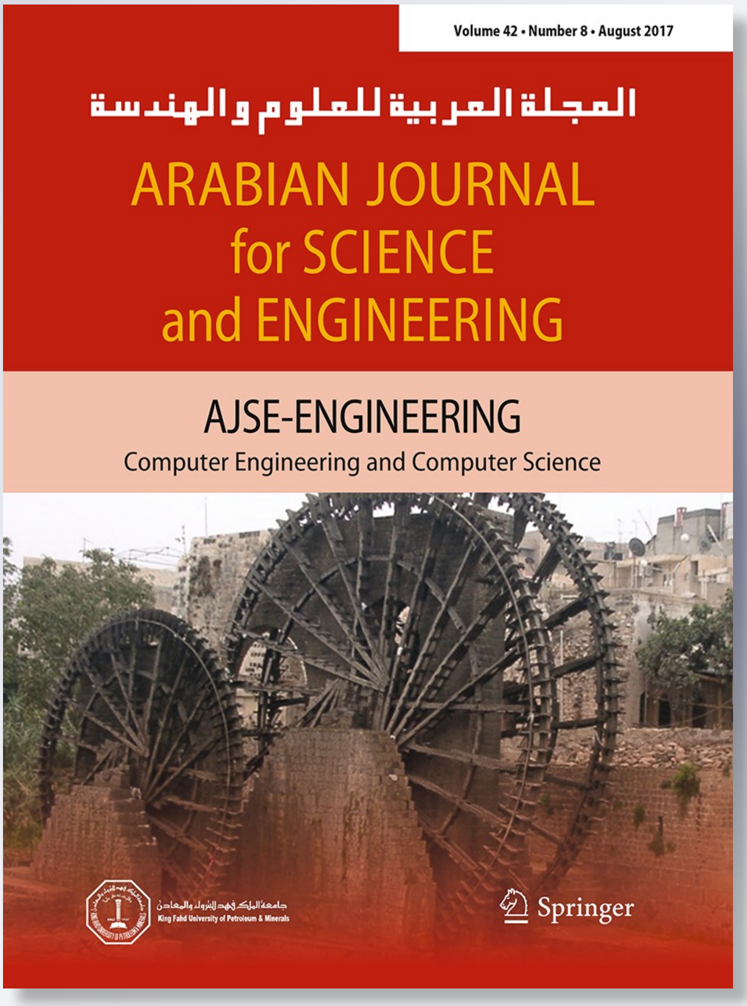
Novel Selection Schemes for Cuckoo Search

Bilal H. Abed-alguni & Faisal Alkhateeb

Arabian Journal for Science and Engineering

ISSN 2193-567X
Volume 42
Number 8

Arab J Sci Eng (2017) 42:3635-3654
DOI 10.1007/s13369-017-2663-3



Springer

Arab J Sci Eng (2017) 42:3635–3654
DOI 10.1007/s13369-017-2663-3

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Novel Selection Schemes for Cuckoo Search

Bilal H. Abed-alguni[1] · **Faisal Alkhateeb**[1]

Received: 24 January 2017 / Accepted: 8 June 2017 / Published online: 23 June 2017

Abstract The selection process is an important part of any optimization algorithm. Usually, an efficient selection process should balance between exploration of the search space and exploitation of the current knowledge about the best solutions. Cuckoo search (CS) is a simple yet powerful optimization algorithm inspired by the parasitic reproduction behavior of some cuckoo species. At each iteration of the original CS algorithm, the selection process is triggered in three places: (i) cuckoo selection where a cuckoo is selected from the population of n nests (stored solutions) based on a uniformly random function, (ii) host selection where a nest is chosen randomly from the n nests and (iii) greedy selection of a portion of the n nests for replacements with new randomly generated solutions. This paper proposes several variations of the CS algorithm by replacing the uniformly random-based selection method (used in step i) with existing randomized selection schemes, namely greedy, proportional, exponential, ε-greedy, softmax and reinforcement learning selection schemes. The proposed variations were evaluated and compared using twenty well-known benchmark functions (12 test functions from CEC 2005). The experimental results show that the proposed variations outperform the original CS algorithm.

Keywords Cuckoo search · Selection scheme · Optimization · Metaheuristic

✉ Bilal H. Abed-alguni
Bilal.h@yu.edu.jo

Faisal Alkhateeb
alkhateebf@yu.edu.jo

1 Computer Sciences Department, Yarmouk University, Irbid, Jordan

1 Introduction

The cuckoo search (CS) algorithm, an optimization algorithm that was proposed by Yang and Deb [1], was inspired by the parasitic breeding behavior of some cuckoo species and the Lévy flight behavior of some bird species. The CS algorithm has many advantages that make it a favorable choice over the other search optimization algorithms. These advantages include but are not limited to the following [1–4]: CS is simpler to implement than most of the known search optimization algorithms, it depends on one tuning parameter that too does not need to be finely tuned for each tested problem, it uses a heavy-tailed Lévy flight which is an efficient method to explore the search space, and it performs better than many known search optimization algorithms for this purpose such as particle swarm optimization (PSO), harmony search, genetic algorithm and Bat algorithm [1].

The CS algorithm starts with an initial population of n solutions stored in n nests. At each iteration of the CS algorithm, the selection process takes place in three different steps, namely (i) cuckoo selection (i.e., selecting a candidate solution for enhancement) where a cuckoo is selected randomly from the population of n nests (stored solutions), (ii) host selection (i.e., selecting a candidate solution for replacement) where a nest is chosen randomly from the n nests and (iii) greedy selection of a portion of the n nests (i.e., selecting worst candidate solutions) for replacements with new randomly generated solutions.

This paper addresses the uniformly random selection scheme of cuckoos. At this point, this selection scheme is unguided because it does not consider the solutions with the best fitness for evolution using Lévy flights. In other words, the random selection scheme only explores the solution space without taking advantage of the best solutions that can be refined using Lévy flights for possible enhancement. In the current paper, several existing selection schemes are investigated and applied to the cuckoo selection step, namely greedy, proportional, exponential, ε-greedy, softmax and reinforcement learning selection schemes. In contrast to the uniformly random selection scheme, some of the above-mentioned selection schemes are capable of balancing between exploration of new solutions and exploitation of current solutions. Some of these selection schemes have been applied with promising results to several optimization algorithms such as Harmony Search algorithm [5], Genetic algorithm [6] and Particle Swarm Optimization algorithm [7].

Valian et al. [8–10] proposed an improved version of the cuckoo search algorithm. Aiming to enhance the accuracy as well as the convergence rate, the proposed algorithm by mainly adjusts some parameters of the original algorithm. The improved algorithm has been tested based on well-known engineering optimization problems. The experimental results show that the proposed algorithm outperforms the original cuckoo search algorithm and some other well-known methods. A modified algorithm of the cuckoo search algorithm for unconstrained optimization problems has been proposed in [11]. In the proposed algorithm, the step size is determined from the sorted fitness matrix. The experimental results, based on eight benchmark functions, show that the modified algorithm provides minor enhancement to the original cuckoo search algorithm. Walton et al. [12] proposed an improved algorithm of the cuckoo search, in which two modifications have been made. The first one deals with modifying the size of the Lévy flight step size, while the second one involves the addition of information exchange between the top eggs (i.e., the best solutions). Based on standard test functions, the experimental results show that the modified cuckoo search outperforms the original CS.

The remainder of this paper is organized as follows: the CS algorithm is discussed in Sect. 2. The novel selection schemes for cuckoos are discussed in Sect. 3. Results of the experiments are presented in Sect. 4. Finally, Sect. 5 presents the conclusion and future work of this paper.

2 Cuckoo Search Algorithm

The cuckoo search (CS) algorithm is a metaheuristic algorithm that was inspired from the parasitic breeding behavior of some cuckoo species by Yang and Deb [1]. In nature, some cuckoo species follow a parasitic reproduction strategy by laying their eggs on the nests of other birds (host birds). However, there is a possibility that the host bird may discover that its eggs have been replaced. In this case, the host bird may either throw the suspicious eggs or abandon its nest. The CS algorithm can find solutions for different optimization problems by simulating two behaviors of birds: the exploitive breeding behavior of some cuckoo species and the Lévy flight behavior of some bird species [1,3]. It should be noticed that the CS algorithm can be used for solving both maximization and minimization problems. However, in the current paper, the CS algorithm and its proposed variations are used for solving minimization functions.

In the CS algorithm, each egg in a nest corresponds to a potential solution and each cuckoo egg represents a new solution. The main goal of CS is to replace current solutions (eggs in the nests) with possibly better solutions (cuckoos' eggs). Figure 1 shows the flow of the CS algorithm. The algorithm relies on two parameters, the number of solutions n (i.e., the population size) and the fraction $p_a \in [0, 1]$ of the solutions that are going to be replaced at each iteration

Begin
Objective function $f(\boldsymbol{X}_i)$, where $\boldsymbol{X}_i = \langle x_1, ..., x_m \rangle$ is a nominated solution (see Section 2.1)
Generate initial population of n solutions $\boldsymbol{X}_i (i = 1, 2, ..., n)$
while ($t <$ *Max number of iterations) or (stop criterion)* **do**
    Select a solution i randomly from the current population and replace its solution $\boldsymbol{X}_i$ by Lévy flights
    Calculate the quality/fitness value $f(\boldsymbol{X}_i)$ of $\boldsymbol{X}_i$
    Select a solution randomly from the current population (say, j)
    if $f(\boldsymbol{X}_i)$ is better than $f(\boldsymbol{X}_j)$ **then**
        Replace $\boldsymbol{X}_j$ by $\boldsymbol{X}_i$
    end if
    A fraction (p_a) of worst solutions are replaced with new ones;
    Keep the best solutions (i.e., solutions with quality solutions)
    Rank the solutions and find the current best;
end while
Post-process results and visualization
End

Fig. 1 Cuckoo search algorithm (CS) [1] of the algorithm. Basically, the CS algorithm is based on the following assumptions:

– A solution consists of m variables.
– The population size n is fixed.
– A fraction $p_a \in [0, 1]$ of the n solutions with bad fitness values are replaced with new solutions at the end of each iteration of the CS algorithm.
– At the end of each iteration of the algorithm, the best solutions will be carried over to the next generation.

2.1 Initialization of the Algorithm

First, we define all the notations used in this paper.

Notations:

– n is the number of stored solutions (nests) in a given population (In experimental settings, we used $n = 15$ as recommended by the authors of the original CS algorithm [1]).
– $\boldsymbol{X} = \langle x_1, \ldots, x_m \rangle$ is a vector of m decision variables representing a solution, where each decision variable $x_i \in [\text{minValue}_i, \text{maxValue}_i]$.
– $\boldsymbol{X}_i$ is the ith solution.
– $f(\boldsymbol{X}_i)$ is the fitness value of solution $\boldsymbol{X}_i$.
– p_i denotes the probability of selecting the ith solution $\boldsymbol{X}_i$.

In the beginning, the population of n solutions $\boldsymbol{X}_i\,(i = 1, 2, \ldots, n)$ is initialized randomly from the range of potential solutions for the objective function $f(\boldsymbol{X}_i)$. Further, the maximum number of iterations or the stop condition of the algorithm is determined based on $f(\boldsymbol{X}_i)$.

2.2 The Update Rule

A new solution ($\boldsymbol{X}_i^{t+1}$) at iteration $t + 1$ is calculated for solution i by performing a Lévy flight as follows:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \beta \oplus \text{Lévy}(\lambda), \tag{1}$$

where $\beta > 0$ is the step size that should be selected based on the size of the tested problem and the symbol $\oplus$ is the entry-wise product. Performing a random walk using Lévy flight is more efficient in exploring the potential solution space than a regular random walk because the step length of Lévy flight gets longer with the progress of the simulation process.

Basically, the Lévy flight is a random walk with a random step length that is extracted from a heavy-tailed Lévy distribution with a power law:

$$\text{Lévy} \sim u = L^{-\lambda}, \tag{2}$$

where L is the step size and $\lambda \in (1, 3]$ is a parameter related to fractal dimension.

The above Lévy distribution function has an infinite mean and an infinite variance [1]. Some of the generated solutions using Lévy flight should be around the local optimal solutions, but a fundamental percentage of the solutions should be random solutions that are far from the local best solutions. This means that this procedure will guarantee that the CS algorithm will not loop around local optimal solutions.

3 Selection Schemes

At each iteration of the CS algorithm (Fig. 1), there are three places where the selection process of solutions has to be triggered:

– In line 5, where the algorithm randomly selects a solution (cuckoo) from the n stored solutions and then updates its value by performing Lévy flights.
– In line 7, where the CS algorithm randomly selects a solution for a possible replacement with the cuckoo solution.
– In line 11, where the CS algorithm follows a greedy selection method to replace the worse solutions with new solutions, then finds the best solution among the n solutions.

In this section, we use several existing selection schemes instead of using the uniformly random selection scheme of cuckoo (line 5), namely greedy, proportional, exponential, ε-greedy, softmax and reinforcement learning selection schemes. Basically, these selection schemes direct the search into better solutions while maintaining a high diversity of population [13]. The following subsections provide more details about these selection schemes.

3.1 Uniformly Random Selection Scheme

The uniformly random selection scheme is used in the original CS algorithm. This scheme works as follows [14]: all solutions (i.e., cuckoo and nests) have equal probability to be selected:

$$p_i = \frac{1}{n}, \tag{3}$$

where n is the number of solutions.

Table 1 shows an example of random selection scheme for four solutions ranked from best to worst. Each solution has an equal opportunity to be selected as a cuckoo that can be enhanced using Lévy flight ($p_i = 1/4 = 0.25$). Note that the values for $f(\boldsymbol{X}_i)$ are given as an example for four possible solutions using the sphere function.

Table 1 Example of random selection scheme

Rank(i)	$f(X_i)$	p_i
1	2.2	0.25
2	4.5	0.25
3	10.9	0.25
4	15.6	0.25
Total		1

3.2 Greedy Selection Scheme

The greedy (or global best) selection scheme has been initially used in Particle Swarm Optimization (PSO) [15,16] and later on in Harmony search algorithm [5]. In this selection scheme, the selection probability of the best nest is unity ($p_1 = 1$), while it is zero for the remaining nests (see Table 2).

3.3 Proportional Selection Scheme

In the proportional selection scheme (sometimes called roulette wheel) [17], the selection probability of any nest depends on its evaluation (fitness) value and it is proportional to the sum of fitness values of all nests, which is defined as follows:

$$p_i = \sum_{j=1}^{n} \frac{1}{f(X_i)/f(X_j)} \tag{4}$$

Table 3 shows an example of proportional selection scheme, where the sum_prob$_i$ is the sum of the probabilities from solution 1 to solution i.

Table 2 Example of greedy selection scheme

Rank(i)	$f(X_i)$	p_i
1	2.2	1
2	4.5	0
3	10.9	0
4	15.6	0
Total		1

Table 3 Example of proportional selection scheme

Rank(i)	$f(X_i)$	$1/f(X_i)$	p_i	sum_prob$_i$
1	2.2	0.4545	0.5459	0.5459
2	4.5	0.2222	0.2669	0.8128
3	10.9	0.0917	0.1102	0.9230
4	15.6	0.0641	0.0770	1
Total		0.8326	1	

3.4 Exponential Selection Scheme

In the exponential selection scheme, the selection probability depends on a parameter s (usually initialized to 0.99 [18]), where the probability of selecting a solution is calculated as follows:

$$p_i = \frac{c_i}{\sum_{j=1}^{n} c_j}, \tag{5}$$

where $c_j = s^{j-1}$, $j = 1, \ldots, n$

It should be noticed that the value of c_1 of the best solution is $c_1 = (0.99)^0 = 1$, which is the highest value while the remaining c_i values are decreasing because the used exponential function is a decreasing function (i.e., $s < 1$) as shown in Table 4.

3.5 ε-Greedy Selection Scheme

Let ε be a real number ($\varepsilon \in (0, 1)$). Suppose the probability of selecting the global best solution is $(1-\varepsilon)$ and the probability of selecting a random nest is $\varepsilon/(n-1)$.

Table 5 shows an example of the ε-greedy selection scheme. In this example, the value of ε is 0.3 which means that the probability of selecting the global best solution is 0.7, while the probability of selecting a solution randomly is $0.3/3 = 0.1$.

Though the ε-greedy selection is a well-known approach for balancing between exploration and exploitation of solutions, it equally selects among all solutions.

Table 4 Example of exponential selection scheme

Rank(i)	$f(X_i)$	c_i	p_i
1	2.2	1	0.2557
2	4.5	0.98	0.2506
3	10.9	0.97	0.2481
4	15.6	0.96	0.2455
Total		3.91	1

Table 5 Example of ε-greedy selection scheme, where $\varepsilon = 0.3$

Rank(i)	$f(X_i)$	p_i
1	2.2	0.7
2	4.5	0.1
3	10.9	0.1
4	15.6	0.1
Total		1

3.6 Boltzman Selection Scheme

The Boltzman (softmax or Gibbs) selection scheme overcomes the drawback of the ε-greedy selection scheme by ranking the probabilities of selecting solutions according to their fitness values [19]. The selection probability of the solution in Boltzman is calculated by:

$$p_i = \frac{e^{-f(X_i)/\tau}}{\sum_{j=1}^{n} e^{-f(X_j)/\tau}}, \tag{6}$$

where τ is a positive parameter called the *temperature*

3.7 Reinforcement Learning Selection Scheme

We adapted the well-known reinforcement learning algorithm, Q-learning [20–23], as a selection scheme as suggested by Humphrys [24].

In the Q-learning algorithm, the fitness (Q-value) of each solution X_i^t at iteration t is learned using a temporal difference function called the Q-function. The Q-value of a solution at iteration t can be calculated as follows [25,26]:

$$Q(X_i^t) \longleftarrow (1-\alpha)\ Q(X_i^t) + \alpha\ [R(X_i^t) - Q(X_i^t)], \tag{7}$$

where $R(X^t)$ is the reward (i.e., numerical signal) received when the solution X^t is enhanced and $\alpha \in [0, 1]$ is the learning rate. The learning rate is used here to determine how will the newly received information affect the current Q-value.

The probability of selecting the solution with highest Q-value is 1, while the probability of selecting any solution among the remaining solutions is zero.

4 Experiments

4.1 Setup

The performances of seven variations of CS based on the proposed selection schemes were experimentally evaluated and compared using 20 well-known test functions. These variations can be distinguished in this section as follows:

1. *Random cuckoo search (RCS)* it represents the original CS algorithm that uses the random selection scheme as proposed by Yang and Deb [1].
2. *Global best cuckoo search (GCS)* a variation of CS that uses the global best selection scheme.
3. *Proportional cuckoo search (PCS)* a variation of CS that uses the proportional selection scheme.
4. *Boltzman cuckoo search (BCS)* a variation of CS that uses the Boltzman selection scheme.
5. *Exponential cuckoo search (ECS)* a variation of CS that uses the exponential selection scheme.
6. *ε-greedy cuckoo search (EGCS)* a variation of CS that uses the ε-greedy selection scheme.
7. Reinforcement learning cuckoo search (RLCS) a variation of CS that uses the RL selection scheme.

The parameters of the algorithms were set as follows:

1. All the variations of CS used the same parameter settings: the population size $n = 15$, the step size of the Lévy flight $\beta = 1$ and the portion of abandon $p_a = 0.25$. These values are recommended by the authors of the original cuckoo search algorithm [1].
2. As suggested in [27,28], the temperature $T = 0.1$ for BCS.
3. As suggested in [5,18], the ranking parameter $s = 0.99$ for ECS.
4. As suggested in [27,28], the exploration parameter $\varepsilon = 0.4$ for EGCS.
5. As recommended in [29,30], the learning rate $\alpha = 0.4$ and the discount factor $\gamma = 0.9$ for RLCS.
6. For RLCS, the reward that a solution X^t at iteration t receives was defined as:

$$R(X^t) = \begin{cases} +10.0 & \text{if the solution enhances at iteration } t \\ 0 & \text{otherwise} \end{cases}$$

7. Each algorithm was executed 100 times as recommended in several research studies [1,4,31] for similar optimization problems in order to provide meaningful statistical analysis of the proposed variations of CS.

The experiments were conducted using an Intel Xeon 2.6 GHz CPU with 8 GB RAM running 64-bit Windows. All the proposed variations of CS were programmed using Java programming language. Table 6 shows a summary of unimodal and multimodal benchmark functions that were previously used to evaluate the performance of CS algorithm as well as other famous optimization algorithms [1,4]. These benchmark functions were implemented with the dimensions 10, 30, 100 and 1000 except for Easom's, Beale's and Booth's test functions which are 2 dimensional functions. In all comparative tables, the best result for each benchmark function is highlighted with bold.

4.2 Results and Discussion

Tables 8, 9, 10 and 11 (10D, 30D, 100D and 1000D, receptively) show the experimental results for each of the 20 benchmark functions described in Table 6. The results are in the format: average of best solutions over 100 independent runs (first row), standard deviation of the best (second

Table 6 Benchmark functions used to evaluate the selection schemes for cuckoo search algorithm

Function name	Expression	Search range	Optimum value	Category
Sphere function [34]	$f_1(X) = \sum_{i=1}^{D} x_i^2$	$x_i \in [-100, 100]$	$\min(f_1) = f(0, \ldots, 0) = 0$	Unimodal
Easom's test function	$f_2(x, y) = -\cos(x)\cos(y)\exp[-(x-\pi)^2 - (y-\pi)^2]$	$(x, y) \in [-100, 100] \times [-100, 100]$	$\min(f_2) = f(\pi, \pi) = -1$	Unimodal
Step function [34]	$f_3(X) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$x_i \in [-100, 100]$	$\min(f_3) = f(0, \ldots, 0) = 0$	Discontinuous unimodal
Schwefel's problem 2.22	$f_4(X) = -\sum_{i=1}^{D} (x_i) + \prod_{i=1}^{D} (x_i)$	$x_i \in [-10, 10]$	$\min(f_4) = f(0, \ldots, 0) = 0$	Unimodal
Rastrigin's function [35]	$f_5(X) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$x_i \in [-5.12, 5.12]$	$\min(f_5) = f(0, \ldots, 0) = 0$	Multimodal
Rotated hyper-ellipsoid function	$f_6(X) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$x_i \in [-100, 100]$	$\min(f_6) = f(0, \ldots, 0) = 0$	Unimodal
Beale's function	$f_7(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	$x, y \in [-4.5, 4.5]$	$\min(f_7) = f(3, 0.5) = 0$	Multimodal
Shifted sphere function [35]	$f_8(X) = \sum_{i=1}^{D} z_i^2 + f\_\text{bias}_1$, where $z = X - o$	$x_i \in [-100, 100]$	$\min(f_8) = f(o_1, \ldots, o_D) = f\_\text{bias}_1 = -450$	Unimodal
Shifted Schwefel's problem 1.2 [35]	$f_9(X) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2 + f\_\text{bias}_2$, where $z = X - o$	$x_i \in [-100, 100]$	$\min(f_9) = f(o_1, \ldots, o_D) = f\_\text{bias}_6 = -450$	Unimodal
Booth's function	$f_10(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$	$x, y \in [-10, 10]$	$\min(f_10) = f(1, 3) = 0$	Unimodal
Shifted Schwefel's problem 1.2 with noise in fitness [36]	$f_{11}(X) = \left(\sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2 \right) * (1 + 0.4 \lvert N(0, 1) \rvert) + f\_\text{bias}_{12}$, where $z = X - o$	$x_i \in [-100, 100]$	$\min(f_{11}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{11} = -450$	Unimodal
Shifted Rosenbrock's function [36]	$f_{12}(X) = \sum_{i=1}^{D} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_\text{bias}_{13}$, where $z = X - o + 1$	$x_i \in [-100, 100]$	$\min(f_{12}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{12} = 390$	Multimodal
Shifted rotated high conditioned elliptic function [36]	$f_{13}(X) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f\_\text{bias}_{11}$, where $z = (X - o) * M$, and M : orthogonal matrix	$x_i \in [-100, 100]$	$\min(f_{13}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{13} = -450$	Unimodal
Shifted Rastrigin's function [36]	$f_{14}(X) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_\text{bias}_{14}$, where $z = X - o$	$x_i \in [-5, 5]$	$\min(f_{14}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{14} = -330$	Multimodal
Shifted rotated Rastrigin's function [36]	$f_{15}(X) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_\text{bias}_{15}$, where $z = (X - o) * M$, and M : linear transformation matrix, condition number=2	$x_i \in [-5, 5]$	$\min(f_{15}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{15} = -330$	Multimodal
Shifted expanded Griewank's plus Rosenbrock's function [36]	$f_{16}(X)$ (for the definition see CEC 2005 [36])	$x_i \in [-5, 5]$	$\min(f_{16}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{16} = -130$	Multimodal
Shifted rotated expanded Schaffer's function [36]	$f_{17}(X)$ (for the definition see CEC 2005 [36])	$x_i \in [-100, 100]$	$\min(f_{17}) = f(o_1, \ldots, o_D) = f\_\text{bias}_{17} = -300$	Multimodal

Table 6 continued

Function name	Expression	Search range	Optimum value	Category
Hybrid composition function [36]	$f_{18}(X)$ (for the definition see CEC 2005 [36])	$x_i \in [-5, 5]$	$\min(f_{18}) = f(o_1, \ldots, o_1) = f\_\text{bias}_{18} = 120$	Multimodal
Rotated version of hybrid composition function [36]	$f_{19}(X)$ (for the definition see CEC 2005 [36])	$x_i \in [-5, 5]$	$\min(f_{19}) = f(o_1, \ldots, o_1) = f\_\text{bias}_{19} = 120$	Multimodal
Rotated hybrid composition function with global optimum on the bounds [36]	$f_{20}(X)$ (for the definition see CEC 2005 [36])	$x_i \in [-5, 5]$	$\min(f_{20}) = f(o_1, \ldots, o_1) = f\_\text{bias}_{20} = 10$	Multimodal

Table 7 Summary of experimental results of seven variations of cuckoo search for twenty benchmark functions with different dimensions (Table 8: $D = 10$, Table 9: $D = 30$, Table 10: $D = 100$, Table 11: $D = 1000$)

Problem size	RCS	**GCS**	PCS	BCS	ECS	EGCS	RLCS
$D = 10$	0	1	2	5	1	7	1
$D = 30$	0	1	2	5	3	5	4
$D = 100$	0	1	2	5	3	5	4
$D = 1000$	0	2	3	4	4	6	1
Total	0	5	9	19	11	23	10

row) and error function value EFV (third row) for the average of the best solutions. The EFV measures the distance from the global optimal solution to the average of the best solutions [32].

Table 7 shows a summary of Tables 8, 9, 10 and 11. The overall results suggest that all the proposed variations of the CS algorithm outperform the original CS algorithm (RCS variation). It can also be noted that the EGCS variation outperforms all the other variations for 23 times out of 80 (20 functions × 4 dimension sizes).

Table 8 shows the results of the ten-dimensional ($D = 10$) test functions. The results show that all the proposed variations of the CS algorithm outperform the original CS algorithm (RCS variation). It can also be noted that the EGCS variation outperforms all the other variations for 7 test functions out of 20 followed by the BCS variation (4 out of 20). A possible reason for this is that EGCS applies the ε-greedy scheme and BCS applies the Boltzmann selection scheme that are capable of balancing between exploration of the solution space (n nests) and exploitation of the current best solutions (see Sect. 3).

Table 9 shows the results of the thirty-dimensional ($D = 30$) test functions. These results confirm that all the CS variations perform better than RCS. An interesting observation is that BCS and EGCS outperform the other variations for 5 test functions each out of 20. This is expected as both variations implement selection schemes that balance between random and greedy selection of cuckoos.

As in Tables 8 and 9, the obtained results in Table 10 ($D = 100$ test functions) suggest that all the variations of CS perform better than RCS. Apparently, BCS achieves five best results for F2, F5, F10, F16 and F18 functions and EGCS achieves five best results for F4, F6, F7, F11 and F20 functions.

No significant observation was noted when the dimensionality of the problems was increased to 1000 as shown in Table 11. The only notable observation is that EGCS performs better than the other variations for six functions.

The overall results of the experiments suggest that using any of proposed selection schemes of cuckoos enhances the performance of CS algorithm. The results also show that none of the proposed variations of CS has a uniformly superior performance for all the benchmark functions. However, EGCS outperforms the other variations for complex test functions (selected functions from CEC 2005, F11-F20 in the current paper).

Table 12 summarizes the experimental results of the variations of CS using the 20 test functions described in Table 6. The results are in the format: average number of iterations±standard deviation of iterations and the percentage of enhancement compared to RCS. In Table 12, the number of iterations for each algorithm was recorded when one of the following conditions is satisfied:

– the variations of the values of the benchmark function are less than a given tolerance (i.e., a threshold value $\omega \le 10^{-5}$); or

Table 8 Experimental results of seven variations of cuckoo search for twenty benchmark functions, $D = 10$, runs = 100, iterations = 10,000

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F1	1.17E−06	4.32E−07	2.63E−07	**1.24E−07**	2.12E−07	2.36E−07	2.10E−07
	1.80E−06	1.46E−06	4.57E−07	**1.75E−07**	3.21E−07	4.20E−07	3.05E−07
	1.17E−06	4.32E−07	2.63E−07	**1.24E−07**	2.12E−07	2.36E−07	2.10E−07
F2	−3.23E−02	−1.11E−28	−3.67E−02	**−8.30E−01**	−7.52E−01	−7.80E−01	−8.00E−18
	1.74E−01	5.87E−28	1.76E−01	**1.56E−01**	2.18E−01	2.10E−01	4.38E−17
	9.68E−01	1.00E+00	9.63E−01	**1.70E−01**	2.48E−01	2.20E−01	1.00E+00
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	1.07E−03	9.89E−04	**0.00E+00**	6.60E−04	8.35E−04	8.24E−04	8.56E−04
	1.21E−03	6.61E−04	**0.00E+00**	5.21E−04	9.98E−04	7.59E−04	5.61E−04
	1.07E−03	9.89E−04	**0.00E+00**	6.60E−04	8.35E−04	8.24E−04	8.56E−04
F5	1.49E−04	7.63E−05	6.40E−05	5.37E−05	3.60E−05	**1.95E−05**	4.43E−05
	2.65E−04	1.51E−04	1.05E−04	1.03E−04	1.03E−04	**2.97E−05**	5.00E−05
	1.49E−04	7.63E−05	6.40E−05	5.37E−05	3.60E−05	**1.95E−05**	4.43E−05
F6	1.27E−03	5.65E−04	2.49E−04	**2.33E−04**	3.52E−04	2.42E−04	2.47E−04
	3.33E−03	1.82E−03	4.23E−04	**2.77E−04**	5.98E−04	3.18E−04	3.07E−04
	1.27E−03	5.65E−04	2.49E−04	**2.33E−04**	3.52E−04	2.42E−04	2.47E−04
F7	2.99E−03	3.30E−04	3.00E−04	3.81E−04	3.31E−04	**2.55E−04**	3.22E−04
	6.38E−03	4.58E−04	5.12E−04	5.02E−04	4.99E−04	**3.83E−04**	4.04E−04
	2.99E−03	3.30E−04	3.00E−04	3.81E−04	3.31E−04	**1.16E−04**	3.22E−04
F8	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02
	6.38E−02	1.89E−06	5.15E−02	3.98E−02	6.15E−02	7.52E−02	5.78E−02
	8.51E−02	1.10E−06	4.60E−02	4.34E−02	6.17E−02	6.24E−02	5.72E−02
F9	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02
	1.91E−06	2.08E−06	5.41E−07	5.78E−07	2.89E−07	8.85E−07	4.42E−07
	1.10E−06	1.08E−06	4.35E−07	4.29E−07	2.14E−07	4.70E−07	2.91E−07
F10	1.13E−02	9.86E−03	7.54E−03	**2.91E−03**	1.03E+01	1.17E−02	1.07E−02
	9.08E−03	1.11E−02	7.41E−03	**3.57E−03**	5.87E+00	9.90E−03	9.84E−03
	1.13E−02	9.86E−03	7.54E−03	**2.91E−03**	1.03E+01	1.17E−02	1.07E−02
F11	2.07E+02	3.05E−04	4.20E−04	3.30E−04	2.89E−04	**2.83E−04**	3.22E−04
	1.17E+02	4.24E−04	5.02E−04	5.21E−04	3.83E−04	**4.04E−04**	4.04E−04
	6.57E+02	4.50E+02	4.50E+02	4.50E+02	4.50E+02	**4.50E+02**	4.50E+02
F12	4.90E+07	7.24E+05	5.32E+06	1.81E+06	**4.26E+05**	4.37E+06	8.29E+06
	2.55E+07	3.99E+05	2.53E+06	1.08E+06	**1.75E+05**	2.07E+06	4.33E+06
	4.90E+07	7.24E+05	5.32E+06	1.81E+06	**4.26E+05**	4.37E+06	8.29E+06
F13	2.32E+02	−4.50E+02	**−4.50E+02**	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02
	1.05E+02	4.79E−02	**4.94E−02**	4.25E−02	5.60E−02	6.28E−02	4.54E−02
	6.82E+02	2.00E−01	**1.93E−01**	2.06E−01	2.02E−01	1.98E−01	2.01E−01
F14	−2.64E+02	**−2.78E+02**	−2.72E+02	−2.72E+02	−2.72E+02	−2.76E+02	−2.58E+02
	2.06E+01	**1.61E+01**	1.60E+01	1.90E+01	1.74E+01	1.81E+01	2.14E+01
	6.62E+01	**5.24E+01**	5.81E+01	5.83E+01	5.78E+01	5.39E+01	7.23E+01
F15	−2.90E+02	−3.01E+02	−3.02E+02	−3.02E+02	−3.02E+02	**−3.09E+02**	−3.03E+02
	7.66E+00	3.84E+00	4.43E+00	3.52E+00	3.87E+00	**2.50E+00**	3.65E+00
	3.99E+01	2.91E+01	2.80E+01	2.81E+01	2.77E+01	**2.11E+01**	2.72E+01

Table 8 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F16	1.56E+01	2.00E+00	1.53E+00	−6.70E−01	−2.38E+03	**−2.69E+00**	−2.45E+00
	9.09E+00	2.26E+00	1.51E+00	8.21E−01	1.35E+03	**2.28E+00**	2.26E+00
	1.46E+02	1.32E+02	1.32E+02	1.29E+02	2.25E+03	**1.27E+02**	1.28E+02
F17	1.56E+00	−5.83E+01	−6.73E+01	−5.23E+01	−5.91E+01	−8.06E+00	**−7.28E+01**
	9.09E−01	7.63E+00	1.70E+01	2.77E+00	7.67E+00	4.55E+00	**2.65E+01**
	3.02E+02	2.42E+02	2.33E+02	2.48E+02	2.41E+02	2.92E+02	**2.27E+02**
F18	2.05E+02	2.14E+02	2.17E+02	**1.52E+02**	1.62E+02	2.04E+02	1.64E+02
	1.43E+02	1.45E+02	1.30E+02	**8.98E+01**	8.63E+01	1.11E+02	1.04E+02
	8.55E+01	9.42E+01	9.68E+01	**3.17E+01**	4.16E+01	8.38E+01	4.43E+01
F19	1.35E+02	4.10E−01	5.84E−01	9.91E−01	6.58E−01	**3.82E−01**	1.17E+00
	1.97E+02	5.81E−01	1.38E+00	1.79E+00	1.19E+00	**7.83E−01**	2.02E+00
	8.01E+06	7.47E+03	1.07E+04	6.95E+03	1.82E+04	**2.15E+04**	1.21E+04
F20	2.39E+01	1.10E+01	1.80E+01	1.80E+01	1.80E+01	**9.59E+00**	1.14E+01
	3.06E+00	2.68E+00	6.94E−03	5.38E−03	6.99E−03	**1.70E+00**	3.08E+00
	1.39E+01	1.02E+00	7.96E+00	7.97E+00	7.96E+00	**4.07E−01**	1.38E+00

Table 9 Experimental results of seven variations of cuckoo search for twenty benchmark functions, $D = 30$, runs = 100, iterations = 10,000

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F1	1.47E−06	**1.35E−07**	1.77E−07	4.96E−07	3.01E−07	4.49E−07	2.62E−07
	3.54E−06	**1.53E−07**	2.84E−07	1.13E−06	5.43E−07	9.03E−07	4.32E−07
	1.47E−06	**1.35E−07**	1.77E−07	4.96E−07	3.01E−07	4.49E−07	2.62E−07
F2	−4.19E−12	−2.06E−16	−9.98E−09	**−8.74E−01**	−7.82E−01	−8.25E−01	−1.26E−08
	2.30E−11	1.13E−15	5.47E−08	**9.58E−02**	1.67E−01	1.32E−01	6.92E−08
	1.00E+00	1.00E+00	1.00E+00	**1.26E−01**	2.18E−01	1.75E−01	1.00E+00
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	1.22E−03	6.55E−04	**0.00E+00**	6.20E−04	7.72E−04	**0.00E+00**	5.10E−04
	1.04E−03	5.41E−04	**0.00E+00**	4.60E−04	7.83E−04	**0.00E+00**	5.50E−04
	1.22E−03	6.55E−04	**0.00E+00**	6.20E−04	7.72E−04	**0.00E+00**	5.10E−04
F5	1.63E−04	4.58E−05	6.32E−05	**2.88E−05**	7.54E−05	4.40E−05	4.90E−05
	3.02E−04	8.64E−05	1.08E−04	**4.59E−05**	2.07E−04	9.49E−05	8.09E−05
	1.63E−04	4.58E−05	6.32E−05	**2.88E−05**	7.54E−05	4.40E−05	4.90E−05
F6	3.80E+00	8.36E−01	1.82E+00	1.66E+00	1.51E+00	**1.16E−04**	8.36E−01
	8.83E+00	1.26E+00	3.63E+00	3.24E+00	2.24E+00	**1.16E−04**	1.28E+00
	3.80E+00	8.36E−01	1.82E+00	1.66E+00	1.51E+00	**1.16E−04**	8.36E−01
F7	3.02E−02	3.04E−02	3.04E−02	3.02E−02	3.04E−02	**3.04E−02**	3.00E−02
	5.88E−04	6.77E−04	5.64E−04	4.28E−04	7.41E−04	**6.95E−04**	3.40E−04
	3.02E−02	3.04E−02	3.04E−02	3.02E−02	3.04E−02	**1.16E−04**	3.00E−02
F8	2.29E+04	2.61E+04	**−4.50E+02**	2.21E+04	2.31E+04	2.27E+04	2.29E+04
	1.62E+03	2.51E+03	**4.09E−04**	1.53E+03	1.35E+03	1.90E+03	1.75E+03
	2.33E+04	2.65E+04	**1.73E−04**	2.26E+04	2.35E+04	2.31E+04	2.34E+04
F9	−4.50E+02	−4.50E+02	−4.50E+02	−4.50E+02	**−4.50E+02**	−4.50E+02	−4.50E+02
	5.21E−07	2.11E−06	4.66E−07	6.94E−07	**5.77E−07**	8.69E−07	1.80E−06
	3.50E−07	6.55E−07	3.24E−07	4.12E−07	**2.30E−07**	4.10E−07	1.06E−06

Table 9 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F10	8.89E−03	8.18E−03	1.11E−02	**7.93E−03**	1.34E+01	1.23E−02	1.21E−02
	8.59E−03	7.46E−03	1.15E−02	**6.70E−03**	5.37E+00	1.27E−02	9.09E−03
	8.89E−03	8.18E−03	1.11E−02	**7.93E−03**	1.34E+01	1.23E−02	1.21E−02
F11	4.82E+07	7.10E+01	9.45E+01	7.81E+01	6.71E+01	**6.54E+01**	8.06E+01
	2.74E+07	9.88E+01	1.18E+02	1.21E+02	8.77E+01	**9.08E+01**	9.76E+01
	4.82E+07	5.21E+02	5.45E+02	5.28E+02	5.17E+02	**5.15E+02**	5.31E+02
F12	9.81E+09	1.45E+08	1.06E+09	3.62E+08	**8.53E+07**	8.74E+08	1.66E+09
	5.11E+09	7.96E+07	5.04E+08	2.16E+08	**3.50E+07**	4.15E+08	8.65E+08
	9.81E+09	1.45E+08	1.06E+09	3.62E+08	**8.53E+07**	8.74E+08	1.66E+09
F13	3.68E+04	−4.45E+02	−4.46E+02	−4.47E+02	−4.44E+02	−4.46E+02	**−4.47E+02**
	1.29E+04	5.26E−01	6.21E−01	3.57E−01	2.12E+00	6.19E−01	**2.99E−01**
	3.73E+04	4.56E+00	3.68E+00	3.30E+00	6.14E+00	3.68E+00	**3.38E+00**
F14	−1.61E+02	−1.55E+02	−1.69E+02	−1.68E+02	−1.69E+02	−1.73E+02	**−1.74E+02**
	2.06E+01	2.14E+01	1.60E+01	1.90E+01	1.74E+01	1.81E+01	**1.61E+01**
	1.69E+02	1.75E+02	1.61E+02	1.62E+02	1.61E+02	1.57E+02	**1.56E+02**
F15	−1.32E+02	−1.55E+02	−1.40E+02	−1.49E+02	−1.48E+02	−1.62E+02	**−1.70E+02**
	1.55E+01	6.78E+00	1.32E+01	1.27E+01	7.52E+00	1.19E+01	**9.75E+00**
	3.18E+02	2.95E+02	3.10E+02	3.01E+02	3.02E+02	2.88E+02	**2.80E+02**
F16	8.23E+01	2.00E+00	1.20E+01	**−2.37E+00**	1.76E+00	1.75E+00	1.39E+00
	4.81E+01	2.26E+00	1.18E+01	**1.35E+00**	3.05E+00	3.13E+00	3.40E+00
	2.12E+02	1.32E+02	1.42E+02	**1.28E+02**	1.32E+02	1.32E+02	1.31E+02
F17	1.61E+01	−6.73E+00	−1.84E+01	1.06E+00	−7.80E+00	−3.88E+00	**−2.57E+01**
	5.67E+00	9.92E+00	2.20E+01	3.59E+00	9.98E+00	1.36E+01	**3.45E+01**
	3.16E+02	2.93E+02	2.82E+02	3.01E+02	2.92E+02	2.96E+02	**2.74E+02**
F18	8.32E+05	3.46E+03	2.25E+03	**1.87E+03**	8.35E+03	2.82E+03	2.22E+03
	1.75E+06	4.13E+03	3.09E+03	**3.31E+03**	1.35E+04	4.03E+03	4.79E+03
	8.32E+05	3.34E+03	2.13E+03	**1.75E+03**	8.23E+03	2.70E+03	2.10E+03
F19	3.37E+03	1.04E+03	1.48E+03	2.51E+03	**9.67E+02**	2.95E+03	1.67E+03
	5.00E+03	1.47E+03	3.50E+03	4.53E+03	**1.98E+03**	5.12E+03	3.01E+03
	3.25E+03	9.18E+02	1.36E+03	2.39E+03	**8.47E+02**	2.83E+03	1.55E+03
F20	9.60E+02	2.46E+02	4.02E+02	4.02E+02	4.02E+02	**2.14E+02**	2.54E+02
	3.45E+02	5.98E+01	5.67E−01	4.08E−01	5.67E−01	**3.78E+01**	6.85E+01
	9.50E+02	2.36E+02	3.92E+02	3.92E+02	3.92E+02	**2.04E+02**	2.44E+02

Table 10 Experimental results of seven variations of cuckoo search for twenty benchmark functions, $D = 100$, runs = 100, iterations = 10,000

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F1	1.79E−06	3.21E−07	4.49E−07	4.68E−07	2.84E−07	3.26E−07	**2.25E−07**
	4.04E−06	4.27E−07	7.27E−07	8.56E−07	4.05E−07	7.14E−07	**3.60E−07**
	1.79E−06	3.21E−07	4.49E−07	4.68E−07	2.84E−07	3.26E−07	**2.25E−07**
F2	−5.72E−08	−7.85E−33	−4.67E−04	**−8.28E−01**	−8.03E−01	−7.58E−01	−9.28E−12
	3.14E−07	4.27E−32	−4.83E−04	**1.28E−01**	1.49E−01	1.74E−01	5.02E−11
	1.00E+00	1.00E+00	1.00E+00	**1.72E−01**	1.97E−01	2.42E−01	1.00E+00
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 10 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F4	1.34E−03	6.07E−04	**0.00E+00**	7.10E−04	9.10E−04	**0.00E+00**	5.78E−04
	1.36E−03	5.71E−04	**0.00E+00**	8.44E−04	1.02E−03	**0.00E+00**	6.06E−04
	1.34E−03	6.07E−04	**0.00E+00**	7.10E−04	9.10E−04	**0.00E+00**	5.78E−04
F5	2.38E−04	8.50E−05	5.36E−05	6.37E−05	1.48E−04	**2.64E−05**	3.80E−05
	3.55E−04	2.16E−04	8.32E−05	1.53E−04	4.62E−04	**6.73E−05**	5.82E−05
	2.38E−04	8.50E−05	5.36E−05	6.37E−05	1.48E−04	**2.64E−05**	3.80E−05
F6	7.58E+01	5.53E+01	5.00E+01	**4.31E+01**	8.66E+01	6.14E+01	5.62E+01
	1.26E+02	7.49E+01	7.36E+01	**7.09E+01**	1.35E+02	1.14E+02	1.16E−04
	7.58E+01	5.53E+01	5.00E+01	**4.31E+01**	8.66E+01	6.14E+01	1.16E−04
F7	3.02E−02	**3.01E−02**	3.05E−02	3.03E−02	3.04E−02	3.04E−02	3.03E−02
	3.45E−04	**2.83E−04**	8.99E−04	7.01E−04	6.05E−04	7.19E−04	1.03E−03
	3.02E−02	**3.01E−02**	3.05E−02	3.03E−02	3.04E−02	3.04E−02	1.16E−04
F8	1.32E+05	1.42E+05	**−4.50E+02**	1.32E+05	1.32E+05	1.33E+05	1.32E+05
	5.58E+03	7.46E+03	**4.11E−03**	4.41E+03	4.56E+03	4.57E+03	5.20E+03
	1.33E+05	1.42E+05	**2.72E−03**	1.33E+05	1.32E+05	1.33E+05	1.32E+05
F9	5.21E+06	5.10E+06	**5.02E+06**	5.30E+06	5.09E+06	5.23E+06	5.11E+06
	7.75E+05	4.35E+05	**3.10E+05**	2.96E+05	1.92E+05	1.85E+05	2.29E+05
	5.21E+06	5.10E+06	**5.02E+06**	5.30E+06	5.09E+06	5.23E+06	5.11E+0
F10	8.60E−03	1.12E−02	8.96E−03	**6.13E−03**	1.31E+01	7.84E−03	8.20E−03
	8.05E−03	1.12E−02	1.03E−02	**7.19E−03**	5.02E+00	7.90E−03	6.89E−03
	8.60E−03	1.12E−02	8.96E−03	**6.13E−03**	1.31E+01	7.84E−03	8.20E−03
F11	1.22E+14	8.84E+07	1.18E+08	9.72E+07	8.35E+07	**8.14E+07**	1.00E+08
	6.94E+13	1.23E+08	1.47E+08	1.50E+08	1.09E+08	**1.13E+08**	1.21E+08
	1.22E+14	8.84E+07	1.18E+08	9.72E+07	8.35E+07	**8.14E+07**	1.00E+0
F12	4.93E+12	1.45E+11	4.81E+12	1.82E+11	**4.28E+10**	4.39E+11	8.32E+11
	2.57E+12	7.96E+10	8.85E+12	1.08E+11	**1.76E+10**	2.08E+11	4.34E+11
	4.93E+12	1.45E+11	4.81E+12	1.82E+11	**4.28E+10**	4.39E+11	8.32E+11
F13	3.91E+05	2.45E+02	3.06E+02	4.83E+02	2.29E+02	4.33E+02	**1.98E+02**
	1.37E+05	4.49E+02	4.74E+02	3.93E+02	4.37E+02	4.52E+02	**4.79E+02**
	3.92E+05	6.95E+02	7.56E+02	9.33E+02	6.79E+02	8.83E+02	**6.48E+02**
F14	1.07E+00	−3.54E+01	**−5.04E+01**	1.07E+01	−4.71E+01	−4.71E+01	1.06E+02
	1.03E+02	7.81E+01	**8.06E+01**	8.54E+01	9.55E+01	7.35E+01	1.52E+02
	3.31E+02	2.95E+02	**2.80E+02**	3.41E+02	2.83E+02	2.83E+02	4.36E+02
F15	6.41E+02	5.26E+02	5.14E+02	5.21E+02	5.32E+02	**4.87E+02**	5.13E+02
	1.98E+01	2.16E+01	1.75E+01	1.75E+01	2.16E+01	**1.76E+01**	2.22E+01
	1.09E+03	9.76E+02	9.64E+02	9.71E+02	9.82E+02	**9.37E+02**	9.63E+02
F16	2.95E+02	1.47E+02	1.15E+02	4.84E+01	4.67E+01	**4.59E+01**	5.42E+01
	1.48E+02	8.55E+01	5.81E+01	2.97E+01	3.19E+01	**3.14E+01**	2.79E+01
	4.25E+02	2.77E+02	2.45E+02	1.78E+02	1.77E+02	**1.76E+02**	1.84E+02
F17	1.43E+02	1.25E+02	5.83E+01	−6.72E+01	1.40E+01	3.37E+01	**−7.55E+01**
	1.32E+01	9.88E+00	1.79E+01	3.84E+02	4.99E+01	6.83E+01	**1.73E+02**
	4.43E+02	4.25E+02	3.58E+02	2.33E+02	3.14E+02	3.34E+02	**2.24E+02**
F18	5.11E+02	**3.20E+02**	6.23E+02	4.02E+02	3.23E+02	6.80E+02	4.71E+02
	3.22E+02	**3.46E+01**	3.39E+02	2.88E+02	2.48E+02	3.07E+02	2.47E+02
	3.91E+02	**2.00E+02**	5.03E+02	2.82E+02	2.03E+02	5.60E+02	3.51E+02

Table 10 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F19	2.43E+05	3.22E+02	4.58E+02	**2.99E+02**	7.77E+02	9.16E+02	5.16E+02
	3.60E+05	4.55E+02	1.09E+03	**6.13E+02**	1.40E+03	1.59E+03	9.33E+02
	2.42E+05	2.02E+02	3.38E+02	**1.79E+02**	6.57E+02	7.96E+02	3.96E+02
F20	5.99E+03	1.53E+03	2.50E+03	2.50E+03	2.50E+03	**1.34E+03**	1.58E+03
	2.15E+03	3.73E+02	3.16E+00	2.27E+00	3.16E+00	**2.35E+02**	4.27E+02
	5.98E+03	1.52E+03	2.49E+03	2.49E+03	2.49E+03	**1.33E+03**	1.57E+03

Table 11 Experimental results of seven variations of cuckoo search for twenty benchmark functions, $D = 1000$, runs=100, iterations = 10,000

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F1	8.68E−06	1.98E−07	**1.98E−07**	3.09E−07	4.10E−07	2.87E−07	2.66E−07
	4.22E−05	3.43E−07	**2.44E−07**	6.21E−07	7.73E−07	5.46E−07	5.47E−07
	8.68E−06	1.98E−07	**1.98E−07**	3.09E−07	4.10E−07	2.87E−07	2.66E−07
F2	−2.75E−18	−2.60E−11	−7.61E−01	−7.93E−01	**−8.30E−01**	−1.27E−05	−1.33E−09
	1.51E−17	1.42E−10	2.04E−01	1.65E−01	**1.31E−01**	6.82E−05	7.15E−09
	1.00E+00	1.00E+00	2.39E−01	2.07E−01	**1.70E−01**	1.00E+00	1.00E+00
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	1.42E−03	8.81E−04	**0.00E+00**	5.10E−04	9.03E−04	**0.00E+00**	6.91E−04
	1.17E−03	9.57E−04	**0.00E+00**	6.12E−04	1.06E−03	**0.00E+00**	7.77E−04
	1.42E−03	8.81E−04	**0.00E+00**	5.10E−04	9.03E−04	**0.00E+00**	6.91E−04
F5	1.06E−03	4.34E−05	5.11E−05	8.45E−05	**3.69E−05**	6.46E−05	4.34E−05
	6.57E−05	6.92E−05	8.81E−05	2.90E−04	**5.49E−05**	1.00E−04	9.34E−05
	1.06E−03	4.34E−05	5.11E−05	8.45E−05	**3.69E−05**	6.46E−05	4.34E−05
F6	5.17E+04	**5.25E+03**	8.93E+11	3.53E+04	5.31E+03	1.14E+04	1.35E+04
	4.63E+04	**1.16E−04**	6.54E+04	6.43E+04	1.87E+04	1.43E+04	6.30E+03
	5.17E+04	**1.16E−04**	8.93E+11	3.53E+04	5.31E+03	1.14E+04	1.35E+04
F7	3.02E−02	**3.01E−02**	3.03E−02	3.01E−02	3.03E−02	3.03E−02	3.04E−02
	4.94E−04	**5.21E−04**	7.50E−04	5.48E−04	6.10E−04	6.77E−04	9.71E−04
	3.02E−02	**1.16E−04**	3.03E−02	3.01E−02	3.03E−02	3.03E−02	3.04E−02
F8	1.32E+05	1.38E+05	**−4.50E+02**	1.32E+05	1.31E+05	1.33E+05	1.32E+05
	6.00E+03	8.84E+03	**4.90E−03**	5.27E+03	5.15E+03	4.78E+03	4.84E+03
	1.32E+05	1.38E+05	**4.20E−03**	1.33E+05	1.31E+05	1.33E+05	1.32E+05
F9	5.88E+06	5.12E+06	5.02E+06	5.18E+06	5.29E+06	**4.45E+06**	5.02E+06
	1.81E+06	4.64E+05	2.75E+05	2.95E+05	1.18E+06	**3.61E+05**	2.80E+05
	5.88E+06	5.12E+06	5.02E+06	5.18E+06	5.29E+06	**4.45E+06**	5.02E+06
F10	9.74E−03	1.01E−02	1.11E−02	**1.21E+01**	1.01E−02	9.51E−03	9.91E−03
	9.95E−03	1.03E−02	1.11E−02	**5.62E+00**	9.28E−03	1.09E−02	8.44E−03
	9.74E−03	1.01E−02	1.11E−02	**1.21E+01**	1.01E−02	9.51E−03	9.91E−03
F11	1.50E+30	1.49E+28	1.50E+28	1.29E+28	1.56E+28	**8.00E+27**	1.54E+28
	2.31E+29	6.54E+26	1.25E+27	4.55E+27	8.89E+26	**6.67E+27**	5.32E+26
	1.50E+30	1.49E+28	1.50E+28	1.29E+28	1.56E+28	**8.00E+27**	1.54E+28
F12	4.54E+32	1.33E+31	4.43E+32	1.68E+31	**3.95E+30**	4.05E+31	7.67E+31
	2.37E+32	7.34E+30	8.15E+32	9.98E+30	**1.62E+30**	1.92E+31	4.01E+31
	4.54E+32	1.33E+31	4.43E+32	1.68E+31	**3.95E+30**	4.05E+31	7.67E+31

Table 11 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F13	4.39E+08	8.71E+04	1.08E+05	1.65E+05	8.19E+04	**7.09E+04**	1.49E+05
	1.54E+08	1.50E+05	1.58E+05	1.32E+05	1.46E+05	**1.60E+05**	1.50E+05
	4.54E+32	1.33E+31	4.43E+32	1.68E+31	3.95E+30	**4.05E+31**	7.67E+31
F14	1.10E+04	1.10E+03	1.51E+03	1.05E+03	1.00E+03	1.02E+03	**9.52E+02**
	3.83E+03	2.83E+02	5.45E+02	2.62E+02	3.15E+02	2.62E+02	**3.45E+02**
	1.13E+04	1.43E+03	1.84E+03	1.38E+03	1.33E+03	1.35E+03	**1.28E+03**
F15	2.95E+04	1.11E+04	1.08E+04	**1.02E+04**	1.11E+04	1.02E+04	1.01E+04
	3.05E+04	1.95E+03	1.26E+03	**4.93E+01**	1.93E+03	5.81E+01	4.80E+01
	2.98E+04	1.15E+04	1.12E+04	**1.05E+04**	1.14E+04	1.05E+04	1.05E+04
F16	1.42E+04	6.58E+03	5.59E+03	2.82E+03	**2.19E+03**	2.61E+03	3.11E+03
	1.33E+04	5.27E+03	3.96E+03	2.40E+03	**1.65E+03**	2.50E+03	2.47E+03
	1.44E+04	6.71E+03	5.72E+03	2.95E+03	**2.32E+03**	2.74E+03	3.24E+03
F17	4.73E+03	4.15E+03	1.95E+03	3.91E+02	2.28E+02	**1.89E+02**	4.07E+02
	4.33E+02	3.28E+02	5.89E+02	7.68E+02	9.98E+01	**1.37E+02**	3.46E+02
	5.03E+03	4.45E+03	2.25E+03	6.91E+02	5.28E+02	**4.89E+02**	7.07E+02
F18	6.17E+04	6.43E+04	6.51E+04	**4.55E+04**	4.85E+04	6.12E+04	4.93E+04
	4.28E+04	4.36E+04	3.90E+04	**2.70E+04**	2.59E+04	3.33E+04	3.13E+04
	6.16E+04	6.42E+04	6.50E+04	**4.54E+04**	4.84E+04	6.11E+04	4.92E+04
F19	8.01E+06	7.59E+03	1.08E+04	**7.07E+03**	1.83E+04	2.16E+04	1.22E+04
	1.19E+07	1.07E+04	2.56E+04	**1.45E+04**	3.31E+04	3.75E+04	2.20E+04
	8.01E+06	7.47E+03	1.07E+04	**6.95E+03**	1.82E+04	2.15E+04	1.21E+04
F20	1.84E+06	2.30E+04	3.72E+04	3.72E+04	3.72E+04	**1.98E+04**	2.36E+04
	6.44E+05	6.96E+03	6.56E+03	6.56E+03	6.56E+03	**4.97E+03**	7.75E+03
	1.84E+06	2.30E+04	3.72E+04	3.72E+04	3.72E+04	**1.98E+04**	2.36E+04

– the results improve by less than one over 100 consecutive iterations.

It can be observed from Table 12 that the GCS and PCS variations converge faster to solutions for five test functions each (unimodal or basic multimodal functions). This may be because GCS follows a greedy approach that is suitable for reproducing solutions for simple functions. Furthermore, PCS always gives an exploration chance for solutions that are weaker than the best solutions.

However, as shown in the table, all the variations of CS perform much better than RCS except for GCS and PCS with some functions. GCS converges slower than RCS for F8, F9, F12, F15, F16 and F20, while PCS converges slower than RCS for F15 and F16. This might be because GCS uses a fully exploitive selection scheme (greedy selection) that searches at each iteration of the CS algorithm for the best global solutions around the current best solution. In addition, PCS is not guaranteed to reproduce solutions around the best solutions [33].

On the other hand, the variations (BCS, ECS, EGCS, RLCS) perform better than GCS and PCS in solving complex problems (i.e., selected expanded and composition functions from CEC 2005, F15-F20 in the current paper). This may be because these variations use selection schemes that balance between the exploration and exploitation of nominated solutions in the solution space, which might be useful for solving complex optimization functions.

4.3 ANOVA Test Results

The one-way analysis of variance (ANOVA) is used to test if the means of the seven variations of the CS algorithm are equal and to determine if there exists one mean that is significantly different from the other means. The hypotheses are:

$\boldsymbol{H_0}$: $\mu_1 = \mu_2 = \cdots = \mu_7$

$\boldsymbol{H_1}$: at least one mean is different from the others.

Tables 13, 14, 15 and 16 provide the ANOVA results for all the test functions with different dimensions (Table 13: $D = 10$, Table 14: $D = 30$, Table 15: $D = 100$ and Table 16: $D = 1000$). As shown in the tables, the P value for all the problems except for F3 (Step Function) is less than 5% and the value of F calculated (F-cal) is larger than the value of F critical (F-crit). Therefore, H_0 will be rejected at 5% significance level and H_1 will be

Table 12 Performance results of seven variations of CS

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F1 ($D = 256$)	1936 ± 1890	1117 ± 1351	1179 ± 1041.7	**1100 ± 1067.2**	1275 ± 1273.2	1317 ± 1482	1241 ± 1176
	0%	42%	39%	**43%**	34%	32%	36%
F2 ($D = 2$)	14,853 ± 14,680.07	**10,511 ± 10,607.02**	13,813 ± 11,907.37	12,073 ± 10,975.88	12,401 ± 11,704.1	11,993 ± 12,218.85	12,210 ± 11,832.16
	0%	**29%**	7%	19%	17%	19%	18%
F3 ($D = 256$)	182 ± 189.28	168 ± 160.44	136 ± 124.47	144 ± 122.96	**125 ± 129.77**	167 ± 145.91	159 ± 153.63
	0%	8%	25%	21%	**31%**	8%	13%
F4 ($D = 256$)	9350 ± 7718.18	7284 ± 6798.72	**12 ± 13.02**	7128 ± 5931.7	24,670 ± 1898.68	755 ± 6485.46	7496 ± 6264.39
	0%	22%	**100%**	24%	−164%	19%	20%
F5 ($D = 256$)	9220 ± 9012	4524 ± 4828	**4687 ± 4151**	5640 ± 5394	5002 ± 4314	5441 ± 4742	5550 ± 6666
	0%	51%	**49%**	39%	46%	41%	40%
F6 ($D = 256$)	10,2254 ± 88,130	**36,463 ± 25,445**	74,008 ± 52,358.85	970,68 ± 56,008	82,579 ± 73,172	68,384 ± 55,682	66,953 ± 63,698
	0%	**64%**	28%	5%	19%	33%	35%
F7 ($D = 2$)	20,688 ± 19,467	**15,233 ± 146,611**	18,511 ± 21,079	21,893 ± 22,016	17,645 ± 16,106	23,580 ± 20,273	16,348 ±
	0%	**26%**	1%	≈ 0%	15%	≈ 0%	≈ 0%
F8 ($D = 10$)	11,644 ± 10,325	14,171 ± 15,063	**4712 ± 3948**	11,191 ± 11,197	11,489 ± 11,871.99	10,755.00226 ± 11,140	10,67 ± 10,344
	0%	−22%	**60%**	4%	1%	8%	8%
F9 ($D = 10$)	5565 ± 5298	6828 ± 7297	5348 ± 5395.31	5037 ± 5712	5448 ± 5633	**4967 ± 4174**	5440 ± 4485
	0%	−23%	4%	9%	2%	**11%**	2%
F10 ($D = 2$)	11,2266 ± 88,011	11,0035 ± 101,103	**22,704 ± 20,509**	25,696 ± 26,736	25,069 ± 24,027	28,101 ± 24,215	23,996 ± 28,009
	0%	2%	**80%**	77%	78%	75%	79%
F11 ($D = 10$)	27,395 ± 29,804	25,734 ± 34,054	23,614 ± 28,157	22,258 ± 34,244	24,598 ± 30,286	**21,221 ± 19,813**	23,407 ± 26,745
	0%	6%	14%	19%	10%	**23%**	15%
F12 ($D = 10$)	132,107 ± 117,148	160,782 ± 170,892	**53,458 ± 44,790**	126,967 ± 127,036	130,348 ± 134,696	122,023 ± 126,391	121,095 ± 117,365
	0%	−22%	**60%**	4%	1%	8%	8%

Table 12 continued

Function name	RCS	GCS	PCS	BCS	ECS	EGCS	RLCS
F13 ($D = 10$)	16,603 ± 9012	**12,855 ± 5320**	12,961 ± 4503	14,289 ± 5889	13,516 ± 4472	14,166 ± 5049	14,076 ± 6759
	0%	**23%**	22%	14%	19%	15%	15%
F14 ($D = 10$)	15,780 ± 8175	**13,665 ±5789**	14,472 ± 9731	15,284 ± 5830	14,496 ± 4577	15,117 ± 5127	15,139 ± 6867
	0%	**13%**	8%	3%	8%	4%	4%
F15 ($D = 10$)	44,522 ± 66,369	47,603 ± 88,746	48,767± 88,242	39,236 ± 58,324	36,317 ± 59,905	**33,632 ± 37,575**	39,864 ± 51,129
	0%	−7%	−10%	12%	18%	**24%**	10%
F16 ($D = 10$)	74,798 ± 111,500	75,895 ± 148,509	77,426 ± 148,694	**56,358 ± 83,335**	58,074 ± 90,207	59,485 ± 141,086	56,503 ± 71,714
	0%	−1%	−4%	**25%**	22%	20%	24%
F17 ($D = 10$)	55,521 ± 46,188	44,644 ± 31,488	53,845 ± 69,604	54,942 ±47,604	**43,847 ± 32,576**	55,282 ±35,651	53,475 ± 38,554
	0%	20%	3%	1%	**21%**	0%	4%
F18 ($D = 10$)	275,406 ±292,529	211,509 ± 210,101	246,503 ± 276,514	276,626 ± 317,943	**186,578 ± 177,829**	270,207 ± 268,327	269,849 ± 253,389
	0%	23%	10%	0%	**32%**	2%	2%
F19 ($D = 10$)	250,199 ± 263,276	197,319 ± 198,962	218,879 ± 206,766	243,507 ± 275,809	**176,365 ± 174,311**	233,223 ± 241,103	229,798 ± 221,628
	0%	21%	13%	3%	**30%**	7%	8%
F20 ($D = 10$)	305,511 ± 329,308	313,645 ± 361,312	269,011 ± 305,080	286,088 ± 296,231	306,258 ± 296,574	238,113 ± 234,492	**214,906 ± 182,043**
	0%	−3%	12%	6%	0%	22%	**30%**

Table 13 ANOVA table with 30 runs and $D = 10$ for benchmark functions

f(X)	Source of variation	SS	*df*	MS	*F*	*P* value	*F* crit
F1	Between groups	2.33141E−11	6	3.88568E−12	4.568379937	0.000226408	2.143453
	Within groups	1.72664E−10	203	8.5056E−13			
F2	Between groups	30.448779	6	5.0747965	207.6135058	9.27E−84	2.1434529
	Within groups	4.962026365	203	0.02444348			
F3	Between groups	0	6	0	65,535	Error	2.1434529
	Within groups	0	203	0			
F4	Between groups	2.26026E−05	6	3.7671E−06	6.510817633	2.66721E−06	2.143453
	Within groups	0.000117454	203	5.78591E−07			
F5	Between groups	3.19308E−07	6	5.3218E−08	2.893982191	0.009954008	2.143453
	Within groups	3.73301E−06	203	1.83892E−08			
F6	Between groups	2.62373E−05	6	4.37288E−06	2.009161289	0.065987607	2.143453
	Within groups	0.000441824	203	2.17647E−06			
F7	Between groups	0.000182999	6	3.04999E−05	5.088858119	6.87254E−05	2.143453
	Within groups	0.001216673	203	5.99346E−06			
F8	Between groups	0.123886618	6	0.02064777	6.853464579	1.22537E−06	2.143453
	Within groups	0.611588079	203	0.003012749			
F9	Between groups	2.38894E−11	6	3.98157E−12	2.88E+00	1.02E−02	2.14E+00
	Within groups	2.80258E−10	203	1.38058E−12			
F10	Between groups	2742.126051	6	4.57E+02	9.28E+01	1.90E−55	2.14E+00
	Within groups	999.764962	203	4.92E+00			
F11	Between groups	1,057,587.486	6	1.76E+05	9.28E+01	1.93E−55	2.14E+00
	Within groups	385,685.3645	203	1.90E+03			
F12	Between groups	5.48E+16	6	9.13E+15	9.36E+01	9.76E−56	2.14E+00
	Within groups	1.98E+16	203	9.75E+13			
F13	Between groups	11,966,520	6	1.99E+06	1.28E+03	3.68E−158	2.14E+00
	Within groups	317,535.7	203	1.56E+03			
F14	Between groups	9510.902	6	1.59E+03	4.84E+00	1.22E−04	2.14E+00
	Within groups	66,498.66	203	3.28E+02			
F15	Between groups	5813.204	6	9.69E+02	5.23E+01	1.23E−38	2.14E+00
	Within groups	3758.345	203	1.85E+01			
F16	Between groups	1.46E+08	6	2.43E+07	9.31E+01	1.45E−55	2.14E+00
	Within groups	52,889,776	203	2.61E+05			
F17	Between groups	156,858	6	2.61E+04	1.61E+02	2.46E−74	2.14E+00
	Within groups	32,965.78	203	1.62E+02			
F18	Between groups	1.89E−07	6	3.16E−08	4.59E+00	2.18E−04	2.14E+00
	Within groups	1.4E−06	203	6.88E−09			
F19	Between groups	461,305.9	6	7.69E+04	1.39E+01	3.24E−13	2.14E+00
	Within groups	1,123,660	203	5.54E+03			
F20	Between groups	461,305.9	6	7.69E+04	1.39E+01	3.24E−13	2.14E+00
	Within groups	1,123,660	203	5.54E+03			

accepted for all the test functions except for F3. Hence, not all means are equal. F3 is considered a simple function and thus all the seven variations provide equal means for 30 runs.

5 Conclusion and Future Work

This paper presented six new variations of cuckoo search (CS) where each variation is based on a new selection

Table 14 ANOVA table with 30 runs and $D = 30$ for benchmark functions

$f(X)$	Source of variation	SS	*df*	MS	*F*	*P* value	*F* crit
F1	Between groups	11,771,521	6	1,961,920	106.6621	5.36E−60	2.143453
	Within groups	3,733,939	203	18,393.79			
F2	Between groups	35.27927421	6	5.879879	756.91081	5.8E−136	2.143453
	Within groups	1.576956524	203	0.007768			
F3	Between groups	0	6	0	65535	Error	2.143452883
	Within groups	1.576956524	203	0.007768			
F4	Between groups	3.35957E−05	6	5.59929E−06	15.67062	9.25E−15	2.143453
	Within groups	7.25342E−05	203	3.57311E−07			
F5	Between groups	3.61609E−07	6	6.02682E−08	2.46653	0.025242	2.143453
	Within groups	4.96019E−06	203	2.44344E−08			
F6	Between groups	171.711673	6	28.61861216	1.682163	0.126926	2.143453
	Within groups	3453.635927	203	17.01298486			
F7	Between groups	4.33875E−06	6	7.23124E−07	2.06382	0.058973	2.143453
	Within groups	7.11274E−05	203	3.50381E−07			
F8	Between groups	14,796,951,092	6	2,466,158,515	874.3646	4.6E−142	2.143453
	Within groups	572,564,550.7	203	2,820,515.028			
F9	Between groups	1.446E−11	6	2.41E−12	1.731198	0.115325	2.143453
	Within groups	2.82596E−10	203	1.3921E−12			
F10	Between groups	4614.462557	6	769.0770928	186.3675	1.03E−79	2.143453
	Within groups	837.713917	203	4.126669542			
F11	Between groups	5.74617E+16	6	9.58E+15	92.85844	1.81E−55	2.143453
	Within groups	2.09364E+16	203	1.03E+14			
F12	Between groups	2.19052E+21	6	3.65E+20	93.62485	9.85E−56	2.143453
	Within groups	7.91591E+20	203	3.9E+18			
F13	Between groups	35,710,935,035	6	5.95E+09	249.8282	7.67E−91	2.143453
	Within groups	4,836,203,431	203	23,823,662			
F14	Between groups	9451.825016	6	1575.304	4.796471	0.000134	2.143453
	Within groups	66,671.26263	203	328.4299			
F15	Between groups	29,533.51672	6	4922.253	41.92357	4.84E−33	2.143453
	Within groups	23,834.26184	203	117.4102			
F16	Between groups	166,382.5878	6	27,730.43	77.9913	5.22E−50	2.143453
	Within groups	72,178.27823	203	355.558			
F17	Between groups	32,656.43762	6	5442.74	18.07733	9E−17	2.143453
	Within groups	61,119.43179	203	301.0809			
F18	Between groups	1.49293E−05	6	2.49E−06	3.094982	0.006378	2.143453
	Within groups	0.000163202	203	8.04E−07			
F19	Between groups	2.89E+08	6	48,175,117	13.48365	7.41E−13	2.143453
	Within groups	7.25E+08	203	3,572,854			
F20	Between groups	11,771,521	6	1,961,920	106.6621	5.36E−60	2.143453
	Within groups	3,733,939	203	18,393.79			

scheme. These variations are global best cuckoo search (GCS), proportional cuckoo search (PCS), Boltzman cuckoo search (BCS), exponential cuckoo search (ECS), ε-greedy cuckoo search (EGCS) and reinforcement learning cuckoo search (RLCS). In contrast to the original CS algorithm, the proposed CS variations attempt to balance between exploration of new solutions and exploitation of current solutions. Several experiments were conducted using widely known

Table 15 ANOVA table with 30 runs and $D = 100$ for benchmark functions

$f(X)$	Source of variation	SS	*df*	MS	*F*	*P* value	*F* crit
F1	Between groups	4.58E+08	6	76,264,024	106.4991	6.02E−60	2.143453
	Within groups	1.45E+08	203	716,100.1			
F2	Between groups	32.69145131	6	5.4485752	552.4589	8.7E−123	2.143453
	Within groups	2.002069038	203	0.0098624			
F3	Between groups	0	6	0	65 535	Error	2.143453
	Within groups	0	203	0			
F4	Between groups	4.14045E−05	6	6.9E−06	11.24934	7.78E−11	2.143453
	Within groups	0.000124528	203	6.13E−07			
F5	Between groups	1.02359E−06	6	1.71E−07	2.811819	0.011926	2.143453
	Within groups	1.23164E−05	203	6.07E−08			
F6	Between groups	41,116.36295	6	6852.727	0.516827	0.795213	2.143453
	Within groups	2,691,621.688	203	13,259.22			
F7	Between groups	4.91028E−06	6	8.18E−07	1.667138	0.130686	2.143453
	Within groups	9.96505E−05	203	4.91E−07			
F8	Between groups	4.66033E+11	6	7.77E+10	3107.928	9.6E−197	2.143453
	Within groups	5,073,294,807	203	24,991,600			
F9	Between groups	1.51616E+12	6	2.53E+11	1.588652	0.151999	2.143453
	Within groups	3.22894E+13	203	1.59E+11			
F10	Between groups	4440.751193	6	740.1252	205.9452	1.87E−83	2.143453
	Within groups	729.5405906	203	3.593796			
F11	Between groups	3.6965E+29	6	6.16E+28	92.85728	1.81E−55	2.143453
	Within groups	1.34685E+29	203	6.63E+26			
F12	Between groups	8.95323E+26	6	1.49E+26	12.2705	9.07E−12	2.143453
	Within groups	2.46866E+27	203	1.22E+25			
F13	Between groups	3.92671E+12	6	6.54E+11	244.3201	5.58E−90	2.143453
	Within groups	5.43768E+11	203	2.68E+09			
F14	Between groups	629,714.1035	6	104952.4	11.56768	3.96E−11	2.143453
	Within groups	1,841,797.146	203	9072.892			
F15	Between groups	441,401.7007	6	73,566.95	208.7042	5.87E−84	2.143453
	Within groups	71,556.24806	203	352.4938			
F16	Between groups	1,522,111.797	6	253,685.3	49.09799	5.67E−37	2.143453
	Within groups	1,048,884.327	203	5166.918			
F17	Between groups	1,300,040.034	6	216,673.3	8.199573	5.94E−08	2.143453
	Within groups	5,364,265.391	203	26,424.95			
F18	Between groups	0.122992204	6	0.020499	4.952971	9.38E−05	2.143453
	Within groups	0.840149501	203	0.004139			
F19	Between groups	1.50679E+12	6	2.51E+11	13.5583	6.36E−13	2.143453
	Within groups	3.76005E+12	203	1.85E+10			
F20	Between groups	4.58E+08	6	76,264,024	106.4991	6.02E−60	2.143453
	Within groups	1.45E+08	203	716,100.1			

benchmark functions. The experimental results showed that the use of any of the proposed selection schemes in most of the tested functions provides improvement of the performance of CS except when the GCS was applied to shifted benchmark functions. This is because that GCS may converge slower to a solution than the original CS algorithm

Table 16 ANOVA table with 30 runs and $D = 1000$ for benchmark functions

f(X)	Source of variation	SS	*df*	MS	*F*	*P* value	*F* crit
F1	Between groups	1.81468E−09	6	3.024E−10	1.187889	0.314089	2.143453
	Within groups	5.16856E−08	203	2.546E−10			
F2	Between groups	1.81468E−09	6	3.024E−10	1.187889	0.314089	2.143453
	Within groups	5.16856E−08	203	2.546E−10			
F3	Between groups	0	6	0	65535	Error	2.143453
	Within groups	0	203	0			
F4	Between groups	4.73606E−05	6	7.89E−06	12.56693	4.89E−12	2.143453
	Within groups	0.000127507	203	6.28E−07			
F5	Between groups	2.61372E−05	6	4.36E−06	248.705	1.15E−90	2.143453
	Within groups	3.55565E−06	203	1.75E−08			
F6	Between groups	2.05012E+25	6	3.42E+24	1	0.42645	2.143453
	Within groups	6.93622E+26	203	3.42E+24			
F7	Between groups	1.90249E−06	6	3.17E−07	0.704426	0.64637	2.143453
	Within groups	9.13761E−05	203	4.5E−07			
F8	Between groups	4.57865E+11	6	7.63E+10	2486.242	5E−187	2.143453
	Within groups	6,230,727,488	203	30,693,239			
F9	Between groups	3.24036E+13	6	5.4E+12	7.192093	5.7E−07	2.143453
	Within groups	1.52434E+14	203	7.51E+11			
F10	Between groups	3781.900824	6	630.3168	139.6392	3.05E−69	2.143453
	Within groups	916.3205331	203	4.513894			
F11	Between groups	5.66985E+61	6	9.45E+60	1246.897	3.3E−157	2.143453
	Within groups	1.53846E+60	203	7.58E+57			
F12	Between groups	7.61332E+66	6	1.27E+66	12.28674	8.76E−12	2.143453
	Within groups	2.09644E+67	203	1.03E+65			
F13	Between groups	4.957E+18	6	8.26E+17	244.4104	5.4E−90	2.143453
	Within groups	6.86189E+17	203	3.38E+15			
F14	Between groups	2,538,303,131	6	4.23E+08	195.9575	1.38E−81	2.143453
	Within groups	438,254,562	203	2,158,889			
F15	Between groups	7,053,713,502	6	1.18E+09	13.33605	1E−12	2.143453
	Within groups	17,895,156,698	203	88,153,481			
F16	Between groups	3,287,667,967	6	5.48E+08	15.97831	5.06E−15	2.143453
	Within groups	6,961,485,096	203	34,293,030			
F17	Between groups	692,201,676.1	6	1.15E+08	585.3387	3.4E−125	2.143453
	Within groups	40,010,150.54	203	197,094.3			
F18	Between groups	8.00275E−09	6	1.33E−09	1.187889	0.314089	2.143453
	Within groups	2.27934E−07	203	1.12E−09			
F19	Between groups	1.64E+15	6	2.74E+14	13.54695	6.51E−13	2.143453
	Within groups	4.11E+15	203	2.02E+13			
F20	Between groups	8.46994E+13	6	1.41E+13	238.4409	4.85E−89	2.143453
	Within groups	1.20183E+13	203	5.92E+10			

because searching for a single best solution may lead to premature convergence (i.e., stuck early in a sub-optimal solution).

In the current paper, the proposed selection schemes were applied to the cuckoo selection step. In future work, we plan to apply the proposed selection schemes to the other two selection steps (host selection and greedy selection).

References

1. Yang, X.-S.; Deb, S.: Cuckoo search via lévy flights. In World Congress on Nature and Biologically Inspired Computing, 2009. NaBIC 2009, pp. 210–214. IEEE(2009)
2. AlRashidi, M.R.; El-Naggar, K.M.; AlHajri, M.F.: Convex and non-convex heat curve parameters estimation using cuckoo search. Arab. J. Sci. Eng. **40**(3), 873–882 (2015)
3. Rakhshani, H.; Rahati, A.: Intelligent multiple search strategy cuckoo algorithm for numerical and engineering optimization problems. Arab. J. Sci. Eng. **42**, 1–27 (2016)
4. Yang, X.-S.; Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
5. Al-Betar, M.A.; Doush, I.A.; Khader, A.T.; Awadallah, M.A.: Novel selection schemes for harmony search. Appl. Math. Comput. **218**(10), 6095–6117 (2012)
6. Goldberg, D.E.; Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms **1**, 69–93 (1991)
7. Shehab, M.; Al-Betar, M.A.; Khader, A.T.: New selection schemes for particle swarm optimization. In: ICIT 2015: the 7th International Conference on Information Technology, pp. 17–25. (2015)
8. Valian, E.; Mohanna, S.; Tavakoli, S.: Improved cuckoo search algorithm for feedforward neural network training. Int. J. Artif. Intell. Appl. **2**(3), 36–43 (2011)
9. Valian, E.; Mohanna, S.; Tavakoli, S.: Improved cuckoo search algorithm for global optimization. Int. J. Commun. Inf. Technol. **1**(1), 31–44 (2011)
10. Valian, E.; Tavakoli, S.; Mohanna, S.; Haghi, A.: Improved cuckoo search for reliability optimization problems. Comput. Ind. Eng. **64**(1), 459–468 (2013)
11. Tuba, M.; Subotic, M.; Stanarevic, N.: Modified cuckoo search algorithm for unconstrained optimization problems. In: Proceedings of the 5th European Conference on European Computing Conference, pp. 263–268. World Scientific and Engineering Academy and Society (WSEAS), (2011)
12. Walton, S.; Hassan, O.; Morgan, K.; Brown, M.R.: Modified cuckoo search: a new gradient free optimisation algorithm. Chaos Solitons Fractals **44**(9), 710–718 (2011)
13. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of an International Conference on Genetic Algorithms and their applications, pp. 101–111. Hillsdale, New Jersey (1985)
14. Geem, Z.W.; Kim, J.H.; Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
15. Eberhart, R.C.; Kennedy, J., et al.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, pp. 39–43. New York (1995)
16. Kennedy, J.; Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. (1995)
17. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. U Michigan Press, Ann Arbor (1975)
18. Hancock, P.J.B.: An empirical comparison of selection methods in evolutionary algorithms. In: AISB Workshop on Evolutionary Computing, pp. 80–94. Springer, (1994)
19. Abed-alguni, B.H.; Chalup, S.K.; Henskens, F.A.; Paul, D.J.: Erratum to: a multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers. Vietnam. J. Comput. Sci. **2**(4), 227–227 (2015)
20. Abed-alguni, B.H.; Chalup, S.K.; Henskens, F.A.; Paul, D.J.: A multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers. Vietnam. J. Comput. Sci. **2**(4), 213–226 (2015)
21. Khater Aziz, A.; El-Bardini, M.; El-Rabaie, N.: Embedded adaptive fuzzy controller based on reinforcement learning for dc motor with flexible shaft. Arab. J. Sci. Eng. **40**(8), 2389–2406 (2015)
22. Watkins, C.: Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England (1989)
23. Watkins, C.; Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**(3), 279–292 (1992)
24. Humphrys, M.: Action selection methods using reinforcement learning. From Anim. Animats **4**, 135–144 (1996)
25. Tokic, M.: Adaptive ε-greedy exploration in reinforcement learning based on value differences. In: Annual Conference on Artificial Intelligence, pp. 203–210. Springer, (2010)
26. Abed-alguni, B.H.: Bat q-learning algorithm. Jordan. J. Comput. Inf. Technol. (JJCIT) **3**(1), 56–77 (2017)
27. Abed-Alguni, B.H.K.: Cooperative reinforcement learning for independent learners. PhD thesis, Faculty of Engineering and Built Environment, School of Electrical Engineering and Computer Science, The University of Newcastle, Australia (2014)
28. Sutton, R.-S.; Barto, A.-G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
29. Abed-Alguni, B.H.; Paul, D.J.; Chalup, S.K.; Henskens, F.A.: A comparison study of cooperative q-learning algorithms for independent learners. Int. J. Artif. Intell. **14**(1), 71–93 (2016)
30. Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM SIGART Bull. **2**(4), 160–163 (1991)
31. Yang, X.-S.: Bat algorithm and cuckoo search: a tutorial. In: Yang, X.-S. (ed.) Artificial Intelligence, Evolutionary Computing and Metaheuristics, pp. 421–434. Springer, Berlin (2013)
32. Hasan, B.H.F.; Doush, I.A.; Maghayreh, E.A.; Alkhateeb, F.; Hamdan, M.: Hybridizing harmony search algorithm with different mutation operators for continuous problems. Appl. Math. Comput. **232**, 1166–1182 (2014)
33. Alabsi, F.; Naoum, R.: Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system. J. Emerg. Trends Comput. Inf. Sci. **3**(7), 1053–1058 (2012)
34. Omran, M.G.H.; Mahdavi, M.: Global-best harmony search. Appl. Math. Comput. **198**(2), 643–656 (2008)
35. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report#2005005, IIT Kanpur, India, Nanyang Technological University, Singapore (2005)
36. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, 2005. Nanyang Technological University Singapore. (2005)