest (x_worst);
            endif
            if f_trial < f_min then
                x_best = u_i;  f_min = f_trial;
            endif
        endfor
    endwhile
    Postprocess results and visualization;
end
```

Fig. 1. Pseudo-code of the original CS.

Some of well-known CS algorithm modifications, that are treated wrongly as the original one, use an explicit balancing among the exploration/exploitation components of the search process via the switching parameter p_a. This balancing is expressed mathematically as

$$\mathbf{x}_t^{(t+1)} = \begin{cases} U(0,1).\left(\mathrm{u_i} - l_j\right) l_j, & U(0,1). \le p_a, \\ \mathbf{x}_t^{(t)} + \alpha \oplus Levy\left(\mathrm{s}, \lambda\right), & \text{otherwise.} \end{cases} \tag{8}$$

where the global random search representing the exploration component is performed only when $U(0,1) \le p_a$. The selection of better solutions in CS can be expressed as

$$\mathbf{x}_{best}^{(t+1)} = \begin{cases} \mathbf{x}_i^{(t+1)}, & \text{if} f\left(\mathbf{x}_i^{(t+1)}\right) \le f\left(\mathbf{x}_{best}^{(t)}\right), \\ \mathbf{x}_{best}^{(t)}, & \text{otherwise.} \end{cases} \tag{9}$$

Let us notice that the majority of Matlab implementations of the original CS available online is a modification of the original CS algorithm according to Eq. (8). As such, many researchers may apply the modified CS based on Eq. (8) code assuming that they are using the original CS. This is pointed out by Walton et al. [24]. To take care of this misconception in our study, we ensured that each study selected for the review must explain clearly exactly what is modified in the CS. Studies that did not vividly describe what is modified in the CS are excluded in our review. The modification of CS in this paper refers to modifications made on CS, not the modification according to Eq. (8). In summary, the original CS demands the settings of three parameters: The population size n, the step-size scaling factor α, and the switching parameter p_a. However, the more critical parameters required for obtaining the optimal solution are parameters p_a and α [25].

3. Modified cuckoo search algorithm

Researchers within the optimization community have attempted to modify the original CS algorithm in order to improve its performance. However, these modifications can be seen from almost two points of view: (1) What has been modified and (2) How those modifications were performed. From the first point of view (Fig. 2), the CS algorithm can be modified mainly on the following components of the CS algorithm: Representation of solutions, generation of the new solutions, evaluation function and replacement operators. The CS algorithms are divided into two broad classes regarding the representation: Continuous CS and binary CS (BCS).

From the second point of view (Fig. 3), the original CS can be divided according to the kinds of parameter control and hybridization [12]. Parameters can be controlled, deterministically, adaptively and self-adaptively. Hybridization means incorporating the problem-specific knowledge into the original optimization algorithm. The more suitable hybrid methods applied to the original CS algorithm are: Constructive heuristics, local search and hybridization with algorithm structures borrowed from the other metaheuristic algorithms (e.g., GA, Differential Evolution (DE) strategies, etc.). Readers are invited to refer to the paper of Fister et al. [16,17] for a detailed review of other metaheuristic algorithms.

Heuristic algorithms cannot be classified easily into one of the proposed classes, because they can borrow the characteristics of both classes, in general. These kinds of heuristic algorithms are also known under the name metaheuristics.

Table 1
Online academic databases used for searching the literature.

Academic online database	Articles Returned
ACM Digital Library	55
Emerald	0
DBLP Computer Science Bibliography	29
IEEE Xplore™	16
SAGE Journals	0
ScienceDirect®	149
Scopus™	263
SpringerLink	507
Taylor & Francis Online	27
Web of Science	650

4. Methodology

This section describes the procedure used when searching for the relevant papers included within our review. The relevant papers were selected according to the following criteria during our research: Keywords search, search strategies, searched databases, paper selection criteria, and extraction of data.

4.1. Keywords search

In this study, the proposed keywords search is based on the objectives of this research. The keywords are used in queries conducted into the specific databases. This searching process was divided into two phases: Preliminary and final searches. After performing the preliminary keywords search, the results are evaluated. comprehensively Based on the results of this preliminary keywords search, the keywords are modified according to the demands of our study. In the final phase, a search for papers within various reputable academic databases was conducted using the following keywords: “Modified CS”, “improved CS”, “enhanced CS”, “review on modified CS”, etc. The related articles, journals, edited books chapters, and conference proceedings were identified from the following academic online databases (Table 1): IEEE Xplore, Association for Computing Machinery (ACM) Digital Library, Dblp, ScienceDirect, Scopus, Emerald, SAGE Journals, Taylor & Francis Online, Web of Science, Wiley Online Library and SpringerLink. Because the original CS algorithm was firstly proposed in 2009, only papers from the period covering 2009–2014 were considered during the study.

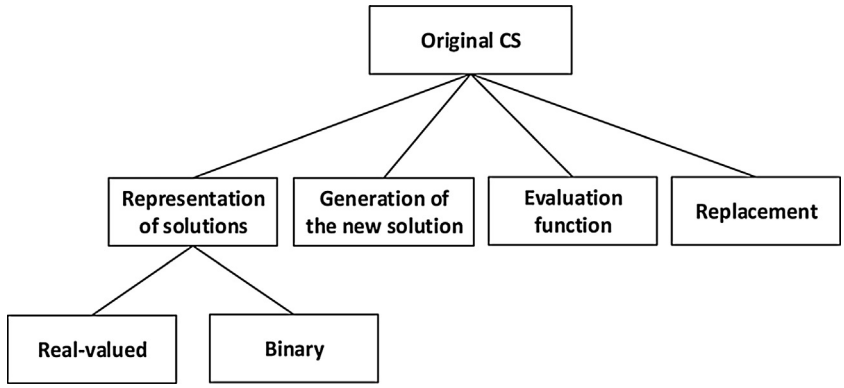

Fig. 2. What has been modified?.

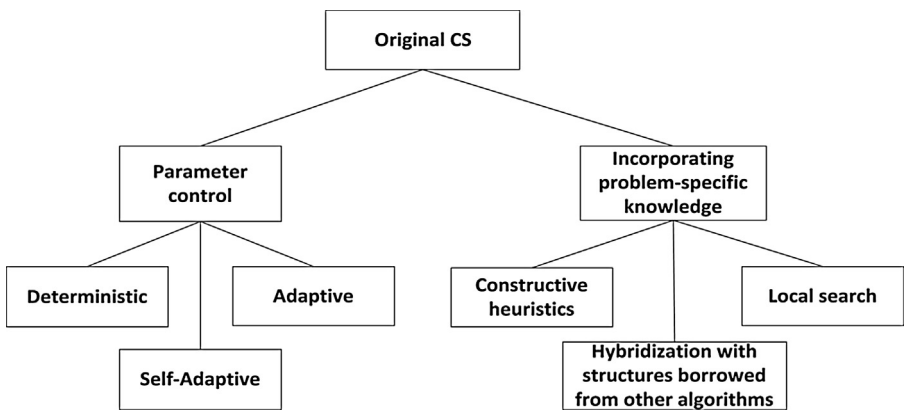

Fig. 3. How are the modifications performed?.

4.2. Search strategies

The search strategies are based on the work of Liberati et al. [26]. The references found after matching the search keywords are scanned in order to extract papers cited in the selected articles appropriate for this study. Among the online databases, some computer science-related databases were also taken into account, as presented in Table 1. On the other hand, the other observed databases are those more popular and prestigious online academic databases that provide international peer reviews of high quality articles [27]. The final searches were conducted between 12th July and 18th July of 2014.

4.3. Searched databases

The keywords search and search strategies described earlier in [26] were used to search for related studies in online databases, similar to the method applied by Aaltonen et al. [27]. The search keywords used during the searching are as follows: "Modified CS*", "enhanced CS*", "improved CS*", "upgraded CS*", "parallelized CS*", "hybridized CS*", "guided CS*", "review of modified CS*", etc, similar to how it was done in [27]. The searching was conducted in an online database, ScienceDirect, one of the more popular academic databases containing research articles in computer science, engineering, mathematics, social sciences, humanities, etc. and other online databases, as listed in Table 1. The Table presents the academic online databases with the corresponding number of articles returned after screening the databases.

4.4. Paper selection criteria

This stage involved selecting the procedure for discarding unsuitable articles. The screening started by sorting relevant titles related to the objective of the study. Similar criteria are used by Hartling et al. [28]. The articles that passed the screening criteria based on titles are scrutinized further based on Abstracts and Conclusions. The Conclusions, Abstracts and titles which scaled through the rigorous screening criteria were given to three independent researchers, who collated those articles that met the stated objective of our study rigorously. Those articles that did not meet the criteria were discarded. Finally, only those relevant articles which presented an experimental description of the modification of CS or implemented within MoCS were selected for the study.

4.5. Extraction of data

The aim of the extraction of data is to eliminate the redundant studies from the obtained CS studies. Data are extracted and entered into Tables created for the purpose of tabulating the extracted data. Similar to the study of Hartling et al. [28], two independent reviewers reviewed the extracted data. Additionally, some data were given to other reviewers for review. The reviews were later compared, and it was found that there are no significant discrepancies between the comparative results. In order to ensure integrity and to avoid any possibility of bias, two independent reviewers were enough to ensure integrity and to avoid any possibility of bias related to the observed data. Re-evaluation of the articles selected by the independent reviewers further eliminated a few more articles. In our study, the number of articles identified from 11 online databases are 1696 (Table 1) and 56 articles from the "search alert" function. As a result, a total of 1752 articles were identified. The number of discarded articles, perhaps of non-matching keywords in titles or identified as duplicate, was 780. In summary, 946 articles were excluded because of non-matching of the keywords in the titles, Abstracts or Conclusions. However, 26

(20 modified the CS and 6 implement modified CS) full-text articles passed the eligibility criteria for inclusion within the study. The initial search covered 12th July and 18th July of 2014 as stated earlier. Thus, we conducted another round of search within 9/14/2015 to 9/21/2015 to update the literature with newly published papers in this area of research. To update the search, another round of literature search was conducted between 25th November, 2016 to 1st December, 2016. We found 13 more articles within the current search period, in total, 76 articles were included.

5. Modifications made to the cuckoo search algorithm

The review was performed according to the classifications illustrated in Figs. 2 and 3. We divided all the analyzed papers into continuous CS and BCS. Both classes were then analyzed according to the used parameter control, hybridization and multi-objectives of the CS algorithm. Finally, the more often MoCS applications by researchers for solving problems in different domains were reviewed.

5.1. Continuous cuckoo search algorithm

The original CS, as described in Section 2, is considered as continuous because of the real-valued representation of solutions. This subsection describes the modifications made to the continuous CS.

5.1.1. Parameter control

Modifications of CS may have an impact on the partial component of the algorithm or the algorithm as a whole. In any metaheuristic algorithm, parameters play a significant role for the algorithm's performance. In line with this, researchers have modified the CS control parameters to enhance its robustness. For instance, Walton et al. [29] reduced scaling factor α by increasing the number of generations and evolved MoCS. In addition, exchange information due to an arithmetical crossover between the two solutions (parents) in the fraction of solutions with the best fitness were added in order to improve the convergence speed. The distance to the location of a new solution (offspring) is computed using the inverse of the golden ratio. This method provides offspring of more similarities to the better of the two parents. MoCS was applied for solving the optimization problem on 7 benchmark functions. Thus, it is found that the MoCS converged to the optimum solution faster than the original CS.

Rani et al. [30] improved the original CS by integrating a scaling factor α with inertia weight and, thus, made the algorithm adaptive. The roulette wheel operator is added to CS in order to improve its performance. The effectiveness of the MoCS is demonstrated on an array of geometric synthesis problems. The results show that the MoCS perform better than the CS according to accuracy and convergence speed. Zhang et al. [31] proposed MoCS because the CS does not explore its potentials completely. The following modifications have been performed on the CS: (1) Introduction of subgroup settings to enhance the survival rates of some substandard solutions by an inferior solution having a chance of becoming the optimal solution in the future. (2) Performing a parallel search with the cuckoos to improve the diversity of the population size n. (3) An incentive mechanism is introduced to improve the effectiveness of the search, hence, improved convergence speed. (4) In order to avoid reducing the efficiency of the groups' search, information is shared among the subgroups. (5) Adaptation: Large scaling factor α is used in the early stages of the search, while becoming smaller towards the end of the search by gradually changing the α during the search process to improve precision. The MoCS is evaluated on 9 benchmark functions and the best results were found within the sphere function showing that the MoCS advanced the performance of the CS. Table 2 summarizes the list of modifications made on CS parameter controls, whereas Table 3 presents the complete list of the continuous CS.

Table 2
Parameter control.

Topic	Reference
Adaptive α	[29,31–38]
Replace α with inertia weight	[30]
Adaptive n	[39,40]
Reconstructing of n	[41]
Removed α	[42]
Reduced control parameters	[43]
Replace α with chaotic sequence	[44]
Adaptive p_a and α	[45–47]
Sorting function computed α	[48]
Adaptive p_a	[49,50]
Only p_a required to be tuned	[51]
Variety of initial n	[52]
Replaced αwith 12 chaotic maps	[53]
Nawaz-Enscore-Ham (NEH) generated n	[54,55]
Adaptive the p_a and β	[56]
Auxilliary function	[57]
Learning regulator	[58]
Dynamic p_a, crossover & mutation	[59]
Self-Adaptive control parameters	[60]

Zheng and Zhou [39] integrated the annexing operator and cooperating operator in CS to enhance its diversity. The population size n in the MoCS was set as adaptive, and not fixed as in the CS. The performance of the MoCS is evaluated on 15 benchmark functions. Simulation analysis demonstrated that the MoCS perform better than the M-elite coevolutionary algorithm and Organizational evolutionary algorithms.

Lin and Lee [44] improved the quality of the neighborhood search in the original CS whenever new solutions are generated, to try as much as possible to prevent the solutions from being trapped in local minima. The proposed modification is applied as follows: The scaling factor α is replaced with the value extracted from a chaotic sequence $C_S^{(t)}$ in each generation. Thus, the chaotic sequence is expressed as follows:

$$C_S{}^{(t)} = 4 \times C_S{}^{(t-1)} \times (1 - C_S{}^{(t-1)}), \tag{10}$$

where t is the generation number. The proposed MoCS is implemented to solve the Lorenz function. The MoCS is not compared with other methods for evaluation purposes. Li et al. [43] proposed an alternative searching approach by integrating an orthogonal learning strategy within the CS in order to improve its exploitation ability, increase variation into the search, as well as reduce the number of control parameters, hence enhancing the performance of the original CS. The proposed MoCS is used to solve 23 benchmark functions. It is discovered that the MoCS outperformed the CS in terms of accuracy. Valian et al. [45] proposed modification of the original CS for improving convergence speed and accuracy. The authors used adaptive switching parameter p_a and αwhen generating new solutions. The performance of the MoCS is evaluated on the breast cancer and IRIS benchmark datasets. The MoCS outperform the CS as suggested by the experiments results.

Tuba et al. [48] used a sorting function during the computation of parameter αas a substitute for the permutation function in the CS, thus improving the performance of the original CS. The performances of the MoCS and CS are compared on 8 benchmark functions. The best results were found within the sphere function. The MoCS outperformed the results of the original CS. Saelim et al. [49] proposed crossover and random replacement for generating new solutions in place of Lévy flight and switching parameter p_a is proposed as being adaptive, unlike the original CS. The proposed MoCS is used to solve seven Traveling Sale Problems (TSP) and the results show that the MoCS improved the performance of the original CS.

Table 3
Continuous cuckoo search algorithm.

Topic	Reference
Golden ratio CS	[29,30]
Chaotic CS	[44,37,61,53,62,63]
Roulette wheel CS	[30]
Adaptive p_a and α CS	[45,46]
Sorting CS	[48]
Constructive NEH heuristic CS	[25]
Orthogonal CS	[43]
Crossover randomized CS	[49]
Parallelized CS	[31,64]
Annexing cooperating CS	[39]
Multi-objective CS	[65]
Gaussian distribution CS	[66]
Complex value CS	[67]
Quantum CS	[68,58]
Ranked CS	[69]
Single and Multi-objective CS	[70]
Simplex CS	[71]
Mutation CS	[72,40]
Communication and search	[73]
Inter-Species CS	[74]
Gradient CS	[51,75]
Sorting Multi-objective CS	[76]
Complex-vlued CS	[77]
GA operators CS	[78–80]
Self −adaptive CS	[50]
PSO CS	[81]
Frog leaping CS	[52]
Gamma CS	[82]
Chaotic and Elitism CS	[53]
Shuffled CS	[83]
Repeat cycle CS	[54,55]
Orthogonal CS	[84]
Adaptive CS	[32–36]
Opposite CS	[85,35]
Multi-objective CS	[86]
Disturbance CS	[87]
Sine wave CS	[88]
Adaptive p_a and βCS	[56]
Bi-CS	[89]
p_a CS	[51]
Gradient	[75]
Frog leaping	[83]
Auxiliary CS	[57]
Dynamic GA operators CS	[59]
Teaching − learning CS	[90]
Gaussian kernel CS	[47]
PSO based CS	[81]
Cauchy mutation CS	[91]
Hybrid Self-Adaptive CS	[60]
Adaptive αCS	[38]
Graph theory CS	[92]
Harmony Search CS	[61]

Fateen and Bonilla-Petriciolet [51] used gradient of the tangent plane distance function in their study as the source of new information in which the cuckoos are guided in searching of the global minimum. In the CS, both magnitude and the direction of the step are random, whereas, in the MoCS, the randomness of the magnitude of the step is reserved. Only one parameter p_a is required to be tuned. The MoCs is tested with 8 phase stability problems and it proved that the MoCS outperform the CS. Li and Yin [50] proposed MoCS with 2 DE new mutation rules based on the rand and best individuals to balance the exploitation and exploration. The 2 new mutation rules are balanced by linearly decreasing probability rule. The diversity of the population is improved by self-adaptive settings. The MoCS is applied to 16 benchmark functions and 2 chaotic systems for evaluating the performance of the MoCS. The MoCS is found to improve the effectiveness of the CS. Liu and Fu [52] the variety of initial populations of CS are upgraded by using the chaos theory. Inertia weight is embedded into the Lévy flights random search to enhance global search. The local search capability of the frog leaping algorithm is applied in MoCS to improve the local search, speed and precision. The MoCS performance is evaluated using test functions. The MoCS is superior to comparative algorithms in most of the test functions, as suggested by the experimental results.

Wang and Zhou [46] proposed MoCS with adaptive parameters instead of fixed as in the CS, to establish the parameters of chaos systems so as to improve accuracy and convergence speed. This is achieved by embedding adaptive parameters adjustment operation and the orthogonal design operation into the CS to create the MoCS. As such, p_a and α are adaptive in the MoCS. The MoCS efficiency is tested on the Lorenz system under both the noise-free and noise-corrupted conditions. The results indicated that the MoCS is better than the CS with the fixed values of p_a and α. Wang et al. [53] incorporated characteristics of chaos theory and elitism scheme into CS to advance the performance of the CS and preserve the best cuckoos, respectively. The α is replaced with 12 different chaotic maps. The MoCS efficiency is evaluated by utilizing 27 benchmark functions. It is found that the MoCS improve the solution quality of the CS and other metaheuristic algorithms. Zhang and Chen [32] proposed an adaptive α instead of the constant to improve population diversity and immune the algorithm from premature convergence. As a result, precision and convergence speed can be enhanced. The MoCS is modified by making the Lévy flights step length factor a global best reducing value with the growing generations to increase the speed of the convergence. The MoCS is applied to solve 5 benchmark functions. The MoCS improve the performance of the CS.

Ong [33] proposes adaptive α in the MoCS instead of the constant value of α in CS to speed up convergence to the global solutions. The MoCS is proposed on the assumption that the cuckoos lay eggs within the area that have a high chance of egg survival rate. As such, by adjusting the α adaptively, the cuckoos perform searches around the best solutions and explore the environment rigorously. The performance of the MoCS is evaluated with 5 benchmark functions. It is found that MoCS improve the convergence speed of the CS. Rangasamy and Manickam [34] proposed an adaptive MoCS to reduce the number of iterations required to reach an optimal solution, improve performance and provide the optimal solution suitable for the stability of multi-machine thermal power systems. The p_a and α in MoCS are made to be constrained instead of constant as in the CS. The MoCS is applied to analyze the stability of multi-machine thermal power systems. Comparative analysis proved the superiority of MoCS over CS. Guerrero et al. [56] put forward a dynamic adaptation MoCS to improve the performance of the CS. The p_a and β parameters of the MoCS are made to be dynamically adaptive using the Mamdani fuzzy system. The performance of the MoCS is found to be better than the CS after evaluation with 5 benchmark functions.

Abdollahi et al. [36] propose an MoCS by decreasing the egg laying rate of the CS. The CS egg laying rate is decrease from the maximum value of αto 0 for enhancing the accuracy of the local search. As such, the MoCS with large radius find solutions closer to the global optimum, whereas a smaller radius finds an exact solution to a problem. The performance of the MoCS is compared with that of CS on benchmark functions. The result indicated that the MoCS is more accurate with less number of benchmark functions evaluations than the CS. Cheung et al. [58] propose a MoCS with nonhomogeneous search strategies based on quantum mechanism to enhance CS search ability and deviate from premature convergence. Another control parameter, called learning regulator, is integrated onto the MoCS. Evaluation of the MoCS on 24 benchmark functions shows that the MoCS improve the CS performance and 10 other classical algorithms. Jia et al. [47] propose variable CS control parameters in view of the fact that the fixed P_a and α of Lévy flight makes it difficult for the CS to find an accurate opti-

mum. The modification of CS created MoCS. In MoCS, the p_a and α of Lévy flight are adjusted by Gaussian kernel and quaternion, which is utilized to represent individuals to be optimized in order to balance exploitation and exploration. The MoCS is evaluated using 5 benchmark functions. The results indicated that the performance of MoCS is better than PSO and CS. Naik and Panda [38] proposed an adaptive MoCS in which the α is made to be adaptive from the knowledge of its fitness function, as well as its current position in the search space. This is to improve the convergence speed of the MoCS, and enables it to be parameter free without the levy step. The performance of the MoCS is compared with that of CS on benchmark functions for evaluation purposes. The MoCS improve the performance of the CS. Subsequently, the MoCS is applied to face recognition analysis and it produced superior results. Boushaki et al. [57] proposed an MoCS by generating cuckoo function, an auxiliary local search after each new solution in order to improve the solution. The local search newly found better solution replaced the current solution. Performance of MoCS is evaluated on a document dataset. The results shows that the MoCS is better than the CS in solving clustering problems.

5.1.2. Hybridization in cuckoo search algorithm

This section discusses hybridization approaches in CS for solving optimization problems. Typically, the hybrid CS borrowed structures from the other nature-inspired metaheuristic algorithms, e.g., GA, or other problem solvers. Hybridization of the original CS eliminates the weakness of the CS and, thus, improves its performances, i.e., faster convergence to the optimal solution over a shorter period of time. The hybridization of two or more algorithms has proven to eliminate their limitations and capitalize on their strengths for obtaining a better solution [93]. Recently, Marichelvam et al. [25] incorporated constructive Nawaz-Enscore-Ham (NEH) heuristic within CS in order to improve its performance. The MoCS was applied for solving the scheduling problem of a furniture manufacturing company. The MoCS is found to perform better than the CS.

Layeb [68] integrated quantum computing operators within the CS to increase optimization capabilities. The quantum operators are integrated with CS as follows:

a *Quantum interference*: Amplifying the amplitude of the optimal result, whereas the bad solution is decreased.
b *Measurement*: Transforming the projection of the quantum vector into binary.
c *Mutation*: Allowing the exploration of new solutions. Hence, improved diversification is obtained of the search process. The MoCS outperformed the PSO with a penalty function and PSO with a repair function, when solving the 7 Knapsack problems.

Zheng and Zhou [66] modified the original CS by using Gaussian distribution for generating a new solution instead of the Lévy flight for improving the convergence rate. The MoCS was applied to solve 6 benchmark functions. The results indicated that the MoCS converged towards the optimal solution faster than the original CS. Zhou and Zheng [67] proposed the encoding of complex values in CS to expand information about the representation of a solution. The MoCS outperformed the results of the original CS in the optimization of 6 benchmark functions. Thus, the MoCS improved the precision of the results by a factor of more than 100–1,000 over the original CS. Subotic et al. [64] introduced parallelization in the original CS to reduce the execution time required to obtain the best solution. During parallelization of the MoCS, each execution of this algorithm was run as a separate thread in order to achieve the fastest execution. The performances of the developed MoCS were evaluated on 4 benchmark functions and the results show that the MoCS is more accurate and faster than the original CS.

Chang et al. [71] modified the CS by using the simplex method of Nelder and Mead algorithm to abandon and generate new nests in place of p_a to create an MoCS. Subsequently, the MoCS was applied to train fuzzy neural network. The performance of the MoCS trained fuzzy neural network was evaluated on nonlinear regression problems. The result shows that the MoCS is better than the original CS and BP in training the neural network. In Chiroma et al. [73], the CS communication capability of the cuckoo births was improved by the introduction of APSO to search for a better location that the best nest can share information with the cuckoo. The MoCS is used to train a neural network for the prediction of OPEC CO_2 emission from petroleum consumption. The MoCS was found to improve the accuracy and convergence speed of the original CS. Coelho et al. [72] used the mutation operator of the differential evolutionary algorithm in CS instead of p_a to overwhelm the problem of local exploration and premature convergence. The MoCS was applied to solve the chiller loading problem. The MoCS was able to improve the performance of the original CS, as suggested by the experimental results. Fateen and Bonilla-Petriciolet [75] incorporated the gradient of the unconstrained Gibbs free energy function to improve the balance between diversification and intensification of the CS. The MoCS was used in finding a global minimum of the Gibbs free energy in phase equilibrium calculations. The result shows that the MoCS improves the performance of the original CS.

Das et al. [74] proposed an inter species CS referred as the MoCS to improve the performance of the CS. The MoCS is a partition of the entire search space into regions by clustering the initial search space into clusters, and each of the clusters is treated as a habitat for particular species. The species of the CS are differentiated with different Lévy flight: Yang-Deb Lévy flight, Cauchy Lévy flight, Rayleigh Lévy flight and Gaussian walk Lévy flight. The variants of the MoCS share information among themselves to explore the search space effectively and reduce computation. The efficacy of the MoCS is tested using unimodal and multimodal benchmark functions. It is found that the MoCS success rate was the same with the original CS in all the benchmark functions. Kanagaraj et al. [78] modified the CS by hybridizing it with GA operators to form an MoCS for solving supplier evaluation combinatorial problems. The crossover is used for the egg reproduction of each cuckoo. The mutation is applied to mimic the color and pattern of the host births to avoid their eggs being recognized by the host bird. The surviving cuckoos were selected by the GA selection operator. In CS, one cuckoo lays one egg at a particular time based on Lévy flights, whereas in MoCS, each 2 parents of the cuckoos produce 2 cuckoo eggs. The MoCS was applied to solve the supplier evaluation problem. The results indicated that the MoCS is able to generate good quality solutions in most cases more than the original CS, and simulated annealing. Li and Yin [81] was inspired by PSO to modify CS. In the MoCS, the global best solution is kept at each level of the iteration to enhance diversity. The global best solutions are used to add information flow between the nests, which assist to increase global and local search. Thereby, they balance exploitation and exploration based on the random probability rule. The MoCS was applied to solve 30 benchmark functions. The result indicated that the MoCS is superior to the original CS and PSO. Roy et al. [82] proposed MoCS by replacing the Lévy flight with Gamma distribution to advance the performance of the original CS. As such, the new solutions in the MoCS are generated by the Gamma distribution instead of the Lévy flight. The MoCS was tested using the standard benchmark functions. The results indicated that the MoCS improved the performance of the original CS.

Lim et al. [79] were inspired by GA to modify CS, in which the GA crossover is used for chromosomal between cuckoos and mating occurs between the cuckoos. The eggs laid by the cuckoos in another host bird‘s nest compete for survival. Only the fittest eggs survive, while lower quality eggs are rejected. The cuckoo’s eggs

are mutated by the cuckoo to mimic the eggs of the host birds. The MoCS is used to solve the whole making sequence. The MoCS shows that the performance of the original CS is improved. Wang et al. [83] proposed an MoCS based on the concept of the frog leaping algorithm to improve local search. In the MoCS, the n is divided into different groups. Each of the groups is optimized by utilizing the CS. Subsequently, the individual in the separate groups are once more combined into n. The process is repeated until the number of required iterations have been reached. The MoCS is used to optimize the parameters of the neural network to build a model for predicting the concentration grade of the soft sensor. The result shows that the MoCS is better than the original CS in the optimization of neural network parameters. Xiang-Tao and Ming-Hao [84] integrated orthogonal learning into the CS to improve the search mechanism of the CS. Rand individual based on orthogonal was applied to balance both exploration and the exploitation. The MoCS effectiveness was evaluated through the Lorenz and the Chen chaotic systems. Experimental results indicated that the MoCS is better than the original CS. Ong et al. [35] incorporated two parent crossover into adaptive CS to allow information exchange between the optimal solutions. This can accelerate the convergence speed of the CS. The MoCS performance was evaluated using benchmark functions and the optimization of paper and pulp properties. The MoCS was found to speed up the convergence of the original CS to the optimal solution.

Wen et al. [88], **3**used sine-wave for generating new solutions instead of through Lévy flights, to ensure the solution feasibility and improve the search mechanism of the CS. The MoCS was tested using the IEEE 33-bus radial distribution system based on bi-objective and single objective optimization methods. The MoCS can attain high penetration of distributed wind generation with added profits. A summary list of the hybridized CSs is presented in Table 4. Jia and He [77] enhanced the diversity of nest information in CS based on variable in place of float encoding by using the real and imaginary parts of a complex number to create an MoCS. The MoCS performance was evaluated with 8 test functions. The result shows that the MoCs performed better than the CS in terms of solution quality and convergence speed. Wang et al. [54,55], **3**incorporated disturbance operation into the CS to create an MoCS to improve convergence velocity and accuracy. An MoCS is a repeat-cycle asymptotic self-learning and self-evolving disturbance. This is achieved by constructing a disturbance factor to make searching close to the bird nest's location more thorough. The effectiveness of the MoCS is evaluated through 6 benchmark functions and results show the superiority of MoCS over CS. Park et al. [85] used opposite based learning in CS to create an MoCS for improving its performance. In the MoCS, each of the nests has an opposite nest. The comparison of fitness is done by either one-to-one, that is comparing the original with the opposite nest, or by way of combining, that is the congregation of all the original and opposite nests and, subsequently, ranking them. Jumping rate is used to control the frequency of the opposite based learning so as to deviate from the possibility of being trapped in local minima. The MoCS is evaluated on 23 benchmark functions. Results show that the MoCS improved the performance of the original CS successfully.

An et al. [69] modified the CS to create one rank MoCS in order to improve optimal solution and convergence speed. In the MoCS, the new eggs generated via Lévy flights were partially replaced. The newly generated eggs are evaluated and ranked at once. The MoCS is used for solving a reactive power dispatch problem (IEEE 30-bus and IEEE 118- bus systems). The MoCS was found to be better than other metaheuristic algorithms. Zhao et al. [87] modified the Lévy flights due to its heavy-tail property to speed up convergence and deviate from convergence vibration during clustering when using CS. The MoCS searches for a new nest according to the following expression:

Table 4
Hybridization in the cuckoo search algorithm.

Topic	Reference
Crossover	[29,49]
Roulette wheel	[30]
Sorting function	[48]
Constructive NEH heuristic	[25]
Orthogonal learning strategy	[43]
Incentive mechanism	[31]
Annexing and cooperating operators	[39]
Gaussian distribution	[66]
Complex value	[67,77]
Mutation, quantum interference and measurement	[68]
Sigmoid function and probability	[94]
Greedy transformation algorithm	[95]
Arithmetic crossover	[70]
Simplex	[71]
Communication and Search	[73]
Mutation	[72,40]
Frog leaping	[96]
Genetic mutation	[97]
Mutation, crossover and selection	[78,79]
DE mutation	[50]
PSO	[81]
Frog leaping	[52]
Geometric operators	[98]
Frog leaping	[83]
Orthogonal	[84]
Two parent crossover	[35]
Sine-wave	[88]
Fuzzy	[56]
Gamma distribution	[82]
Disturbance	[54,55]
Rank	[85,69]
Disturbance	[87]
Gradient	[75]
Inter-Species	[74]
Chaotic and elitism	[53]
GA operators	[79,80]
Adaptive GA operators	[59]
Penalty function	[90]
Gauss map	[37]
Cauchy mutation	[91]
Random & stochastic	[60]
Graph theory	[92]
Nawaz-Enscore-Ham	[61,53,62,63]
Harmony Search	[61]
Orthogonal and simulated annealing	[61,53,62,63]

$$\mathbf{x}_i^{(t+1)}=\mathbf{x}_i^{(t)}+\alpha\oplus Levy(\beta)+w*rand*(\mathbf{x}_{best}-\mathbf{x}_i^{(t)}),\quad (i=1,\ 2,\ 3,\ \ldots,\ n)$$

Where $rand$ is the random number, w is the disturbance constant, $\mathbf{x}_{best}$ is the cluster center of the best cluster. The MoCS is applied for clustering of benchmark datasets. The experimental results show that the clustering convergence performance of MoCS is better than the CS.

Akumalla et al. [80] hybridized GA with CS to create MoCS to improve solution quality and convergence speed by extending search space. The MoCS update each chromosome/nest using the GA for generating an offspring through selection, crossover, and mutation operations. Evaluation of MoCS on the IEEE 30 bus power system network shows the optimum solution and run time of the MoCS is better than the CS. The MoCS is used to find the best location of FACTS devices in a multiple machine power system, thus, improving the voltage profile with minimum loss of power in the system. Daniel and Anitha [59] proposed an adaptive MoCS by modifying the CS to improve the image contrast of medical images. In the MoCS, the p_a is changed dynamically in each iteration, adaptive genetic algorithms operators, crossover and mutation are used for the replacing of the worst nest, unlike in the CS. The MoCS was applied for contrast improvement in medical images and its performance was compared with other algorithms in the literature. The

Absolute Mean Brightness Error (AMBE) of images for MoCS is better than other algorithms. However, the effectiveness of the MoCS was not compared with the original CS.

Huang et al. [90] proposes an MoCS that used penalty function to transform constrained optimization into an unconstraint. Subsequently, teaching–learning based optimization was applied to improve the local search capability of the CS. The MoCS is used for solving engineering optimization problems such as welded beam design, tension/compression string design and pressure vessel design. Results show that the performance of the MoCS is better than the results published in the literature. The possibility of CS simply to get stuck into local optimal solutions and cause low precision prompted Huang et al. [37] to propose an MoCS based on chaotic sequence. This is to prevent the MoCS from being stuck in the local optimum solution to enhance global convergence. In MoCS, the chaotic sequence (Gauss map) is applied to improve the initialized host nest location, change the α of Lévy flight and resetting of the host nest location. The performance of the MoCS is evaluated using benchmark functions. It is found that MoCS performance is better than that of CS, CS with Logistic map, CS with tent map, CS with Sinusoidal iterator and CS with Circle map. Maurya et al. [91] proposed MoCS in order to enhance images automatically. The MoCS algorithm generated a new nest using levy flight while p_a was abandoned. Subsequently, a new solution of the MoCS is created by Cauchy mutation. The MoCS is appled to enhance an image and its performance is compared with PSO, Histogram equalization and Linear contrast stretching. The MoCS enhanced the image more effectively than the conventional methods.

Mlakar and Fister [60] propose an MoCS called hybrid self-adaptive CS using the explicit control exploration search technique, self-adaptation of control parameters and reduction of population feature to improve the effectiveness and efficiency of the CS. In the MoCS, the search space is explored based on a random long distance search strategy to explore new solutions, while stochastic moderate distance and short distance search strategies exploit the current solution. The MoCS control the self-adaptive parameter settings. Reduction of population size is achieved by directing the search process towards a promising region. The effectiveness and efficiency of the MoCS is evaluated on CEC-2014 benchmark function suites. MoCS comparison with variants of CS, DE and original CS indicated that MoCS has potential for solving global optimization problems. Mohseni et al. [40] proposes MoCS by defining a flexible motion coefficient instead of using random or constant value. For n to converge into optimum point adaptively, mutation is applied to the global best before each immigration process. The mutation is aimed at creating diversity and increasing the MoCS convergence rate. The MoCS is applied on benchmark functions before solving feature selection and spacecraft attitude control problems for the purpose of evaluating the effectiveness of the MoCS. The MoCS was found to be better than the CS, PSO, harmony search and DE.

Nguyen et al. [92] propose an adaptive MoCS to reduce power loss and improve voltage stability. The infeasible individuals in the CS violating radial constrain are reduced in MoCS by using the graph theory. The graph theory is applied to determine the search space responsible for reducing infeasible network configurations in a distribution network. The MoCS is evaluated on three different distribution networks at 7 different scenarios. The result of the experiment indicated that the MoCS is able to obtain optimal location and size of the distribution generation. Wang et al. [61] hybridized CS with harmony search to create MoCS so as to improve the CS search ability. To enhance convergence speed, the pitch adjustment operation of the harmony search with similar function to mutation operator is applied to complement the CS update. Elitism is added into the MoCS to retain the best solutions within the cuckoo population. The evaluation of the MoCS on benchmark functions shows that the MoCS is better than the CS and other meta-heuristic algorithms in most cases. Wang et al. [53,61–63] propose MoCS to establish optimum parameters of chaotic systems. Orthogonal design and simulated annealing operation are added into CS to enhance the CS exploitation capability. The MoCS is applied to establish the parameters of Lorenz and Chen chaotic systems under different (noiseless and noise) conditions. The experimental result indicated that the MoCS can estimate parameters of chaotic systems with higher accuracy, reliability and energy efficiency than the CS, PSO and GA.

5.1.3. Multi-objectives cuckoo search algorithm

The single objective CS has been extended to multi-objective by researchers in order to solve multi-objective problems. In a single objective CS, only one optimal solution can be found in the search space, except in the case of multiple and equally optimal points. Single objective CS cannot be applied to solve multiple objective problems. Therefore, researchers attempted to modify the original CS to multi-objective CS. In multi-objective problems, the algorithm is required to deal with a Pareto front. In 2013, Yang and Deb [65] extended the single-objective CS to a multi-objective CS capable of solving multi-objective problems. The following modifications are performed to form the multi-objective CS: (1) Unlike the original CS, each cuckoo lays k eggs at a time and subsequently dumps them into a randomly chosen nest. The k-th egg corresponds to the solution of the k-th objective. (2) Each of the nests will be abandoned with switching parameter p_a. Then, new nests with k eggs will be built based on the similarities/differences of the eggs. Diversity can be generated by some random mixing. The MoCS is evaluated on multi-objective functions, as well as solving structural engineering design problems. The result indicated performance improvement of the MoCS. Balasubbreddy et al. [70] proposed MoCS by introducing arithmetic crossover in CS instead of the Lévy flight that control the balancing of exploration and exploitation to decrease the diversity of the problem, thereby obtaining the best final solution in less number of iterations. The proposed MoCS is capable of solving both single and multi-objective problems. The MoCS is tested on the power system, single and multi-objective functions. The results indicated that the MoCS performs better than the CS in minimizing the cost of fuel and convergence speed.

He et al. [76] modified multi-objective CS by embedding archiving and non-dominated sorting to advance its performance. In the MoCS, all the non-dominated and dominance solutions are sorted out from the generated solutions. The sorting is continuous until all solutions are sorted into layers differently. The MoCS is evaluated by using 7 benchmark functions. The MoCS performance is compared to a non-dominated sorting GA. It is found that the MoCS is better than the non-dominated sorting GA. Zeltni and Meshoul [86] translate the basic concept of updating solutions in CS in the context of multi-objective using leader selection strategies to create a Pareto based multi-objective CS. The authors used archive to manage non-dominated solutions and leaders, such as random selection, the sigma method, crowding distance method, a hybrid selection strategy, and MaxiMin methods are used to select leaders for updating solutions in MoCS. This differs from the CS, in which the best solutions are used to update the current generation for the computation of a new generation. Five benchmark functions with two objectives related to numerical optimization are used to evaluate the effectiveness of the MoCS. Experimental evidence shows that the MoCS have the capability of solving multi-objective problems and none of the leader selection strategies are the best in terms of convergence and diversity.

Zhou et al. [133] proposes a multi-objective BCS with local search to detect community. The multi-objective BCS redefined the nest location updating strategy and abandon operator of the CS discrete form. The optimum initial population are generated by the local search strategy and a clone operator. The negative

Table 5
Multi-objective in the cuckoo search algorithm.

Topic	References
Multi-objective CS	[65]
Archiving and non-dominated multi-objective CS	[76]
Leader selection multi-objective CS	[86]
Crossover multi-objective CS	[70]
Local search and clone operator CS	[99]

Table 6
Binary cuckoo search algorithm.

Topic	Reference
BCS	[94]
Application of BCS in power distribution	[100]
Cuckoos' matters Randomized BCS	[41]
Discrete quaternary BCS	[95]
Non- Lévy flight BCS	[42]
Inter-species BCS	[101]
Shuffled frog leaping BCS	[96]
Mutation and confidence interval BCS	[97]
Variable neighborhood descent BCS	[102]
Theory of distance BCS	[103]
Geometric BCS	[98]
	[54,55]
Permutation BCS	[104]
Bound BCS	[105]
Bi-CS	[89]
Interchange and cyclical shift BCS	[106]
Smallest position values BCS	[61,53,62,63]
Multi-objective BCS	[99]

ratio association and ratio cut are the two objective functions that were minimized. The experiment was conducted on synthetic and real-world networks. It was found from the experimental result that the multi-objective BCS can detect high quality community structure without prior information and had better performance than multi-objective BCS without local search, multi-objective BCS without clone operator, among others. The summary of the list of multi-objective CS are presented in Table 5.

5.2. Binary cuckoo search algorithm

Gherboudj et al. [94] proposed the binary form of the CS for dealing with binary optimization problems instead of the continuous one. In the original CS, the elements of a solution are represented as real numbers, whereas in the BCS, each solution is composed as a sequence of bits. The Sigmoid function and probability model are used in the CS to obtain the binary solution. The BCS is used for solving the Knapsack problem. The results show that the BCS outperformed the quantum inspired CS, binary PSO (BPSO) and harmony search algorithms. Further statistical tests indicated that the performance of the BCS is significant. The BCSs observed in our study are summarized in Table 6.

The improvement of the BCS proposed by Ouaarab et al. [41] considered a population of cuckoo nests as the first level of controlling intensification and diversification because this population consists of n solutions, i.e., nest positions. The n can be quantified as a second level of control that can be reconstructed by adding a new category of cuckoos' matters therefore impoving the effectiveness of the CS search. As a result, the search was strengthened intensively around the best solutions, while randomization was used properly to explore new areas using Lévy flights' randomization. These modifications amended the BCS to the MoBCS. Hence, improving the speed and minimizing the probability of falling into local optima is expected. Simulation comparative analysis of the results indicated that the MoBCS performed better than the BCS in both accuracy and convergence speed. The MoBCS is run for 30 trials across 14 TSPs.

Zhou et al. [95] modified the BCS (discrete quaternary version of the original CS) by adding a greedy transformation algorithm within BCS to obtain the MoBCS. The MoBCS was used to solve the seven nodes graph coloring problem. The results indicated that the MoBCS performed better than the PSO and the modified PSO in terms of speed and accuracy. However, the results of MoBCS are not compared with the performance of the original BCS as performed by Ouaarab et al. [41]. Jati et al. [42] proposed a non-Lévy flight BCS by not using Lévy flight in the updating scheme to improve the exploitation of the BCS. The performance of the MoBCS is evaluated on 7 TSP problems. The MoBCS is found to perform better than the memetic algorithm, but MoBCS could be trapped in local minima for some problems.This study did not compare the results of MoBCS to those of the BCS. The BCS suitability for solving binary problems motivated Rodrigues et al. [100] to use BCS for the detection of theft in power distribution. The obtained results show that the BCS outperforms the binary gravitational search algorithm, whereas the binary bat algorithm, binary firefly algorithm and BPSO had similar accuracies as the BCS. Conversely, the BCS was faster than all the other compared algorithms in tests.

Hanoun [102] hybridized CS with variable neighborhood descent based on the insert and interchange operators o as to advance performance and speeding up convergence rate. The Lévy flight is replaced by the discrete neighbor sequence operator based on the insert and interchange methods to create the BCS. The BCS is applied to solve NP-hard single objective and multi-objective job scheduling problems. The results indicated that the BCS outperforms state-of-the-art algorithms. Feng et al. [96] used hybrid encoding, and each of the cuckoos is represented by two tuples to perform a search in the solution space. Also, Sigmoid function is utilized in the CS to convert a real-coded vector to a binary-coded vector to create BCS. The BCS was hybridized with an improved shuffled frog leaping to solved 0–1 knapsack problems. The result shows the solutions quality of the BCS is better than the CS, GA and DE in a strongly correlated knapsack problem. Feng et al. [97] modified the CS by introducing confidence interval and genetic mutation to improve its performance. The confidence interval is used for updating new positions of cuckoos to allow the population moves towards the global best solution. Moreover, genetic mutation is applied to speed up convergence time and prevent the MoCS from being trapped into local optimum. The concept of Feng et al. [96] is utilized to transform the MoCS into a BCS. The BCS effectiveness is evaluated through the 0–1 knapsack problems. The results indicated that the BCS outperforms the CS.

Dasgupta and Das [101] present a binary version of inter species CS to minimize makespan and mean flow time in hybrid flow-shop scheduling and permutation flow-shop sequencing problems. In this study, different species of BCS with different Lévy flight, such as Yang-Deb Lévy flight, Cauchy Lévy flight, Rayleigh Lévy flight and Gaussian walk Lévy flight, are used as different species of the BCS. The variants of the BCS share information among themselves to explore the search space effectively and reduce computation. The BCS is used to solve flow-shop scheduling problems. Results indicated that the proposed BCS outperformed several comparison algorithms, including Discrete Firefly algorithm (DFA), parallel GA, simulated annealing, PSO, Ant colony optimization (ABC), NEH and TABU. Mahmoudi and Lotf [103] convert CS to BCS by redefining addition, subtraction and multiplication of the CS based on the theory of distance on discrete space to create BCS for solving a graph coloring problem. The BCS is applied to solve graph coloring. The result indicated that the BCS performs better than the original and the modified version of simulated annealing.

Soto et al. [98] embedded partially matched crossover, multi-parental sorting crossover and feasible geometric mutation into CS to improve its performance and explore discrete space for solving sudoku problems. This forms a BCS capable of exploring discrete

search space. The BCS is applied to solve the sudoko problems with different complexity. The result shows that the BCS is better than GA and hybrid tabu search algorithms. Wang et al. [54,55], **3** proposed a BCS that utilizes swap and insert operators to enhance local search, which help in solving flow-shop problems with blocking. The smallest position value rule is used for the conversion of continuous solutions into discrete. Hence, the BCS is able to produce a binary solution. The n is generated using the NEH heuristic method. Experiments are conducted on the Taillard's benchmark to evaluate the efficiency of the BCS. Results indicated that the BCS performs better than the CS and other metaheuristic algorithms. Laha and Behera [104] converted the continuous position of the CS into discrete permutation scheduling for obtaining a cuckoo by Lévy flight to the proposed BCS. The BCS is applied to minimize makespan for solving the identical parallel machine scheduling problems. The results show that the BCS converges faster than the state-of-the- art algorithms. Soto et al. [105] proposed a BCS for solving a non-uni-cost set covering problem that belongs to the Karp's 21 NP-complete problems. The generated real numbers are converted to binary according to the following expression:

$$x_j = \begin{cases} 1 & \text{if } r < x'_j \text{ or } x'j > U_b \\ 0 & \text{if } r \geq x'_j \text{ or } x'j < L_b \end{cases}$$

where x'_j holds the value to be discretized for variable x_j of the set covering problem solution, and r is a normal distributed random value, U_b and L_b are the upper and lower bound respectively. This makes it possible to solve the set covering problems. Experimental results show that the BCSconverges rapidly to the global optimum and it is robust. The BCS is not compared with any algorithm. Huihua et al. [89] proposed an MoCS that deals with both binary and the continuous variables to solve a crude oil selection and blending optimization problem. The continuous variables are updated by using the Lévy flights, whereas updating the binary variables are done by abandoning the nest with a probability. The MoCS was applied to crude oil selection and blending optimization. The experimental results indicated that the MoCS is better than the GA and PSO.

Majumder and Laha [106] propose a BCS in which the fractional scaling factor is used to determine α of the Lévy flight in the BCS. Current solutions in the BCS are improved by the application of interchange and cyclical shift neighborhood search methods. The convergence speed of the BCS is improved by the response surface method based on desirability function. The BCS is applied to solve the problem of two machine robotic cell scheduling. It was found to solve the problem better than GA and simulated annealing. Wang et al. [61,53,62,63] proposes BCS by hybridizing CS based on 4 strategies to convert the CS to a BCS, generate a quality solution, improve convergence speed and enhance exploitation. The smallest position values are used to convert continuous solution into discrete. The Nawaz-Enscore-Ham is applied to the initialized population in order to generate a quality solution. The modified generalized opposition-based learning is used to enhance convergence speed. The exploitation is enhanced by local search strategy. To evaluate the effectiveness of the BCS, its performance is evaluated on Taillard's benchmark. The experimental results indicated that the CS performance is improved. Note that this section was not divided into parameter control and hybridization because these papers are already addressed in Tables 2 and 4.

5.3. The most frequently used modified cuckoo search algorithm

A modified CS, as proposed by Walton et al. [29] (also MCS), was found to achieve the greatest benefits among the other modifications proposed by researchers in the literature. The MCS is the most frequently used modified CS among researchers to solve problems in different domains of applications. For example, Piechocki et al. [107] used the MCS due to its superior performance over the original CS for optimizing a selection process within an energy-efficient system. The MCS was also applied for minimizing energy demands, total costs of building the system and carbon dioxide emissions. This study found that the global approaches for the designing of energy-efficient, cost effective and environmentally-friendly farm power systems are more effective. Finally, the MCS is used by Salimi et al. [108] for the training of ANN for eliminating the limitations of the backpropagation algorithm in order to improve convergence speed and accuracy. The effectiveness of the proposed algorithm is evaluated on 7 UCI benchmark datasets, but the best results are found within the IRIS dataset. However, the performance of this algorithm is not compared directly with the performance of another CS trained ANN model.

Table 7
Applications of MCS by [29].

Application	Reference
Energy efficiency	[107]
UCI benchmark dataset	[108]
Diabetes	[109]
Temperature and relative humidity	[110]
Mesh optimization	[114]
Uni-modal and multi-modal optimization	[111]
Design of power transmission lines	[112]
Short term cascaded hydrothermal scheduling	[113]

Similarly, in order to improve accuracy and convergence of speed, Giveki et al. [109] applied the MCS for optimizing the parameters of SVM for building a robust diabetes diagnostic model. The results obtained by this algorithm outperformed the results of the other comparison algorithms in both accuracy and convergence speed. Unfortunately, this study did not compare the obtained results with the results of the optimized CS-SVM. Abubakar et al. [110] used the MCS to train a Functional Link NN (FLNN) in order to eliminate the limitations of the backpropagation algorithm for improving the convergence of speed and accuracy. The trained FLNN model was applied for prediction of temperature and relative humidity prediction in Malaysia. The study concluded that the performance of the model was better than the backpropagation algorithm, but is not compared further with the performance of the CS trained FLNN.

The MCS was also implemented for solving a mesh optimization problem, where this algorithm outperformed the Laplacian smoothing and Lloyd's algorithms in the generation of mesh. Nasa-ngium et al. [111] modified the MCS further by replacing the inverse golden ratio with a chaotic random number generator in order to enhance its performance. Results from simulation analysis on unimodal and multi-modal functions indicated that the MoCS proposed by Nasa-ngium et al. [111] improved the performance of the MCS.

Güneş et al. [112] used MCS to design transmission lines with minimum noise and maximum power delivery at each operation frequency. It was found that the composite right/left-handed transmission lines allow a designer to design a high performance miniaturized amplifier with low noise. The MCS shows performance improvement over CS, PSO and Differential Evolution. Nguyen and Vo [113] applied MCS for solving short-term cascaded hydrothermal scheduling. This is to determine the optimal operations of thermal plants, as well as cascaded reservoir. The result indicated that the MCS is superior to other methods already discussed in the literature in terms of efficiency. A summary and applications of the MCS are presented in Table 7.

5.4. Applications of the cuckoo search

The CS has been used for optimization in many areas, including computational intelligence. The applications of the CS have

proven to be robust and promising over other algorithms [6]. For example, the CS has been applied in the training of ANN, robots, wireless sensor networks, classification, clustering, design optimization, images, algorithms' analysis, control, etc.

From the ANN training point of view, metaheuristic algorithms were applied to train an ANN in order to deviate from the limitations of the ANN and improve its effectiveness [115]. The CS was found to train spiking ANN better than DE [116], CS trained Elman ANN better than ABC [8,9], CS trained Recurrent ANN better than ABC and PSO [8,9].

Mohanty and Parhi [134] proposed CS for the path planning problem of an autonomous mobile robot in an environment that is unknown or partially known to the mobile robot. The environment was populated with varieties of stationary objects. The CS based mobile robot was able to avoid the obstacles by guiding the movement of the mobile robot to the target point with the shortest path length more than the robots based on PSO and GA.

Manikandan and Selvarajan [135] used CS to find the optimal clustering object into clusters using multiple datasets. The CS performance on the data clustering was compared with that of K–means, PSO, genetic K–means, fuzzy C means and Fuzzy PSO. The results indicated that the CS outperformed the comparison algorithms. To minimize active power loss and improve voltage in the distribution of power in a power distribution system, CS was proposed to solve the power problem. The CS was able to solve the distribution of the network reconfiguration. Comparative analysis of the CS results with that of PSO and GA signified that the CS is superior [117]. The CS was applied for the selection of a cluster head in a wireless sensor network to reduce energy consumption. The performance of the CS was evaluated and it shows that the CS has the capability of selecting the optimal clusters that resulted to prolong the lifetime of the network more than the traditional methods [118]. In addition, Dhivya and Sundarambal [119] used CS to improve the performance of a wireless sensor network by incorporating balanced energy dissipation, which results in the formation of the optimal number of clusters. As such, the wireless sensor network was more able to consume minimum energy than the traditional cluster-based routing method.

The study conducted by Bhandari et al. [136], applied CS to the examination of a standard set of satellite images based on several numbers of thresholds. Experimentally simulated results based on Kapur's entropy show that the CS has the potential to be used for accurate and efficient utilization in multilevel threshold problems. Further study by Brajevic and Tuba [137] applied CS to multilevel image thresholding using two different objective functions, namely, Kapur's maximum entropy thresholding function and multi Otsu between-class variance. An experiment was conducted on standard benchmark images with known optima from an exhaustive search. The simulation results suggested that the CS is more robust and efficient than the PSO and DE. Ismail et al. [120] applied CS for fractal image compression based on ordered vector of range blocks. The CS is found to be scalable and robust in image compression compared to PSO and GA.

Araghi et al. [121] applied CS to tune the parameters of intelligent controllers, namely, ANN and adaptive neuro-fuzzy inference systems for traffic signal control in a multi-intersection network. The study has shown that the CS intelligent controller is more robust and effective than the Q-learning controllers in responding to traffic demands. In addition, Mehta et al. [122] used CS for the optimization of controllers at the terminal of a synchronous generator.

Civicioglu and Besdok [14] analyzed CS, PSO, Differential Evolution (DE) and Artificial Bee Colony (ABC) algorithms for comparison purposes. The CS and DE have proven to have better performances than the ABC and PSO. An extensive competition between CS and established *meta*-heuristic algorithms such as PSO and GA was carried out by Gandomi et al. [123] for validation against structural engineering optimization problems which was applied to 13 design problems. The comparative simulation results proved the robustness and efficiency of the CS over PSO and GA.

6. Performance analysis of CS

This section presents the performance analysis of CS regarding the results found in the papers. The section consists of the following major components: The benchmark function suites, the most widely used parameter control, the widely used performance measures, computation of the percentage improvement in the CS convergence speed and solution quality. Lastly, the trend of the modifications made to the CS benchmark suites are depicted.

6.1. The benchmark function suites

The modification of the CS is evaluated typically on benchmark datasets or functions. Table 8 shows that optimization is the main task used by most researchers to evaluate the performance of MoCS. None of the researchers had evaluated the MoCS, MoBCS and BCS across multiple tasks (e.g, classification, optimization, and numerical prediction). It could have been interesting to observe the behavior of the proposed modification on multiple tasks.

Let us notice that the selection of the benchmark function suite has a great influence on the results for the MoCS. Typically, benchmark suites consist of functions that are relatively easy to be solved because these have the global optima located close to the center of the search space [124]. On the other hand, benchmark function suites, like CEC'14 [125] and COCO'14 [126], consist typically of the following function types: Unimodal, multi-modal, hybrid and composite.

Unimodal functions have a single global optimum and no local optimums while, normally, functions in these suites are non-separable and rotated. The separability of a function determines how difficult the function is to solve. This means, the function $f(\mathrm{x})$ is separable if its parameters $\{x_i\}$ are independent. In general, the separable functions are considered to be the easiest. In contrast, the fully-nonseparable functions are usually the more difficult to solve. The degree of separability could be controlled by dividing the object variables into several groups randomly, each of which contains a particular number of variables. Although some of the used functions are separable in their original forms, applying techniques such as Salomon's random coordinate rotation makes them non-separable.

Multi-modal functions are either separable or non-separable, and they are also rotated or shifted. To develop the hybrid functions, the variables are divided randomly into some subcomponents and then different basic functions are used for different subcomponents [125]. Composition functions consist of a sum of two or more basic functions. In the CEC'14 suite, for instance, hybrid functions are used as the basic functions for constructing composition functions. The characteristics of these hybrid and composition functions depend on the characteristics of the basic functions.

6.2. The most widely used parameter control

The control of exploration and exploitation are probably those challenges facing metaheuristic algorithms which still remain an open question. Parameters are controlled during the execution of algorithms in order to yield the optimal solution within a short CPU time. However, a systematic way of varying the parameters is an unresolved problem [3]. Like other metaheuristic algorithms, the CS is also sensitive to parameter settings [18]. The performances of the MoCS and CS are sensitive to the population size n, and problem

Table 8
The benchmark dataset/functions used for evaluation of the modified CS.

Benchmarks	Task	Reference
Breast cancer and IRIS	Classification	[45]
Array geometry synthesis	Optimization	[30]
Sonar, Breast, Pima, Glass, Vehicle, Iris, and Pen Digits	Classification	[108]
Unconstrained standard benchmark	Optimization	[29,31,39,43,48,66,67,64,81]
UCI Diabetes diseases	Classification	[109]
The scheduling problem of a furniture manufacturing company	Optimization	[25]
Temperature and relative humidity	Prediction	[110]
Knapsack	Optimization	[68,94]
TSP	Optimization	[49,41,42]
Energy efficiency	Optimization	[107]
Brazilian electrical power company dataset	Pattern recognition	[100]
Graph coloring problem	Optimization	[95,103]
Mesh generation	Optimization	[114]
Schaffer's Min–Min, ZDT1 function with a convex front, ZDT2 function with a non-convex front, ZDT3 function with a discontinuous front, LZ function. Design of a welded beam, Design of a disc brake.	Optimization	[65]
Lorenz	Optimization	[44]
IEEE 30-bus and IEEE 118- bus systems	Optimization	[69]
Power system (generation fuel cost, emission, and total power loss), Schaffer function, Booths function, Matyas functions,	Optimization	[70]
Nonlinear regression problem	Optimization	[71]
OPEC CO_2 emission	Prediction	[73]
Chiller loading problem	Optimization	[72]
Flow-shop scheduling problem	Optimization	[101,61,53,62,63]
De Jong's, Easom's, Shubert's bivariate, Griewangk's, Ackley's, Schwefel's, Michalewicz's, Multiple Peaks, Michalewicz's, Rosenbrock's (unimodal and multi modal)	Optimization	[74]
Phase stability problems	Optimization	[51]
Phase equilibrium calculations (multimodal)	Optimization	[75]
0-1 knapsack problems	optimization	[96]
0-1 knapsack problems	Optimization	[97]
Job scheduling problems	Optimization	[102]
Schaffer's, ZDT1 function with a convex front, ZDT2 function with a non-convex front, ZDT3 function with a discontinuous front, ZDT4 function with a multimodal front, ZDT6 function has some non-uniformity in its search Space,	Optimization	[76]
Benchmark function (names not given)	Optimization	[77]
Supplier evaluation combinatorial problem	Optimization	[78]
Shifted sphere function, Shifted Schwefel's, Rosebrock's function, Shifted Schwefel's, Shifted Ackley's, Shifted rotated Ackley's, Shifted Griewank's, Shifted rotated Griewank's, Shifted Rastrigin's, Shifted rotated Rastrigin's, Shifted rotated Rastrigin's, Schwefel's function, Penalty 1's, Penalty 2's, Composition function 1, Composition function 6, Lorenz system., Chen system	Optimization	[50]
Hole making sequence	Optimization	[79]
Sphere, Rastrigin, Rosenbrock, Griewank, Ackley, Camel, Schwefrl's, Quadric, Easom, Shaffer, Schaffer, Schwefrl's, ohachevsky1, Bohachevsky2	Optimization	[52]
Ackley, Griewank, Rosebrock's, Rastrign, Sphere, Zakharov	Optimization	[82]
Sudoku problems	Optimization	[98]
Ackley, Pathological function, Alpine, Penalty, Brown, Csendes, Perm, Dixon and Price, Fletcher-Powell, Powel, Griewank, Quartic with noise, Holzman 2 function, Rastrigin, Levy, Rosenbrock etc and 12 chaotic maps	Optimization	[53]
Soft sensor	Prediction	[83]
Chaos systems (noise free and noise corrupted conditions)	Optimization	[46,61,53,62,63]
Taillard's benchmark	Optimization	[54,55]
Sphere, Rosenbrock, Griewank, Rastrigrin, Sumsquares and Michalewicz	Optimization	[54,55]
Lorenz and Chen chaotic systems	Optimization	[84]
Rosenbrock, DeJong, Schaffer, Rastrigin, Ackley,	Optimization	[32]
Ackley, De Jong, Griewank, Rastrigin, Rosenbrock	Optimization	[33]
Benchmark functions 23 (names not mentioned)	Optimization	[85]
Stability analysis of multi-machine thermal power systems	Optimization	[34]
ZDT1- 3 from the ZDT series proposed by Ziztler et al., Fonseca and Flemings study, Schaffers,	Optimization	[86]
Parallel machine scheduling problems	Optimization	[104]
De Jong, Rosenbrock, pulp and paper properties	Optimization	[35]
65 Set covering problems non-unicost instances from OR-Library	Optimization	[105]
Iris, Glass, Wine and Sonar	Clustering	[87]
IEEE 33-bus radial distribution system	Optimization	[88]
Sphere, Rosenbrock, Rastrigin, Ackley and Griewank	Optimization	[56]
Crude oil selection and blending	Optimization	[89]
Benchmark functions	Optimization	[36]
IEEE 30 bus power system network	Optimization	[80]
Classic300, classic400, IRIS	Clustering	[57]
24 uni-modal and multi-modal functions	Optimization	[58]
Medical images	Optimization	[59]
Welded beam design, tension/compression string design and pressure vessel design	Optimization	[90]
Weierstrass, Sphere, Ackley, Axis parallel hyper-ellipsoid, high conditioned elliptic, Schwefel P2.22, Griewangk, Zakharov and Rosenbrock	Optimization	[37]
Rosenbrock, Ackley, Schwefel, Easom and Xin-She Yang function.	Optimization	[47]
Power transmission lines	Optimization	[112]
Two machine robotic cell sequence	Optimization	[106]

Table 8 (*Continued*)

Benchmarks	Task	Reference
Lathe tool images (cameraman, house and Mandrill)	Optimization	[91]
CEC-2014 custom benchmark test suite (unimodal, multi-modal, hybrid and composite functions), pressure vessel design and speed reducer	Optimization	[60]
Bukin N.6, Greiwank, Schwefel, Styblinski, f1 (CEC13), f5 (CEC13), f11 (CEC05), f15 (CEC05), f18 (CEC05), f22 (CEC05), feature selection and spacecraft attitude control.	Optimization	[40]
Unimodal, multimodal and face recognition analysis	Optimization	[38]
Network topology and distribution network	Optimization	[92]
Short-term cascaded hydrothermal scheduling	Optimization	[113]
Ackley, Fletcher–Powell, Griewank, Penalty #1, Penalty #2, Quartic with noise, Rastrigin, Rosenbrock, Schwefel 2.26, Schwefel 1.2, Schwefel 2.22, Schwefel 2.21, Sphere and Step.	Optimization	[61]
Synthetic and real-world network	Detection	[99]

dimensionality [25]. Large n does not guarantee that the CS can produce the optimal solution [48].

In the following Tables 9–14 the phrase "Not Applicable" implies that there was no implementation. "Not reported" signifies that the researchers have not reported the values used during the implementation of the algorithm, or the values of the performance criteria were not reported. "Graphical presentation" indicates that the solution quality or convergence speed is presented graphically, but not as equally as numerical values. In the case of phrases "Not Applicable", "Graphical representation", and "Not Reported" the percentage improvement could not be computed due to lack of numerical values.

Metaheuristic algorithms, including CS, require the starting n cuckoo nests to be initialized, which shows bias towards the population size n [114]. It would be better for an algorithm to generate optimal parameter values automatically and operate without requiring significant input from the users [114]. The values of the CS parameters can be selected based on certain preliminary trials (trial-and-error) [41,114] or adopting their parameter values from the literature [31]. The optimal parameter values used by the studies reviewed in this paper (see Section 5) are presented in Tables 9 and 10 respectively.

The parameters p_a and α are critical for the performance of the CS. Tables 9 and 10 show that most researchers adopted the switching parameter $p_a = 0.25$ and scale factor $\alpha = 1$ as suggested by Yang and Deb [4], whereas others used different values that are realized through experimental trials. Researchers can experiment with the standard p_a and α values, as well as some other trial-and-error values for comparison before adopting the most effective results, because there is no guarantee that the standard p_a and α can always produce the best results. There are inconsistencies in the values of the parameters, as is shown clearly in Tables 9–10. Probably the parameter settings depended on the problem domain, which could have been responsible for the inconsistencies in the optimal values of the parameters. However, Walton et al. [114] suggested that it is better to use a parameter that works on a variety of problems. The $p_a = 0.25$ and $\alpha = 1$ works well for a variety of problems as revealed in this review (see Tables 8–10). Therefore, this combination of the optimal parameter setting could be considered as standard.

6.3. The most widely used performance measures

As stated by Yang [3], the best algorithm converges to the best solution within a short period of time. However, the qualities of stochastic algorithms cannot be judged according to a single run. In order to make comparisons between more stochastic algorithms as fairly as possible, more independent runs need to be conducted. Additionally, each of the algorithms in the study must have the same amount of resources and use the same performance measures. Unfortunately, the results of experiments in the analyzed papers were measured in many different ways, e.g., the average results over a given time, average time for giving results, best results over some runs, etc. That is, some studies only considered convergence speed but ignored the quality and vice versa, while some other studies considered both the convergence speed as well as the solution quality. The performance criteria for measuring depends on the objective of the study. As a result, such measures are hard (or even impossible) to compare between each other.

In general, these measures can be divided into two groups, i.e., efficiency and effectiveness measures. The former is dedicated to measuring the algorithm's speed, while the latter for the algorithm's quality. Typically, measures like CPU time, and the average evaluations for solutions (EAS), are devoted to measuring the algorithm's speed, while measures like the Success Rate (SR), Mean Best Fitness (MBF), Best Fitness (BF), Average Best Fitness (ABF), Mean Square Error (MSE), makespan, Average Percentage of Classification Accuracy (APCA), Percentage Deviation of the best solution (PDbest), Percentage of Classification Accuracy (PCA), Power Loss (PL), Best Classification Accuracy (BCA), Best Accuracy (BA), makespan, etc. for algorithms' qualities at termination. Metaheuristic algorithms are run more than once to compute the mean, best and worst results as metaheuristic algorithms are not deterministic.

The EAS is the number of iterations required for an algorithm to converge towards the optimal solution. CPU time refers to the time taken for the algorithm to converge to the optimum solution measured in seconds. BCA is defined as the optimal accuracy found among several runs. BA is the accuracy achieved by the algorithm in finding the shortest possible path. Makespan is used in job scheduling problems for measuring successful production, and is defined as the time taken for the last job to go out of the production system. PDbest is the percentage deviation of the best solution length found over the best solution length of several runs. MBF is the mean value of the best fitness value computed over several runs. SR can be defined as the ratio of the success to maximum runs multiplied by 100. ABF is the average value of the best fitness value computed over several runs. APCA is the average percentage rate of correct classification computed over several runs, whereas PCA is not an average of the accuracy and is simply the best classification accuracy achieved. BF is the best fitness value found over several runs of the algorithm, and the MCA is the mean classification accuracy computed over several runs. MSE refers to the estimator that measures the average of the square error between the actual and estimated solution found by the algorithm.

Power Loss (PL) (MW) is the total loss of power in electrical systems which is required to be minimized to avoid wastage of power lost. Uniformity Index (UI) can be defined as the variance of the minimum normalized Euclidean distance [76]. The Percentage Relative Difference (PRD), formulated based on makespan, can measure the performance of an algorithm. The larger the PRD, the better the algorithm is [54,55]. The damping ratios are computed for the weakly damped oscillatory mode eigenvalues. The Damping Ration (DR) is better in terms of providing more positive damping to the oscillations [34]. Diversity metric shows the uniformity of

Table 9
The optimal parameter settings of MoCS and CS used to converge towards the optimal solution.

MoCS	CS	Reference
p_a (min) = 0.05, p_a (max) = 0.5, α (min) = 0.01,α (max) = 0.5, λ = 1.5 .	n = 10, p_a= 0.1 and α =1.5 .	[45]
p_a = 0.25, *nest* = 30, α = 2, roulette wheel inertia weight [0.9 1.0]	p_a = 0.25, *nests* = 30, α = 2, roulette wheel inertia weight [0.9 1.0]	[30]
n = 25 and p_a = 0.25	n = 25 and p_a = 0.25.	[48]
α = 1, p_a = 0.25, *nests* = 0.75	Not Applicable	[108]
Initial value α = 1, golden ratio φ = (1 + square root of 5)/2, *nests* = 0.75, p_a = 0.25, n = 30	α = 1, p_a = 0.25, n = 30	[29]
nest = 0.75, p_a = 0.25, α=1, golden ratio φ = (1 + square root of 5)/2	Not Applicable	[109]
α = 0.63, p_a = 0.42, α = 2.45, n=10	α = 0.63, p_a = 0.42, α = 2.45, n=10	[25]
α = 1	Not Applicable	[110]
p_a = 0.25	n = 25	[43]
nests = 500	Not Applicable	[107]
p_a initially set to 0.2, n = 8, length of replacement (L) is d/3 and p_a is 1/d, where d is the random probability	p_a = 0.05, β = 1	[108]
nest=4 (each contains 20 eggs),	n = 25, p_a = 0.25	[31]
Not Reported	Not Applicable	[39]
p_a = 0.25 and α = 1	p_a = 0.25, α = 1	[66]
p_a = 0.25, the size of complex value encoding *nests* = 20, size of real encoding *nests* = 40.	Not Reported	[67]
Colony size = 40, p_a = 0.25, number of runs with different speed = 30	p_a = 0.25	[64]
Not Reported	Not Applicable	[68]
n = 25-50, p_a = 0.25–0.5, Lèvy exponent β = 1 or1.5, α = 0.1	Not Applicable	[65]
Fraction of nest placed in top group = *0.25*, p_a=0.75, λ=0·24	Not Applicable	[114]
α = 0.1, p_a = 0.25,	Not Applicable	[44]
α = 0.1, p_a = 0.7, n = 10	Not Applicable	[69]
Not Reported	Not Reported	[70]
Not Reported	Not Reported	[71]
Pa = 0.25, α = 1, n = 25	p_a = 0.25, α = 1, n = 25	[73]
p_a and α not reported n = 25 and n = 10	p_a and α not reported n = 25 and n = 10	[72]
p_a = 0.25, α = 1, n = 40 divided into 4 clusters	p_a = 0.25, α = 1	[74]
p_a = 0.25	p_a = 0.25	[51]
p_a = 0.25	p_a = 0.25	[75]
n = 40, the maximum number of archives = 40 and 100, and the niche radius = 0.25, 0.05 and 0.1, λ=1.5	Not applicable	[76]
p_a = 0.25, n = 30	p_a = 0.25, n = 30	[77]
Crossover probability = 0.7 and mutation probability = 0.3	Not reported	[78]
n = 30, p_a adaptive	n = 30, p_a = 0.75	[50]
n = 50	n = 50	[81]
n = 100	n = 100	[79]
n = 100, p_a = 0.25, α = 0·01, update times of frog leaping searching group *Ne* = 10, group number m = 10	Not applicable	[52]
n = 25, p_a = 0.25	n = 25, p_a = 0.25	[82]
n = 50, p_a = 0.25, Elitism = 2	n = 50, p_a = 0.25	[53]
p_a = 0.25	p_a = 0.25	[83]
Pa = 0.5 (maximum) and *Pa* = 0.01 (minimum), α = 0.5 (maximum) and α = 0.005 (maximum)	p_a = 0.25, α = 0·01	[46]
n = 25, p_a = 0.25, α = 0·01, disturbance scope control quantity γ = [0.25, 1.5], the scale of learning and evolution is set = 0.75	n = 25, p_a = 0.25, α=0·01,	[54,55]
p_a = 0.25, n = 30	p_a = 0.25, n = 30	[84]
p_a = 0.25, n = 20, α = 1 (maximum) and α = 0.1 (maximum)	p_a = 0.25, n = 20, α = 1	[32]
p_a = 0.25, n = 20, α (minimum) and α (maximum) are chosen as 0.2∗domain and 0.8∗domain	p_a = 0.25, n = 20 α = 1	[33]
p_a = 0.25, n = 25, α = 1, jumping rate = 0.1	p_a = 0.25, n = 25, α = 1	[85]
p_a = 0.001(minimum) and p_a = 1 (maximum), α = 0.01 and α = 0.6, n = 65	p_a = 0.22, n = 75, α = 0.9	[34]
p_a = 0.25, n = 150	Not applicable	[86]
Not reported	α = 1	[35]
p_a = 0.25, n = 15, α = 0.01, w=0.06	p_a = 0.25, n = 15, α = 0.01,	[87]
Not reported	Not applicable	[88]
p_a = [0.7, 1], β = [1,2], α = 0.05, n = 100	α = 0.05, β = 1.7, n = 100, p_a = 0.75	[56]
p_a = 0.25, α=0.1	Not applicable	[89]
β = 0.9, α = 0.9, initial n = 5	Not reported	[36]
Not reported	Not reported	[80]
p_a = 0.25 and initial n = 10	p_a = 0.25, initial n = 10	[57]
p_a = 0.3, β = 1.7, α = 1.1 and learning regulator = 1.6	p_a = 0.25, β = 1.5, α = 0.01	[58]
Dynamic p_a, mutation and crossover	Not applicable	[59]
p_a = 0.25, n = 20 and α = 1	Not applicable	[90]
p_a = 0.25, n = 30 and α = adaptive	n = 30, p_a = 0.25 and α = 0.01	[37]
Kernel parameter = 0.5, p_a = adaptive, and α = adaptive	p_a = 0.25, and α = 0.01	[47]
p_a = 0.25 and n = 75	p_a = 0.25 and n = 50	[112]
P_a = 0.25, n = 25, location parameter = 0 and scale parameter = 1	Not applicable	[91]
Initial n = 200 and 10 maximum and minimum respectively, initial P_a = 0.1, initial α = 0.1, initial λ = 1.5, initial balancing probability = 0.8, initial elastic probability = 0.1, initial crossover rate = 0.9, initial s = 1.0 and initial scaling factor regulating the magnitude of change = 0.5	p_a = 0.25, λ = 1.5, n = 100 and α = 0.9	[60]
Radius coefficient = adaptive and n = adaptive	Radius coefficient = 3, n = 15	[40]
p_a = 0.25 and n = 25	p_a = 0.25, α = 1, λ = 1.5 and n = 25	[38]
Number of nodes = 30 and p_a = 0.2	Not applicable	[92]
n = 100, the ratio of top nests to abandoned nests = 25:75 and p_a = [0.1 to 0.9 with a step of 1]	Not applicable	[113]
p_a = 0.25, n = 50 and elitism number = 2,	p_a = 0.25 and n = 50	[61]
p_a = 0.25	p_a = 0.01	[61,53,62,63]

Table 10
The optimal parameter settings of MoBCS and BCS used to converge towards the optimal solution.

MoBCS	BCS	Reference
$\alpha = 0.1$, $p_a = 0.25$	Not Applicable	[100]
nests = 25, $\alpha = 1$	Not Applicable	[95]
$n = 5$, $p_a = 0.00$, number of evolution = 6	Not Applicable	[42]
$n = 20$, $p_a = 0.2$, $Pc = 0.6$ Portion of intelligent cuckoos	$n = 20$, $p_a = 0.2$	[41]
$p_a = 25\%$, $n = 40$	Not Applicable	[94]
Not reported	Not applicable	[101]
Probability of mutation (p_m) = 0.15, $n = 10$	Not applicable	[96]
Not applicable	$p_a = 0.25$, $p_m = 0.15$	[97]
Not Applicable	$p_a = 0.65$, n=25 and n=30	[102]
Not applicable	n=5	[103]
Not applicable	$p_a = 0.25$, $n = 10$, partially matched and multi-parental sorting crossover = 0.7, feasible geometric mutation = 0.9	[98]
Not applicable	$n = 50$, $p_a = 0.25$, probability of swap and insert = 0.1	[54,55]
Not applicable	$\lambda = 2$ and $\alpha = 10^6$ *x where, x denotes the number of jobs.	[104]
Not applicable	$\alpha = 0.01$, $p_a = 0.25$, n=15, U_b=1, L_b=0	[105]
Not applicable	$Pa = 0.25$; α=0.1	[89]
Not applicable	Factors for reducing the scale of a particular iteration (P_k and P_s), p_a, α, and λ are adaptive.	[106]
Not applicable	$n = 50$, $p_a = 0.25$ and probability of generalized opposition-based learning = 0.1	[61,53,62,63]
$Pa = 0.25$, $n = 200$, $\alpha = 0.01$, size of candidate nests = 40, and repository size = 30	Not applicable	[99]

solution distribution along the Pareto front [86]. RPD in% is the difference between the best known optimum value and the best optimum value converged by the CS [105]. Cluster objective function (JC) is the sum of the Euclidean distances of the objects to their centers, which is the clustering objective function. The objective is to find the centers of the clusters by minimizing the JC [87]. The Average Relative Difference (ARD) measured the performance of an algorithm, the smaller the value of ARD, the better the algorithm [61,53,62,63]. Normalized Mutual Information (NMI) is often used for measuring the quality of community detection by an algorithm. The NMI measured the similarity between the actual partition and detected partition (Zhou et al. [133]).

In Table 11, to demonstrate improvements made as a result of CS modifications, examples of quality solutions of the reviewed studies arec presented for both the continuous and binary CS. Tables 12 and 13 report examples of convergence speeds for the CS and BCS respectively. The *metric* in Tables 11–13 denotes quality measures used by researchers of the specific CS. Each of the studies has a preferred metric for assessing the solution of the problem being solved since there is no best performance indicator for all types of problems [29].

In Table 12, similar analysis is conducted for CS papers according to the convergence speed. Only a comparison between CS and MoCS is taken into account. Also, the measure is denoted by column metric. They are also presented in Table 13 for MoBCS and BCS.

6.4. Performance study

This section calculates any improvement in modification that is applied to the original CS algorithm. Each paper in the study is written using the non-standard measures. Therefore, comparing the achieved results between papers is impossible and unfair. However, it is possible to estimate any improvements in modification regarding the original CS algorithm. In line with this, the following measures are defined:

$$I(\%) = \frac{CPUTime_{CS} - CPUTime_{MoCS}}{CPUTime_{CS}} x100 \tag{11}$$

where I, $CPU\ Time_{MoCS}$ and $CPU\ Time_{CS}$ are percentage improvement and convergence speed required by MoCS and CS to converge towards the optimal solution. Eq. (12) is used for computing the accuracy percentage improvement in some studies in Table 11.

$$Accuracy(\%) = \frac{E_{CS} - E_{MoCS}}{E_{CS}} x100, \tag{12}$$

where E_{MoCS} and E_{CS} are values of the metrics for MoCS and CS, respectively. In reality, the performances between papers were not measured because of the constraints discussed earlier. However, the impact of the modification on CS in each paper is computed using Eqs. (11)–(12). Though, the numerator of Eq. (12) is reverse to suit the computation in [57,34,66] because their performance metrics differ with that of other studies (refer to Table 15). Table 14 shows percentage improvements based on values extracted from Table 12. Several studies had not compared the effectiveness of their modification with the CS, while some researchers did. This made it difficult to compute the impact of the CS modifications for the studies that did not compare their modifications with CS. The performance of the MoCS should have been compared to the original CS for measuring the impacts of the modifications before further comparing with other metaheuristic algorithms as done in [43,41,25]. The convergence speed improvement made by MoCS over CS is computed using Eq. (11) and is presented in the column 'Improvement' in Table 14 in bold.

The convergence speed improvements as achieved in the studies (Table 14) could not be compared because of hardware, which plays a crucial role in the convergence speeds of the algorithms. The algorithms (CS and MoCS) are run on different systems with varying configurations, e.g., processor speed, memory capacity, etc. Thus, it would not have been fair to compare the algorithms' execution times which are run on different systems with different hardware specifications. It can be observed that the system configurations in Table 14 differed between each study, whereas others had not reported system configurations. Therefore, the systems had different performances, depending on the configurations. This is difficult when comparing the convergence speeds among the studies since the algorithms are run on systems with different performances. The modification made by Chiroma et al. [73] achieved the highest percentage of advancement. The lowest is recorded by Wang et al. [61] with no improvement over the CS. Hence, conclusions cannot be made about which of those MoCSs had the better convergence speeds among the different studies because of the hardware variability among the different studies as already explained. However,

Table 11
Comparing performance quality between the MoCS and CS, and between the MoBCS and BCS.

Metric	MoCS	CS	MoBCS	BCS	Reference
BCA	98.45%	97.99	Nil	Nil	[45]
BF	Graph presentation	Graph presentation	Nil	Nil	[30]
BF	4.63E – 28	8.31E – 26	Nil	Nil	[48]
APCA	99.57	Not Applicable	Nil	Nil	[108]
PCA	93.58	Not Applicable	Nil	Nil	[109]
Makespan	2336	2357	Nil	Nil	[25]
MSE	0.00001905	Not applicable	Nil	Nil	[110]
SR	Nil	Nil	Not Applicable	103.26%	[94]
MBF	5.5768E – 147	3.1447E – 010	Nil	Nil	[43]
PDbest	Nil	Nil	5.34	0	[41]
Minimization of energy demand, cost, CO_2 emission	151.38 (KWh), 65262.92 (PLN), and 1984.66 (Kg/a)	Not Applicable	Nil	Nil	[107]
MCA	Nil	Nil	Not Applicable	97.33%	[100]
BA	80.02	73.33	Nil	Nil	[49]
SR	Nil	Nil	1000	Not Applicable	[95]
SR	100%	81%	Nil	Nil	[31]
BF	Nil	Nil	100	Not Applicable	[42]
BF	-3.0666E + 004	Not Applicable	Nil	Nil	[39]
BF	25.4291	1.4446E + 003	Nil	Nil	[66]
BF	8.9793E – 004	0.1141	Nil	Nil	[67]
BF	0	3.0716E – 11	Nil	Nil	[64]
BF	1.17E – 07	Not Applicable	Nil	Nil	[65]
SR	100%	Not Applicable	Nil	Nil	[68]
BF	Graph presentation	Graph presentation	Nil	Nil	[44]
PL	4.5	Not Applicable	Nil	Nil	[69]
Minimization of generation fuel cost	802.0347	802.7283	Nil	Nil	[70]
MSE	0.000060	0.000082	Nil	Nil	[71]
MSE	2.15E – 10	0.0000097	Nil	Nil	[73]
Minimized cooling load	583.923	672.824	Nil	Nil	[72]
Makespan	Nil	Nil	Not applicable	0.0	[101]
SR	100%	100%	Nil	Nil	[74]
SR and GSR	Graphical representation	Graphical representation	Nil	Nil	[51]
SR	100%	0%	Nil	Nil	[75]
BF	Nil	Nil	Not applicable	14807	[96]
SR	Nil	Nil	Not applicable	95%	[97]
Makespan	Nil	Nil	Not applicable	25	[102]
UI	0.0113	Not applicable	Nil	Nil	[76]
BF	0	3.7937	Nil	Nil	[77]
BF	1,415.48	1,422.23	Nil	Nil	[78]
MBF	3.23E – 10	1.81E – 04	Nil	Nil	[50]
MBF	0.9980	0.9981	Nil	Nil	[81]
BF	6.10	6.26	Nil	Nil	[79]
BF	0	Not applicable	Nil	Nil	[52]
SR	Nil	Nil	Nil	20/20	[103]
BF	7.5188e – 31	4.846e – 10	Nil	Nil	[82]
SR	Nil	Nil	Nil	100	[98]
BF	1.00	39.82	Nil	Nil	[53]
MSE	0.0146	0.0390	Nil	Ni	[83]
BF	1.7684e – 004	0.007466	Nil	Nil	[46]
PRD	Nil	Nil	Not applicable	0.34	[54,55]
BF	3.6450e – 012	0.0382	Nil	Nil	[54,55]
BF	9.9984	10.0000	Nil	Nil	[84]
BF	Graphical representation	Graphical representation	Nil	Nil	[32]
BF	Not reported	Not reported	Nil	Nil	[33]
SR	1	Not reported	Nil	Nil	[85]
DR	0.013165	0.00834	Nil	Nil	[34]
Diversity	0.53539	Not applicable	Nil	Nil	[86]
Makespan	Nil	Nil	Not applicable	1.001	[104]
ABF	1.80e-08	Not applicable	Nil	Nil	[35]
RPD	Nil	Nil	Not applicable	0	[105]
JC	232.20	271.52	Nil	Nil	[87]
Profits	$580101.90	Not applicable	Nil	Nil	[88]
BF	4.44089E – 18	4.64E – 12	Nil	Nil	[56]
SR	100%	Not applicable	Not applicable	100%	[89]
BF	0	1.277641345977e–003	Nil	Nil	[36]
BF	7.3835	8.3732	Nil	Nil	[80]
BF	88.5799	29.1706	Nil	Nil	[57]
BF	0	2.38E – 01	Nil	Nil	[58]
AMBE	0.1295	Not applicable	Nil	Nil	[59]
BF	0.0126654	Not applicable	Nil	Nil	[90]
BF	1.17E – 11	5.12E – 05	Nil	Nil	[37]
Mean Fitness	1.52E – 10	2.74E + 00	Nil	Nil	[47]
BF	0.6	0.675	Nil	Nil	[112]
BF	Nil	Nil	Not applicable	35,534	[106]

Table 11 (*Continued*)

Metric	MoCS	CS	MoBCS	BCS	Reference
BF	1.0418	Not applicable	Nil	Nil	[91]
BF	0.00E+0	1.95E+7	Nil	Nil	[60]
BF	0	Not applicable	Nil	Nil	[40]
BF	0.2631	1.7152	Nil	Nil	[38]
% of power loss reduction	56.16	Not applicable	Nil	Nil	[92]
Minimum cost	922773.6	Not applicable	Nil	Nil	[113]
ARD	Nil	Nil	Not applicable	0.00	[61,53,62,63]
BF	1.00	70.83	Nil	Nil	[61]
BF	0.000908	0.001228	Nil	Nil	[61,53,62,63]
NMI	Nil	Nil	1	Not applicable	[99]

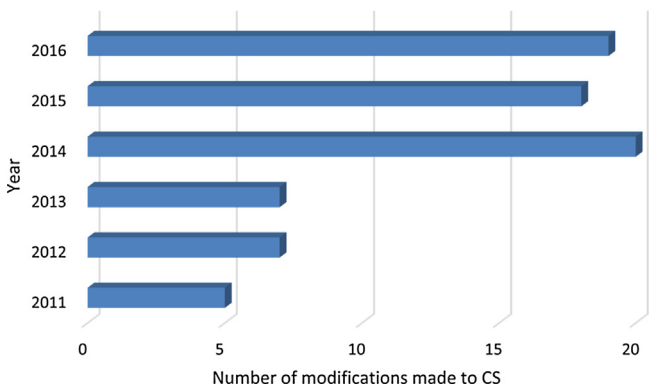

Fig. 4. Trends of modifications on CS from inception to 2016.

the improvements made by each of the MoCSs are computed as shown in Table 14.

In Table 15, the computation of percentage improvement in accuracy for the studies in [45,31,49,74,75] is the difference between the% accuracy. Table 15 indicates that the highest advancement in accuracy recorded is 100% whereas the minimum achieved is 0. Tables 11 and 13, with the exception of Ouaarab et al. [41], indicate that it is difficult to compute the percentage improvement made by the MoBCS over BCS. This is because the performances obtained for the MoBCS were not compared directly with the performances of the BCS. The effectiveness of the modification could have been measured if the comparison were made between the original algorithm and the proposed modified version. From Tables 11 and 13, the percentage improvement in accuracy was computed using Eq. (12). The computed accuracy indicated that the MoBCS proposed by Ouaarab et al. [41] achieved 100% improvement over the BCS. The improvement of the convergence speed computed using Eq. (11) recorded 58.65% advancement over the BCS.

6.5. Trends in the modifications to the cuckoo search algorithm

As can be seen clearly in Fig. 4, there had been recent advances on the modifications of the CS, where the highest number of modifications is observed from 2014 to 2016. This trend shows how CS is currently attracting attention and may likely continue to do so into the future, as argued by Yang and Deb [6]. The lowest number of modifications on CS so far were recorded in 2011 and this is not surprising because that is the year the CS modification started. Fig. 4 shows that modifications on CS started less than two years after its inception and have continued to date. Generally, the trend (see Fig. 4) shows increasing numbers of modifications on the CS.

On average, there had been an average of 12.7 modifications to CS each year from 2011 to 2015. The MCS proposed by Walton et al. [29] attracted the attentions of researchers more than the modifications proposed by other researchers (refer to Section 5.4). The modified CS is found to be implemented successfully for solving problems within different domains. This could possibly be attributed to the performance, easy understanding and implementation of the modified concept proposed by Walton et al. [29]. This corroborates the statement of Yang [3] that the Walton's et al. MCS has proved to be robust and effective in solving nonlinear problems, including the generation of meshes.

7. Open research Problems/Challenges

Big data analytics: The modifications of the CS as shown in the literature, focus mainly on evaluating the effectiveness and efficiency of the modified CS on a small data set without considering the big data which is the modern day reality of data. However, big data have attracted concern from both researchers and industries. It is estimated that about 2.5 quintillion bytes of data are being generated every day [127]. The major challenge is the discovery of knowledge from the very large data set using novel approaches [128]. Therefore, the excellent performance of the modified CS as noted in the literature is only limited to a small data set, not big data.

CS Sensitivity to Parameters: Despite the fact that the CS has less sensitivity to parameter settings compared to other metaheuristic algorithms, parameters settings are still an issue because user defined parameters are required for implementing the CS. This is because there is no guarantee on the parameters that can work on any problem successfully. As such, the success of the CS still relies on parameter settings which, thereby, limits the efficiency and effectiveness of the CS. An effective algorithm is required to converge to the optimal solution to a problem with a minimal effort as argued by Yang and He [129].

The parameter free CS: This means the CS that does not requires the setting of parameters and its performance does not depend on parameter settings/tuning. The literature review has shown that no attempt was made by researchers to propose a parameter free CS despite the great attention received by CS from researchers. Setting of parameters for the CS requires great human effort of a laborious trial and error approach to realize the best parameters settings. This is because the systematic method for realizing the optimal parameters values remain an open research problem.

The exploration and exploitation: The balancing of exploration and exploitation of the CS algorithm remains a challenging issue because the best systematic way to control exploration and exploitation in CS remains an open research problem that is worth solution. Despite the overwhelming modifications made to CS, an attempt has not been made by researchers to propose a systematic approach of balancing exploitation and exploration of the CS. Though CS is considered as one of the bio-inspired algorithms with good balancing between exploitation and exploration, the CS still has room for improving its exploitation and exploration balancing. Yang and He [129] argued that a proper balance of the exploitation and exploration is crucial in ensuring the effectiveness and efficiency of an algorithm.

Table 12
Comparison of convergence speeds between MoCS and CS.

Metric	MoCS	CS	Reference
EAS	500	750	[45]
EAS	Not Reported	Not Reported	[30]
Not reported	Not Reported	Not Reported	[48]
Not reported	Not Reported	Not Applicable	[108]
CPU time [sec]	Graphical representation	Graphical representation	[29]
Not reported	Not Reported	Not Reported	[109]
CPU time [sec]	4.44	4.61	[25]
CPU time	27.8524	Not Applicable	[110]
Not applicable	Not Applicable	Not Applicable	[43]
EAS	50–80 iterations	Not Applicable	[107]
CPU time [sec]	1000	1000	[49]
Not reported	Not Reported	Not Reported	[31]
CPU time [sec]	Not Reported	Not Applicable	[39]
CPU time [sec]	13.95	20.15	[66]
CPU time [sec]	Graphical representation	Graphical representation	[67]
CPU time [sec]	710.7	4509.6	[64]
Not reported	Not Reported	Not Reported	[68]
EAS	Graphical representation	Not applicable	[65]
EAS	Graphical representation	Graphical representation	[44]
CPU time [sec]	15	Not applicable	[69]
CPU time [sec]	17.9948	23.3948	[70]
Not reported	Not Reported	Not Reported	[71]
CPU time [sec]	0.35	27.23	[73]
CPU time [sec]	Not Reported	Not Reported	[72]
CPU time [sec]	Not Reported	Not Reported	[74]
NFE	Graphical representation	Graphical representation	[51]
NFE	36270	77490	[75]
CPU time [time]	0.004	Not applicable	[76]
			[89]
EAS	4000	8000	[77]
CPU time [sec]	Graphical representation	Graphical representation	[78]
EAS	Graphical representation	Graphical representation	[50]
EAS	Graphical representation	Graphical representation	[81]
CPU time [sec]	49.44	84.55	[79]
CPU time [sec]	Not reported	Not applicable	[52]
CPU time [sec]	Not reported	Not applicable	[82]
EAS	Graphical representation	Graphical representation	[53]
CPU time [sec]	Not reported	Not applicable	[83]
EAS	Graphical representation	Graphical representation	[46]
EAS	Graphical representation	Graphical representation	[54,55]
EAS	Graphical representation	Graphical representation	[84]
EAS	Graphical representation	Graphical representation	[32]
EAS	1087	1690	[33]
Relative success performance	0.941	No specific value reported	[85]
CPU time [time]	Graphical representation	Graphical representation	[34]
CPU time [time]	Not applicable	Not applicable	[86]
CPU time [time]	31.77	34.46	[35]
CPU time [time]	Not applicable	Not applicable	[87]
EAS	40	Not applicable	[88]
EAS	Graphical representation	Graphical representation	[56]
EAS	Graphical representation	Not applicable	[89]
EAS	Graphical representation	Graphical representation	[36]
CPU time [time]	36.55	98.56	[80]
EAS	Graphical representation	Graphical representation	[57]
EAS	Graphical representation	Graphical representation	[58]
EAS	Not reported	Not reported	[59]
EAS	Not reported	Not reported	[90]
EAS	Graphical representation	Graphical representation	[37]
CPU time [time]	Graphical representation	Graphical representation	[47]
CPU time [time]	24.06	46.07	[112]
EAS	Graphical representation	Graphical representation	[91]
Time complexity	21.89	Graphical representation	[60]
EAS	Not reported	Not reported	[40]
CPU time [time]	8.3765	10.4827	[38]
CPU time [time]	Not reported	Not reported	[92]
CPU time [time]	234	Not applicable	[113]
CPU time [time]	1.86	1.33	[61]
CPU time [time]	Not reported	Not reported	[61,53,62,63]

Statistical validation of results: The attention of the knowledge engineering community was drawn to the need for statistical validation of results because of real/world applications of the algorithms [130]. In spite of the need for the statistical validation of results, researchers that proposed modifications of the CS did not use inferential statistics to find out the significant difference between the performance of the modified CS and the original CS. The typical practice by the researchers is to estimate the error and convergence time between the modified CS and the original CS. The modified CS with the lowest error and convergence time is cho-

Table 13
Convergence speeds between the MoBCS and BCS.

Metric	MoBCS	BCS	Reference
Not Applicable	Not applicable	Not Applicable	[94]
CPU time [sec]	674.715332	Not Applicable	[100]
EAS	0	Not Applicable	[95]
CPU time [sec]	0.134 s	Not Applicable	[42]
CPU time [sec]	6.53	2.70	[41]
Makespan	0.0	Not applicable	[101]
CPU time [sec]	Not applicable	Not applicable	[96]
CPU time [sec]	Not applicable	1.719	[97]
CPU time [sec]	Not applicable	0.02	[102]
CPU time [sec]	Not applicable	0.4	[103]
CPU time [sec]	Not applicable	1.005	[98]
CPU time [sec]	Not applicable	1.61	[54,55]
CPU time [sec]	Not applicable	0.020	[104]
EAS	Graphical representation	Graphical representation	[105]
EAS	Not applicable	Graphical representation	[89]
CPU time [sec]	Not applicable	60.14	[106]
CPU time [sec]	Not applicable	Graphical representation	[61,53,62,63]
CPU time [time]	Not applicable	Not applicable	[99]

Table 14
Percentage improvement of convergence speed made by MoCS over CS.

MoCS	CS	System Configuration	Improvement (%)	Reference
500	750	Not Reported	**33.33**	[45]
4.44	4.61	C+ +, Pentium 4 3.4 GHz PC with 512 MB memory	**3.688**	[25]
1000	1000	Not Reported	**0**	[49]
13.95	20.15	Matlab 7.1, CPU T3100, 1.90GHZ with 2GB memory capacity	**30.77**	[66]
710.7	4509.6	Java, Intel I7-2600k processor with 8GB of RAM on Windows 7 × 64 OS and NetBeans 7.0.1 IDE	**80.24**	[64]
17.9948	23.3948	Not reported	**23.o8**	[70]
0.35	27.23	Intel Core (TM) 2 Quad CPU 2.33 GHz, RAM 2GB, 32-bit operating system	**98.71**	[73]
36270	77490	Not reported	**53.19**	[75]
4000	8000	Simulation is taken on MATLAB 7.8 under Windows XP operating system, run on PC with the processor of 5000 Daul Core 2.61 GHz and the memory of 768MB	**50**	[77]
49.44	84.55	under 64-bit Linux Mint 14: Nadia operating system. Experiments are conducted on a desktop with Intel_ CoreTM i7- 2600 CPU@3.40 GHz x 4, and 12GB of RAM	**41.53**	[79]
1087	1690	Not reported	**35.68**	[33]
31.77	34.46	Not reported	**7.81**	[35]
36.55	98.56	Not reported	**62.92**	[80]
24.06	46.07	Intel Core i5 CPU, 2.6 GHZ processor and 4GB RAM		[112]
8.38	10.48	Not reported	**20.04**	[38]
1.86	1.33	Not reported	**No improvement**	[61]

sen as the best algorithm. However, estimation error could likely be responsible for the simple observed difference. A statistical test that has a confidence bound is vital to determine the best algorithm between the modified CS and the original CS. This will surely show that the observed difference between the modified CS and the original CS is not a mere chance effect from the process of estimation.

Representation of result: Researchers focus mainly on modifying the CS to work with binary and real-values while the complex number is ignored. Thus, the CS is limited to working with real-value and binary. As such, the result representation of the CS is limited to only binary and real-value as indicated clearly in Fig. 2.

8. Future research directions

In the future, researchers should modify the CS by parallelizing the CS to work on the Hadoop platform. Parallelizing the CS to work on the Hadoop platform will allow application of the CS in big data analytics. To improve general usage of the CS in solving real/life engineering problems, the sensitivity of the CS to the parameter settings should be eliminated, while maintaining the performance of the CS. This will result in a parameter free CS similar to a parameter free Bat algorithm that does not require user intervention to specify parameters [131]. Adding control parameters to CS as practiced by some researchers [59,132,58], should be highly discouraged because it increases CS reliance on parameters settings. To improve the searching capabilities of the CS, a novel systematic way of controlling the balancing between exploitation and exploration of the CS should be proposed by researchers. To increase the general applications of the CS, the CS can be modified to work with complex numbers. As such, it represents results in complex numbers.

9. Conclusions

This paper presented a review of the recent advances in the modifications of the original CS. The percentages of performance improvements made by MoCS over the original CS were computed for several studies, except for studies which had not reported the numerical values of their performance metric. It is found that it is hard (or even impossible) to compare performances of modifications on CS between different studies, because the MoCS are run on different benchmarks and under different conditions. The parameter values reported in the study could provide a guide for future researchers when choosing the optimal parameter values of the CS. This review has revealed that $p_a = 0.25$ and $\alpha = 1$ are the

Table 15
Advancement of solution quality by MoCS over CS.

Metric	MoCS	CS	Improvement (%)	Reference
BCA	98.45%	97.99%	**0.46**	[45]
BF	4.63E − 28	8.31E − 26	**99.44**	[48]
Makespan	2336	2357	**0.89**	[25]
MBF	5.5768E − 147	3.1447E − 010	**100**	[44]
BA	80.02%	73.33%	**6.69**	[49]
BF	100%	81%	**19**	[31]
BF	25.4291	1.4446E + 003	**98.24**	[66]
BF	8.9793E − 4	0.1141	**99.21**	[67]
BF	0	3.0716E − 11	**100**	[64]
MSE	0.000060	0.000082	**26.83**	[71]
MSE	2.15E − 10	0.0000097	**99.99**	[73]
Minimized cooling load	583.923	672.824	**13.21**	[72]
SR	100%	100%	**0**	[74]
SR	100%	0%	**100**	[75]
BF	0	3.7937	**100**	[77]
MBF	3.23E − 10	1.81E − 04	**99.99**	[50]
MBF	0.9980	0.9981	**0.01**	[81]
BF	6.10	6.26	**2.5**	[79]
BF	7.5188e − 31	4.846e − 10	**99.99**	[82]
BF	1.00	39.82	**97.49**	[53]
MSE	0.0146	0.0390	**62.56**	[83]
BF	1.7684e − 004	0.007466	**97.63**	[46]
BF	3.6450*e* − 012	0.0382	**99.99**	[54,55]
BF	9.9984	10	**0.016**	[84]
DR	0.013165	0.00834	**36.65**	[34]
BF	4.44089E − 18	4.64E − 12	**99.99**	[56]
BF	0	1.277641345977e − 003	**100**	[36]
BF	7.3835	8.3732	**11.82**	[80]
BF	88.5799	29.1706	**67.07**	[57]
BF	0	2.38E − 01	**100**	[58]
BF	1.17E − 11	5.12E − 05	**99.99**	[37]
Mean Fitness	1.52E − 10	2.74E + 00	**99.99**	[47]
BF	0.6	0.675	**11.11**	[112]
BF	0	1.95E + 7	**100**	[60]
BF	0.2631	1.7152	**84.66**	[38]
BF	1.00	70.83	**98.59**	[61]
BF	0.000908	0.001228	**26.06**	[61,53,62,63]

optimal parameter settings that are used successfully for implementing both original and modified CS across different domains. Hence, this paper recommends these values as the standard CS parameter values.

It was found that the modification of CS based on population reduction and usage of biased random walk is the appropriate variant of CS for solving mesh and energy efficiency problems. The Inter species binary CS with different Lévy flight is appropriate for solving scheduling problems. The multi objective CS is suitable for solving engineering design problems such as welding and disc brake. The CS with adaptive parameters is suitable for the optimization and stability analysis of thermal power systems. The modified CS has shown to be a promising and robust alternative or even replacement for existing algorithms in solving engineering problems.

Each year starting from 2011, the CS has been modified by researchers to improve its efficiency and effectiveness in solving optimization problems, and this trend could probably continue into the future. This is because of the great attention CS is receiving from the research community and its proven performance in solving engineering problems. It is evident from the depicted trend that the modifications of CS are growing promptly, because 39 out of 58 research studies published on CS modifications are within 2014–2015. This review can assist expert researchers by developing the future modifications of the CS, as well as the novice researchers in searching for the proper modifications of the original CS.

Acknowledgement

This research is supported by International Islamic University Malaysia.

References

[1] X.S. Yang, Nature-inspired Metaheuristic Algorithms, Luniver press, UK, 2010.
[2] X.S. Yang, S. Deb, S. Fong, Accelerated particle swarm optimization and support vector machine for business optimization and applications Networked Digital Technologies, vol. 538, Springer Berlin Heidelberg, 2011, pp. 53–66.
[3] X.S. Yang, Cuckoo search and firefly algorithm: overview and analysis Cuckoo Search and Firefly Algorithm, vol. 610, Springer International Publishing, 2014, pp. 1–26.
[4] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, in: Proceedings of World Congress on Nature & Biologically Inspired Computing, 2009 (NaBIC 2009), IEEE Press, 2009, pp. 210–214.
[5] X.S. Yang, S. Deb, Engineering optimisation by cuckoo search, Int. J. Math. Modell. Num. Optim. 1 (4) (2010) 330–343.
[6] X.S. Yang, S. Deb, Cuckoo search: recent advances and applications, Neural Comput. Appl. 24 (1) (2014) 169–174.
[7] X. Sun, W. Sun, J. Wang, Y. Zhang, Y. Gao, Using a Grey–Markov model optimized by Cuckoo search algorithm to forecast the annual foreign tourist arrivals to China, Tour. Manag. 52 (2016) 369–379.
[8] N.M. Nawi, A. Khan, M.Z. Rehman, T. Herawan, M.M. Deris, CSLMEN: a new cuckoo search levenberg marquardt elman network for data classification Recent Advances on Soft Computing and Data Mining, vol. 287, Springer International Publishing, 2014, pp. 173–182.
[9] N.M. Nawi, A. Khan, M.Z. Rehman, A new optimized cuckoo search recurrent neural network (CSRNN) algorithm, in: The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications, Springer, Singapore, 2014, pp. 335–341.
[10] S.A. Elazim, E.S. Ali, Optimal power system stabilizers design via cuckoo search algorithm, Int. J. Electr. Power Energy Syst. 75 (2016) 99–107.
[11] L.A.M. Pereira, D. Rodrigues, T.N.S. Almeida, C.C.O. Ramos, A.N. Souza, X.S. Yang, J.P. Papa, A binary cuckoo search and its application for feature selection Cuckoo Search and Firefly Algorithm, vol. 516, Springer International Publishing, 2014, pp. 141–154.
[12] I. Fister, D. Strnad, X.S. Yang, I. Fister Jr., Adaptation and hybridization in nature-inspired algorithms Adaptation and Hybridization in Computational Intelligence, vol. 18, Springer International Publishing, 2015, pp. 3–50.

[13] I. Fister Jr., X.S. Yang, D. Fister, I. Fister, Cuckoo search: a brief literature review Cuckoo Search and Firefly Algorithm, vol. 516, Springer International Publishing, 2014, pp. 49–62.
[14] P. Civicioglu, E. Besdok, A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artif. Intell. Rev. 39 (4) (2013) 315–346.
[15] B. Xing, W.J. Gao, T. Marwla, An overview of cuckoo-inspired intelligent algorithms and their applications, in: Proceedings of 2013 IEEE Symposium on Swarm Intelligence (SIS), IEEE Press, 2013, pp. 85–89.
[16] I. Fister Jr., D. Fister, I. Fister, A comprehensive review of cuckoo search: variants and hybrids, Int. J. Math. Modell. Num. Optim. 4 (4) (2013) 387–409.
[17] I. Fister Jr., X.S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, Elektrotehniški Vestnik 80 (3) (2013) 116–122.
[18] S. Walton, O. Hassan, K. Morgan, M.R. Brown, A Review of the Applications of the Cuckoo Search Algorithm. Swarm Intelligence and Bio-Inspired Computation, Elsevier, 2013, pp. 258–270, Chapter 11.
[19] S. Kamat, A.G. Karegowda, A brief survey on cuckoo search applications, Int. J. Innov. Res. Comput. Commun. Eng. 2 (2) (2014) 7–14.
[20] R.B. Payne, M.D. Sorenson, K. Klitz, The Cuckoos, Oxford University Press, 2005.
[21] N.B. Davies, M.D.L. Brooke, An experimental study of co-evolution between the cuckoo, Cuculus canorus, and its hosts. I. Host egg discrimination, J. Anim. Ecol. (1989) 207–224.
[22] C.T. Brown, L.S. Liebovitch, R. Glendon, Lévy flights in Dobe Ju/'hoansi foraging patterns, Hum. Ecol. 35 (1) (2007) 129–138.
[23] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, J. Comput. Phys. 226 (2) (2007) 1830–1844.
[24] S. Walton, O. Hassan, K. Morgan, Selected engineering applications of gradient free optimisation using cuckoo search and proper orthogonal decomposition, Arch. Comput. Methods Eng. 20 (2) (2013) 123–154.
[25] M.K. Marichelvam, T. Prabaharan, X.S. Yang, Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan, Appl. Soft Comput. 19 (2014) 93–101.
[26] A. Liberati, D.G. Altman, J. Tetzlaff, C. Mulrow, P.C. Gøtzsche, et al., The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration, Ann. Intern. Med. 151 (4) (2009) W–65.
[27] K.J. Aaltonen, L.M. Virkki, A. Malmivaara, Y.T. Konttinen, D.C. Nordstrom, M. Blom, Systematic review and meta-analysis of the efficacy and safety of existing TNF blocking agents in treatment of rheumatoid arthritis, PLoS One 7 (1) (2012) e30275.
[28] L. Hartling, C. Spooner, L. Tjosvold, A. Oswald, Problem-based learning in pre-clinical medical education: 22 years of outcome research, Med. Teach. 32 (1) (2010) 28–35.
[29] S. Walton, O. Hassan, K. Morgan, M.R. Brown, Modified cuckoo search: a new gradient free optimisation algorithm, Chaos Solit. Fract. 44 (9) (2011) 710–718.
[30] K.A. Rani, W.F. Hoon, M.F. Abd Malek, N.A.M. Affendi, L. Mohamed, N. Saudin, et al., Modified cuckoo search algorithm in weighted sum optimization for linear antenna array synthesis, Proceedings of 2012 IEEE Symposium on Wireless Technology and Applications (ISWTA) (2012) 210–215.
[31] Y. Zhang, L. Wang, Q. Wu, Modified Adaptive Cuckoo Search (MACS) algorithm and formal description for global optimisation, Int. J. Comput. Appl. Technol. 44 (2) (2012) 73–79.
[32] Z. Zhang, Y. Chen, An improved cuckoo search algorithm with adaptive method, in: Proceedings of 2014 Seventh International Joint Conference on Computational Sciences and Optimization (CSO), IEEE Press, 2014, pp. 204–207.
[33] P. Ong, Adaptive cuckoo search algorithm for unconstrained optimization, Sci. World J. 2014 (2014) 8, http://dx.doi.org/10.1155/2014/943403, Hindawi Publishing Corporation, The Scientific World Journal Article ID 943403.
[34] S. Rangasamy, P. Manickam, Stability analysis of multimachine thermal power systems using nature inspired modified cuckoo search algorithm, Turk. J. Electr. Eng. Comput. Sci. 22 (2013) 1099–1115.
[35] P. Ong, Z. Zainuddin, C.K. Sia, B.A.M. Zain, Adaptive cuckoo search algorithm with two-Parent crossover for solving optimization problems, in: Advances in Swarm and Computational Intelligence, Springer International Publishing, 2015, pp. 427–435.
[36] M. Abdollahi, A. Bouyer, D. Abdollahi, Improved cuckoo optimization algorithm for solving systems of nonlinear equations, J. Supercomput. 72 (3) (2016) 1246–1269.
[37] L. Huang, S. Ding, S. Yu, J. Wang, K. Lu, Chaos-enhanced Cuckoo search optimization algorithms for global optimization, Appl. Math. Modell. 40 (5) (2016) 3860–3875.
[38] M.K. Naik, R. Panda, A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition, Appl. Soft Comput. 38 (2016) 661–675.
[39] H. Zheng, Y. Zhou, A cooperative coevolutionary cuckoo search algorithm for optimization problem, J. Appl. Math. (2013), http://dx.doi.org/10.1155/2013/912056, Article ID 912056.
[40] S.A. Mohseni, T. Wong, V. Duchaine, MCOA: mutated and self-adaptive cuckoo optimization algorithm, Evolution. Intell. 9 (1–2) (2016) 21–36.
[41] A. Ouaarab, B. Ahiod, X.S. Yang, Discrete cuckoo search algorithm for the travelling salesman problem, Neural Comput. Appl. 24 (7-8) (2014) 1659–1669.
[42] G.K. Jati, H.M. Manurung, S. Suyanto, Discrete cuckoo search for traveling salesman problem, in: Proceedings of 2012 7th International Conference on Computing and Convergence Technology (ICCCT), IEEE Press, 2012, pp. 993–997.
[43] X. Li, J. Wang, M. Yin, Enhancing the performance of cuckoo search algorithm using orthogonal learning method, Neural Comput. Appl. 24 (6) (2014) 1233–1247.
[44] J.H. Lin, H.C. Lee, Emotional chaotic cuckoo search for the reconstruction of chaotic dynamics, Proceedings of 11th WSEAS Int. Conf. on Computational Intelligence, Man-machine Systems and Cybernetics (CIMMACS'12) (2012) 123–128.
[45] E. Valian, S. Mohanna, S. Tavakoli, Improved cuckoo search algorithm for feedforward neural network training, Int. J. Artif. Intell. Appl. 2 (3) (2011) 36–43.
[46] J. Wang, B. Zhou, A hybrid adaptive cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation, Neural Comput. Appl. 27 (6) (2016) 1511–1517.
[47] B. Jia, B. Yu, Q. Wu, X. Yang, C. Wei, R. Law, S. Fu, Hybrid local diffusion maps and improved cuckoo search algorithm for multiclass dataset analysis, Neurocomputing 189 (2016) 106–116.
[48] M. Tuba, M. Subotic, N. Stanarevic, Modified cuckoo search algorithm for unconstrained optimization problems, in: Proceedings of the 5th European Conference on European Computing Conference, WSEAS Press, 2011, pp. 263–268.
[49] A. Saelim, S. Rasmequan, P. Kulkasem, K. Chinnasarn, A. Rodtook, Migration planning using modified cuckoo search algorithm, in: Proceedings of 2013 13th International Symposium on Communications and Information Technologies (ISCIT), IEEE Press, 2013, pp. 621–626.
[50] X. Li, M. Yin, Modified cuckoo search algorithm with self adaptive parameter method, Inform. Sci. 298 (2015) 80–97.
[51] S.E.K. Fateen, A. Bonilla-Petriciolet, A note on effective phase stability calculations using a Gradient-Based Cuckoo Search algorithm, Fluid Phase Equilib. 375 (2014) 360–366.
[52] X. Liu, M. Fu, Cuckoo search algorithm based on frog leaping local search and chaos theory, Appl. Math. Comput. 266 (2015) 1083–1092.
[53] G.G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. 20 (9) (2016) 3349–3362.
[54] H. Wang, W. Wang, H. Sun, C. Li, S. Rahnamayan, Y. Liu, A modified cuckoo search algorithm for flow shop scheduling problem with blocking, in: Proceedings of 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2015, pp. 456–463.
[55] J.S. Wang, S.X. Li, J.D. Song, Cuckoo search algorithm based on repeat-cycle asymptotic self-learning and self-evolving disturbance for function optimization, Comput. Intell. Neurosci. 2015 (2015) 12, http://dx.doi.org/10.1155/2015/374873, Article ID 374873.
[56] M. Guerrero, O. Castillo, M. Garcia, Fuzzy dynamic parameters adaptation in the Cuckoo Search Algorithm using Fuzzy logic, in: Proceedings of 2015 IEEE Congress on Evolutionary Computation (CEC), May, 2015, pp. 441–448.
[57] S.I. Boushaki, N. Kamel, O. Bendjeghaba, Improved cuckoo search algorithm for document clustering, in: IFIP International Conference on Computer Science and its Applications_x000D_, Springer International Publishing, May, 2015, pp. 217–228.
[58] N.J. Cheung, X.M. Ding, H.B. Shen. A nonhomogeneous cuckoo search algorithm based on quantum mechanism for real parameter optimization. IEEE Transactions on Cybernetics. "This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination", 2016.
[59] E. Daniel, J. Anitha, Optimum wavelet based masking for the contrast enhancement of medical images using enhanced cuckoo search algorithm, Comput. Biol. Med. 71 (2016) 149–155.
[60] U. Mlakar, I. Fister, Hybrid self-adaptive cuckoo search for global optimization, Swarm Evol. Comput. 29 (2016) 47–72.
[61] G.G. Wang, A.H. Gandomi, X. Zhao, H.C.E. Chu, Hybridizing harmony search algorithm with cuckoo search for global numerical optimization, Soft Comput. 20 (1) (2016) 273–285.
[62] J. Wang, B. Zhou, S. Zhou, An improved cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation, Comput. Intell. Neurosci. 2016 (2016) 8.
[63] H. Wang, W. Wang, H. Sun, Z. Cui, S. Rahnamayan, S. Zeng, A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems, Soft Comput. (2016) 1–11.
[64] M. Subotic, M. Tuba, N. Bacanin, D. Simian, Parallelized cuckoo search algorithm for unconstrained optimization, in: Proceedings of the 5th WSEAS Congress on Alied Computing Conference, and Proceedings of the 1st International Conference on Biologically Inspired Computation, WSEAS Press, 2012, pp. 151–156.
[65] X.S. Yang, S. Deb, Multiobjective cuckoo search for design optimization, Comput. Oper. Res. 40 (6) (2013) 1616–1624.
[66] H. Zheng, Y. Zhou, A novel cuckoo search optimization algorithm based on Gauss distribution, J. Comput. Inform. Syst. 8 (10) (2012) 4193–4200.
[67] Y. Zhou, H. Zheng, A novel complex valued cuckoo search algorithm, Sci. World J. 2013 (2013) 6, http://dx.doi.org/10.1155/2013//597803, Article ID597803.

[68] A. Layeb, A novel quantum inspired cuckoo search for knapsack problems, Int. J. Bio-Insp. Comput. 3 (5) (2011) 297–305.
[69] N.H.T. An, V.N. Dieu, T.T. Nguyen, V.T. Kien, One rank cuckoo search algorithm for optimal reactive power dispatch, Gmsarn Int. J. 9 (2015) 73–82.
[70] M. Balasubbreddy, S. Sivanagaraju, C.V. Suresh, Multi-objective optimization in the presence of practical constraints using non-dominated sorting hybrid cuckoo search algorithm, Int. J. Eng. Sci. Technol. 18 (4) (2015) 603–615.
[71] J.Y. Chang, S.H. Liao, S.L. Wu, C.T. Lin, A hybrid of cuckoo search and simplex method for fuzzy neural network training, in: Proceedings of 2015 IEEE 12th International Conference on Networking, Sensing and Control (ICNSC), IEEE Press, 2015, pp. 13–16.
[72] D.S.L. Coelho, C.E. Klein, S.L. Sabat, V.C. Mariani, Optimal chiller loading for energy conservation using a new differential cuckoo search approach, Energy 75 (2014) 237–243.
[73] H. Chiroma, S. Abdul-kareem, A. Khan, N.M. Nawi, A.Y. Gital, L. Shuib, et al., Global warming: predicting OPEC carbon dioxide emissions from petroleum consumption using neural network and hybrid cuckoo search algorithm, PLoS One 8 (2015) e0136140, http://dx.doi.org/10.1371/journal.pone.01361.
[74] S. Das, P. Dasgupta, B.K. Panigrahi, Inter-species cuckoo search via different Lévy flights, in: Swarm, Evolutionary, and Memetic Computing, Springer International Publishing, 2013, pp. 515–526.
[75] S.E.K. Fateen, A. Bonilla-Petriciolet, Unconstrained gibbs free energy minimization for phase equilibrium calculations in nonreactive systems, using an improved cuckoo search algorithm, Ind. Eng. Chem. Res. 53 (26) (2014) 10826–10834.
[76] X.S. He, N. Li, X.S. Yang, Non-dominated sorting cuckoo search for multiobjective optimization, in: Proceedings of 2014 IEEE Symposium on Swarm Intelligence (SIS), IEEE Press, December, 2014, pp. 1–7.
[77] R.M. Jia, D.X. He, Complex valued cuckoo search with local search, in: Proceedings of 2013 Ninth International Conference on Natural Computation (ICNC), IEEE Press, 2013, pp. 1804–1808.
[78] G. Kanagaraj, S.G. Ponnambalam, N. Jawahar, Reliability-based total cost of ownership approach for supplier selection using cuckoo-inspired hybrid algorithm, Int. J. Adv. Manuf. Technol. (2014) 1–16.
[79] W.C.E. Lim, G. Kanagaraj, S.G. Ponnambalam, A hybrid cuckoo search-genetic algorithm for hole-making sequence optimization, J. Intell. Manuf. 27 (2) (2016) 417–429.
[80] S.S. Akumalla, S. Peddakotla, S.R.A. Kuppa, A modified cuckoo search algorithm for improving voltage profile and to diminish power losses by locating multi-type FACTS devices, J. Control Autom. Electr. Syst. 27 (1) (2016) 93–104.
[81] X. Li, M. Yin, A particle swarm inspired cuckoo search algorithm for real parameter optimization, Soft Comput. 20 (4) (2016) 1389–1413.
[82] S. Roy, A. Mallick, S.S. Chowdhury, S. Roy, A novel approach on Cuckoo search algorithm using Gamma distribution, in: Proceedings of 2nd International Conference on Electronics and Communication Systems (ICECS 2015), IEEE Press, 2015, pp. 466–468.
[83] J.S. Wang, S. Han, N.N. Shen, S.X. Li, Features extraction of flotation froth images and BP neural network soft-Sensor model of concentrate grade optimized by shuffled cuckoo searching algorithm, Sci. World J. 2014 (2014).
[84] L. Xiang-Tao, Y. Ming-Hao, Parameter estimation for chaotic systems using the cuckoo search algorithm with an orthogonal learning method, Chin. Phys. B 21 (5) (2012), 050507-6.
[85] S.Y. Park, Y.J. Kim, J.J. Kim, J.J. Lee, Speeded-up cuckoo search using opposition-based learning, in: Proceedings of 14th International Conference on Control, Automation and Systems (ICCAS 2014), IEEE Press, 2014, pp. 535–539.
[86] K. Zeltni, S. Meshoul, Multi-objective cuckoo search with leader selection strategies, in: Combinatorial Optimization, Springer International Publishing, 2014, pp. 421–432.
[87] J. Zhao, X. Lei, Z. Wu, Y. Tan, Clustering using improved cuckoo search algorithm, in: Advances in Swarm Intelligence, Springer International Publishing, 2014, pp. 479–488.
[88] J. Wen, J. Tong, B. Zeng, J. Zhang, A hybrid approach based on cuckoo search for bi-level distributed wind generation allocation in active distribution network, in: Proceedings of Power & Energy Society General Meeting, 2015, IEEE Press, 2015, pp. 1–5.
[89] Y. Huihua, M. Wei, Z. Xiaofeng, L. Hu, T. Songbai, A method for crude oil selection and blending optimization based on improved cuckoo search algorithm, China Petrol. Process. Petrochem. Technol. 16 (4) (2014) 70–78.
[90] J. Huang, L. Gao, X. Li, A teaching–learning-based cuckoo search for constrained engineering design problems, in: Advances in Global Optimization, Springer International Publishing, 2015, pp. 375–386.
[91] L. Maurya, P.K. Mahapatra, G. Saini, Modified cuckoo search-Based image enhancement, in: Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015, Springer, India, 2016, pp. 625–634.
[92] T.T. Nguyen, A.V. Truong, T.A. Phung, A novel method based on adaptive cuckoo search for optimal network reconfiguration and distributed generation allocation in distribution network, Int. J. Electr. Power Energy Syst. 78 (2016) 801–815.
[93] A. Abraham, E. Corchado, J.M. Corchado, Hybrid learning machines, Neurocomputing 72 (13) (2009) 2729–2730.
[94] A. Gherboudj, A. Layeb, S. Chikhi, Solving 0–1 knapsack problems by a discrete binary version of cuckoo search algorithm, Int. J. Bio-Insp. Comput. 4 (4) (2012) 229–236.
[95] Y. Zhou, H. Zheng, Q. Luo, J. Wu, An improved cuckoo search algorithm for solving planar graph coloring problem, Appl. Math. Inform. Sci. 7 (2) (2013) 785–792.
[96] Y. Feng, G.G. Wang, Q. Feng, X.J. Zhao, An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0knapsack problems, Comput. Intell. Neurosci. 2014 (2014) 1–17, http://dx.doi.org/10.1155/2014/857254, Article ID 857254.
[97] Y. Feng, K. Jia, Y. He, An improved hybrid encoding cuckoo search algorithm for 0knapsack problems, Comput. Intell. Neurosci. 2014 (2014) 1–9, http://dx.doi.org/10.1155/2014/970456, Article ID 970456.
[98] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, F. Paredes, A prefiltered cuckoo search algorithm with geometric operators for solving sudoku problems, Sci. World J. 2014 (2014) 12, http://dx.doi.org/10.1155/2014/465359, Article ID 465359.
[99] X. Zhou, Y. Liu, B. Li, A multi-objective discrete cuckoo search algorithm with local search for community detection in complex networks, Mod. Phys. Lett. B 30 (07) (2016) 1650080.
[100] D. Rodrigues, L. Pereira, T.N.S. Almeida, J.P. Papa, A.N. Souza, C.C. Ramos, X.S. Yang, BCS: a binary cuckoo search algorithm for feature selection, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2013) (2013) 465–468.
[101] P. Dasgupta, S. Das, A discrete inter-species cuckoo search for flowshop scheduling problems, Comput. Oper. Res. 60 (2015) 111–120.
[102] S. Hanoun, D. Creighton, S. Nahavandi, A hybrid cuckoo search and variable neighborhood descent for single and multiobjective scheduling problems, Int. J. Adv. Manuf. Technol. 75 (9–12) (2014) 1501–1516.
[103] S. Mahmoudi, S. Lotfi, Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem, Appl. Soft Comput. 33 (2015) 48–64.
[104] D. Laha, D.K. Behera, An improved cuckoo search algorithm for parallel machine scheduling, in: Swarm, Evolutionary, and Memetic Computing, Springer International Publishing, 2014, pp. 788–800.
[105] R. Soto, B. Crawford, R. Olivares, J. Barraza, F. Johnson, F. Paredes, A binary cuckoo search algorithm for solving the set covering problem, in: Bioinspired Computation in Artificial Systems, Springer International Publishing, 2015, pp. 88–97.
[106] A. Majumder, D. Laha, A new cuckoo search algorithm for 2-machine robotic cell scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 28 (2016) 131–143.
[107] J. Piechocki, D. Ambroziak, A. Palkowski, G. Redlarski, Use of Modified Cuckoo Search algorithm in the design process of integrated power systems for modern and energy self-sufficient farms, Appl. Energy 114 (2014) 901–908.
[108] H. Salimi, D. Giveki, M.A. Soltanshahi, J. Hatami, Extended mixture of MLP experts by hybrid of conjugate gradient method and modified cuckoo search, Int. J. Artif. Intell. Appl. 3 (1) (2012) 1, http://dx.doi.org/10.5121/ijaia.2012.3101.
[109] D. Giveki, H. Salimi, G. Bahmanyar, Y. Khademian, Automatic Detection of Diabetes Diagnosis Using Feature Weighted Support Vector Machines Based on Mutual Information and Modified Cuckoo Search. ArXiv Preprint ArXiv:1201.2173, 2012.
[110] S.Z.B.A. Abubakar, R.B. Ghazali, L.H.B. Ismail, Implementation of modified cuckoo search algorithm on functional link neural network for temperature and relative humidity prediction, Springer, Singapore, in: Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013), vol. 285, 2014, pp. 151–158.
[111] P. Nasa-ngium, K. Sunat, S. Chiewchanwattana, Enhancing modified cuckoo search by using Mantegna Lévy flights and chaotic sequences, in: Proceedings of 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2013, IEEE Press, 2013, pp. 53–57.
[112] F. Güneş, T. Karataev, S. Demirel, Composite right/leftühanded transmission lines in use for ultraüwideband matching of frontüend amplifiers with modified cuckoo search optimization, Int. J. Numer. Modell.: Electron. Netw. Dev. Fields 30 (2017) 3–4.
[113] T.T. Nguyen, D.N. Vo, Solving short-term cascaded hydrothermal scheduling problem using modified cuckoo search algorithm, Int. J. Grid Distrib. Comput. 9 (1) (2016) 67–78.
[114] S. Walton, O. Hassan, K. Morgan, Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search, Int. J. Numer. Methods Eng. 93 (5) (2013) 527–550.
[115] D. Karaboga, B. Akay, C. Ozturk, Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks, in: International Conference on Modeling Decisions for Artificial Intelligence, Springer Berlin, Heidelberg, 2007, pp. 318–329.
[116] R.A. Vazquez, Training spiking neural models using cuckoo search algorithm, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE, 2011, pp. 679–686.
[117] T.T. Nguyen, A.V. Truong, Distribution network reconfiguration for power loss minimization and voltage profile improvement using cuckoo search algorithm, Int. J. Electr. Power Energy Syst. 68 (2015) 233–242.
[118] M. Dhivya, M. Sundarambal, L.N. Anand, Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (CBPA), Int. J. Commun. Netw. Syst. Sci. 4 (04) (2011) 249–255.

[119] M. Dhivya, M. Sundarambal, Cuckoo search for data gathering in wireless sensor networks, Int. J. Mobile Commun. 9 (6) (2011) 642–656.
[120] B.M. Ismail, B.E. Reddy, T.B. Reddy, Cuckoo inspired fast search algorithm for fractal image encoding, J. King Saud Univ.-Comput. Inform. Sci. (2016), in press http://www.sciencedirect.com/science/article/pii/S1319157816301100.
[121] S. Araghi, A. Khosravi, D. Creighton, Intelligent cuckoo search optimized traffic signal controllers for multi-intersection network, Expert Syst. Appl. 42 (9) (2015) 4422–4431.
[122] P. Mehta, P. Bhatt, V. Pandya, Optimized coordinated control of frequency and voltage for distributed generating system using Cuckoo Search Algorithm, Ain Shams Eng. J. (2017), in press http://www.sciencedirect.com/science/article/pii/S2090447917300151.
[123] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (1) (2013) 17–35.
[124] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, (SIS 2007), in: Proceedings of Swarm Intelligence Symposium, 2007, 2007, pp. 120–127.
[125] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China & Technical Report, Nanyang Technological University, Singapore, 2013.
[126] S. Finck, N. Hansen, R. Ros, A. Auger, Comparing Continuous Optimizers (COCO) Documentation, Release 15.02, Available at http://coco.gforge.inria.fr.
[127] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data knowledge and data engineering, IEEE Trans. 26 (1) (2014) 97–107.
[128] J. Abawajy, Comprehensive analysis of big data variety landscape, Int. J. Parall. Emerg. Distrib. Syst. 30 (1) (2015) 5–14.
[129] X.S. Yang, X. He, Nature-Inspired optimization algorithms in engineering: overview and applications, in: Nature-Inspired Computation in Engineering, Springer International Publishing, 2016, pp. 1–20.
[130] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (January) (2006) 1–30.
[131] I. Fister Jr., U. Mlakar, X.S. Yang, I. Fister, Parameterless bat algorithm and its performance study, in: Nature-Inspired Computation in Engineering, Springer International Publishing, 2016, pp. 267–276.
[132] X. Ding, Z. Xu, N.J. Cheung, X. Liu, Parameter estimation of Takagi–Sugeno fuzzy system using heterogeneous cuckoo search algorithm, Neurocomputing 151 (2015) 1332–1342.
[133] X. Zhou, Y. Liu, B. Li, A multi-objective discrete cuckoo search algorithm with local search for community detection in complex networks, Mod. Phys. Lett. B 30 (07) (2016) 1650080.
[134] P.K. Mohanty, D.R. Parhi, A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach, Memetic Comput. 7 (4) (2015) 255–273.
[135] P. Manikandan, S. Selvarajan, Data clustering using cuckoo search algorithm (CSA), in: Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28–30, Springer India, 2012, pp. 1275–1283 (2014, January).
[136] A.K. Bhandari, V.K. Singh, A. Kumar, G.K. Singh, Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy, Expert Syst. Appl. 41 (7) (2014) 3538–3560.
[137] I. Brajevic, M. Tuba, Cuckoo search and firefly algorithm applied to multilevel image thresholding, in: X.S.I. Yang (Ed.), Cuckoo Search and Firefly Algorithm, Springer International Publishing, 2014, pp. 115–139.