Engineering Optimization

ISSN: 0305-215X (Print) 1029-0273 (Online) Journal homepage: http://www.tandfonline.com/loi/geno20

Cuckoo search with composite flight operator for numerical optimization problems and its application in tunnelling

Hossein Abedi Firouzjaee, Javidan Kazemi Kordestani & Mohammad Reza Meybodi

To cite this article: Hossein Abedi Firouzjaee, Javidan Kazemi Kordestani & Mohammad Reza Meybodi (2017) Cuckoo search with composite flight operator for numerical optimization problems and its application in tunnelling, Engineering Optimization, 49:4, 597-616, DOI: 10.1080/0305215X.2016.1206535

To link to this article: http://dx.doi.org/10.1080/0305215X.2016.1206535

ENGINEERING OPTIMIZATION, 2017
VOL. 49, NO. 4, 597–616
http://dx.doi.org/10.1080/0305215X.2016.1206535

Taylor & Francis
Taylor & Francis Group

Cuckoo search with composite flight operator for numerical optimization problems and its application in tunnelling

Hossein Abedi Firouzjaee[a], Javidan Kazemi Kordestani[b] and Mohammad Reza Meybodi[a]

[a]Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran; [b]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

ABSTRACT

This article presents two modifications of the cuckoo search (CS) algorithm for numerical optimization problems. The first modified algorithm is cuckoo search with composite flight (CSCF), which is aimed at improving the performance of the CS by introducing a novel composite flight operator in the standard CS. The main idea of the composite flight operator is to allow a new cuckoo egg to be generated by taking different random walks. The second modified algorithm is aimed at improving the technique used by CSCF by adaptively choosing the flight operator at each time step via the learning automata. Moreover, a model based on support vector regression and CSCF, in which CSCF is used to adjust the parameters of support vector regression (*i.e.* C and γ), is developed to estimate the penetration rate of a tunnel-boring machine. The experimental results show that the proposed modifications can significantly improve the performance of the standard CS.

ARTICLE HISTORY
Received 28 November 2015
Accepted 21 June 2016

KEYWORDS
Cuckoo search; global numerical optimization; composite flight; support vector regression; tunnel-boring machine

1. Introduction

Nature-inspired algorithms have gained considerable attention from the scientific community as they can serve as optimization tools in a vast variety of problem domains. One of the recently proposed nature-inspired algorithms is cuckoo search (CS) (Yang and Deb 2009), which imitates the parasitic behaviour of the cuckoo bird. Theoretical studies have shown that CS is an efficient algorithm that can guarantee global convergence (Wang, He, *et al.* 2012). A study by Civicioglu and Besdok (2013) revealed that CS can perform better than particle swarm optimization (PSO) and artificial bee colony in solving numerical optimization problems. However, there is still room for improving the performance of CS.

In this article, two modifications to the CS are suggested. The first suggested modification is a cuckoo search with composite flight (CSCF). The primary objective of the composite flight is to generate a new egg through competition among three different eggs that are generated by three different random walks, *i.e.* Levy flight, Gaussian distribution and Cauchy distribution. The idea of combining the knowledge of different search strategies to generate a new individual has already achieved good results in composite differential evolution (CoDE) (Wang, Cai, and Zhang 2011). However, the composition in the present study differs from that of CoDE. The main difference is that CoDE employs three trial vector generation strategies, each of which is coupled with a randomly chosen parameter

CONTACT Javidan Kazemi Kordestani Javidan.kazemi@gmail.com

setting from the parameter candidate pool, whereas CSCF uses three types of random walk to compose. The second modification is an adaptive cuckoo search (ACS) algorithm which aims to automate the selection of the flight operator for each individual with respect to its search progress. For this purpose, a learning automata (LA)-based approach is introduced to adapt the selection of random walk in CSCF. Finally, a hybrid method based on support vector regression (SVR) and CSCF, called SVR-CSCF, is presented in which CSCF is used to select the optimal values for the SVR parameters. The proposed hybrid model is then applied to predict the performance of a tunnel-boring machine (TBM) in hard rock conditions.

The rest of this article is structured as follows. Section 2 gives an introduction to the basic CS algorithm. In Section 3, a comparison of Gaussian, Cauchy and Levy distributions is provided. A brief description of LA is given in Section 4. Section 5 describes details of the SVR. Section 6 presents a review on some previous attempts to improve the performance of CS on global optimization problems. The proposed algorithms are presented in Section 7. An empirical study of the proposed method on 25 test functions of CEC 2005 and prediction of TBM performance in hard rock conditions are given in Section 8. Finally, Section 9 concludes the article and makes some suggestions for future work.

2. Cuckoo search

CS is a stochastic population-based algorithm which was proposed by Yang and Deb (2009), motivated by the breeding strategy of some cuckoos. These birds show extraordinary behaviour when they want to lay their eggs. Instead of building their own nests, cuckoos lay their eggs in the nest of other species, making those species responsible for caring for the cuckoo chicks. This behaviour is the main inspiration behind the CS algorithm.

For the sake of simplicity of description and implementation, Yang and Deb (2009) made three idealized assumptions, as follows:

- Each cuckoo lays only one egg at a time and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs are transferred to the next generations.
- The number of nests is fixed and there is a probability that a host can recognize a cuckoo egg. If this happens, the host can either throw out the egg or leave the nest and construct a new one.

Based on the above assumption, the search process for standard CS can be summarized as shown in Algorithm 1.

Algorithm 1. CS via Levy flight

```
1. begin
2.     Objective function f(x),  x = (x_1, x_2, ..., x_n)^T;
3.     Generate initial population of n host nests x_i (i = 1, 2, ..., n);
4.     while (e < MaxEvals) or (Stopping criteria not met) do
5.         Get a cuckoo randomly by Levy flight;
6.         Evaluate its quality/fitness F_i;
7.         Choose a nest among n (say, j) randomly;
8.         if F_i > F_j then
9.             Replace j by the new solution;
10.        end-if
11.        A fraction (p_a) of worst nests are abandoned and new ones are built;
12.        Keep the best solutions (or nests with quality solutions);
13.        Rank the solutions and find the current best;
14.    end-while
15. end
```

CS has been successfully applied to a vast variety of science and engineering problems, including phase equilibrium and stability calculations (Bhargava, Fateen, and Bonilla-Petriciolet 2013), design of power systems for energy self-sufficient farms (Piechocki *et al.* 2014), selection of optimal machining parameters in milling operations (Yildiz 2013) and multi-objective optimization problems (Chandrasekaran and Simon 2012; Yang and Deb 2013).

3. Comparison between Gaussian, Cauchy and Levy distributions

There is a class of probability distributions of an infinite second moment that yields a stable process. Such a probability distribution is called the Levy probability distribution and has the following form (Wang, He, *et al.* 2012):

$$f(x;\beta,\gamma)=\frac{1}{\pi}\int_0^{\infty}\exp(-\gamma s^{\beta})\cos(sx)\mathrm{d}s,\quad x\in(-\infty,+\infty) \tag{1}$$

where the parameter $\gamma > 0$ is the scale factor, $0 < \beta < 2$ is the shape parameter, and the distribution is symmetrical with respect to $x = 0$.

Two of the famous stable distributions (Gaussian and Cauchy distributions) are special cases of the Levy distribution. Figure 1 shows the importance of parameter β in the above equation.

Regarding Figure 1, decreasing the value of β results in more samples being generated farther away from the zero (distributions are symmetrical around zero). This phenomenon shows the critical importance of β in controlling the exploration and exploitation of CS. High values of β make CS have much more exploitation; conversely, low values of beta increase the exploration of CS.

To justify the above statement, without loss of generality, $(x;\beta)=(1/\pi)\int_0^{\infty}\exp(-s^{\beta})\cos(sx)\mathrm{d}s$ is defined by setting the scale factor $\gamma = 1$. It can be easily shown that the standard Cauchy distribution is a special case of $l(x;\beta)$ when $\beta = 1$ using the following equation:

$$f(x;\beta=1)=\frac{1}{\pi}\int_0^{\infty}\exp(-s)\cos(sx)\mathrm{d}s=\frac{1}{\pi(1+x^2)},\quad x\in(-\infty,+\infty) \tag{2}$$

where $(1/\pi(1+x^2))$ is the famous probability density function of the Cauchy distribution. $l(x;\beta)$ and its cumulative probability distribution $Pr_{l(x;\beta)}$ have the following limit properties (Lee and Yao

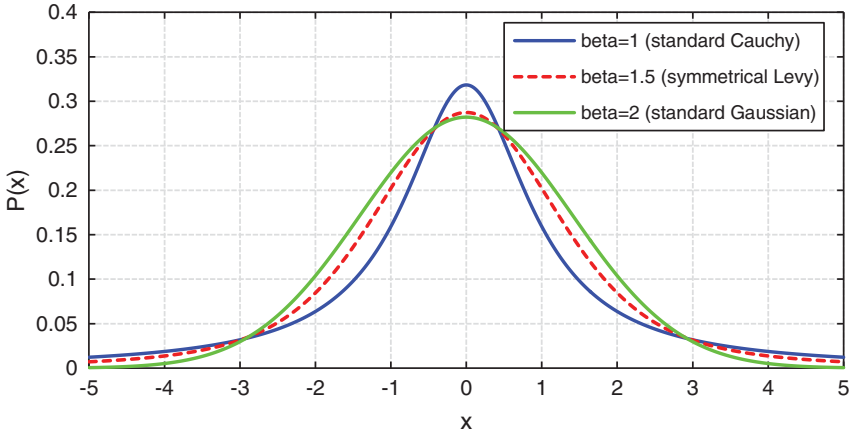

Figure 1. Levy probability distribution for different values of β.

2004):

$$\lim_{|x|\to\infty} l(x;\beta) \sim x^{-(\beta+1)}, \quad \beta \in (0,2) \tag{3}$$

$$\lim_{|x|\to\infty} Pr_{l(x;\beta)} \sim 1 - x^{-\beta}, \quad \beta \in (0,2) \tag{4}$$

That is, the probability density function and cumulative density function of $l(x;\beta)$ are asymptotically equivalent to $x^{-(\beta+1)}$ and $1 - x^{-\beta}$, respectively.

To investigate the effect of β on generated flights (jumps), $T_{l(x;\beta)}(x)$ is defined as the limit of the right tail function of $l(x;\beta)$ as follows:

$$T_{l(x;\beta)}(x) = \lim_{x\to+\infty} 1 - Pr_{l(x;\beta)} \sim x^{-\beta}, \quad \beta \in (0,2) \tag{5}$$

This function enables us to further analyse the effect of β on generated flights (jumps) in extreme values ($x \gg 0$). For $\beta \in \{1, 1.5, 2\}$, consider:

$$T_{l(x;1)}(x) = Pr_{\beta=1}\{X > x\} = x^{-1} \tag{6}$$

$$T_{l(x;1.5)}(x) = Pr_{\beta=1.5}\{X > x\} = x^{-1.5} \tag{7}$$

$$T_{l(x;2)}(x) = Pr_{\beta=2}\{X > x\} = x^{-2} \tag{8}$$

Setting $x = 10,000$ results in the following asymptotically approximated probabilities:

$$T_{l(x;1)}(10000) = Pr_{\beta=1}\{X > 10000\} = 0.0001 \tag{9}$$

$$T_{l(x;1.5)}(10000) = Pr_{\beta=1.5}\{X > 10000\} = 0.000001 \tag{10}$$

$$T_{l(x;2)}(10000) = Pr_{\beta=2}\{X > 10000\} = 0.00000001 \tag{11}$$

As can be seen, $(Pr_{\beta=1}\{X > 10000\}/Pr_{\beta=2}\{X > 10000\}) = 10000$. The same analysis could be carried out on the left tail function with similar results.

The above analysis shows that by decreasing the value of β from 2 to 0, the probability of generating jumps with great length increases. Conversely, by increasing the value of β, the probability of generating jumps with great length decreases.

4. Learning automata

LA are adaptive decision-making devices operating in an unknown random environment that learn to choose the optimal action from a set of available actions through iterative interaction with the environment (Narendra and Thathachar 1974; Thathachar and Sastry 2002). The learning process of the automaton can be divided into three steps:

(1) The automaton selects an action from a set of finite actions and applies it to the environment.
(2) The environment evaluates the effectiveness of the selected action and generates a response for the learning automaton.
(3) The automaton uses the response of the environment to update its internal action probabilities.

By repeating this three-step procedure, the learning automaton will be gradually guided towards selecting the optimal action.

To just name a few, LA have a wide variety of applications, such as in parameter control (Rezvanian and Meybodi 2010; Hashemi and Meybodi 2011; Kordestani, Ahmadi, and Meybodi 2014), wireless sensor networks (Moghis, Meybodi, and Esnaashari 2010; Esnaashari and Meybodi 2011), the vertex colouring problem (Akbari Torkestani and Meybodi 2011), traffic signal control (Barzegar *et al.* 2011) and information retrieval (Akbari Torkestani 2012).

4.1. Finite action-set learning automaton

A finite action-set learning automaton (FALA) is a type of LA which is defined with a quadruple $\{A, P(k), B, T\}$, where $A = \{\alpha_1, \ldots, \alpha_n\}$ is the set of finite actions, $P(k) = [p_1(k), \ldots, p_n(k)]$ denotes the vector of action probabilities at time step k, whose elements should satisfy the following conditions:

$$\sum_{i=1}^{n} p_i(k) = 1 \tag{12}$$

$$\forall i \epsilon \{1, \ldots, n\}: 0 \leq p_i(k) \tag{13}$$

B is the set of all possible outputs received from the environment. Finally, T is a learning algorithm, which is used to modify the action probabilities so that $P(k+1) = T[\alpha(k), \beta(k), P(k)]$.

4.2. Learning algorithm

There are many algorithms for updating the action probabilities of an automaton. In this article, a learning algorithm called the pursuit algorithm (Thathachar and Sastry 2003) was applied that uses the histories of actions to update $P(k)$.

Let $Z_i(k)$ be the sum of all rewards obtained in response to action α_i until instant k, and $\eta_i(k)$ the number of times that action α_i is chosen until instant k. $P(k)$ is then updated using the following formula:

$$P(k+1) = P(k) + \lambda(e_{M(k)} - P(k)) \tag{14}$$

where λ is the learning rate ($0 < \lambda < 1$) and $e_{M(k)}$ is the unit vector with all elements equal to zero, except one whose position is determined as follows:

$$M(k) = \max_i \left\{ \frac{Z_i(k)}{\eta_i(k)} \right\}, \quad i = 1, 2, \ldots, n \tag{15}$$

An interesting aspect of the pursuit algorithm is that it does not involve $\beta(k)$ directly.

4.3. Environment

The random environment that interacts with the automaton is characterized by a response set B, where $\beta(k) \in B$. Different environments can be modelled using B. In the P-model, $B = \{0, 1\}$, in which the value '0' corresponds to an unfavourable response or punishment and '1' corresponds to a favourable response or reward. In the Q-model, $B = \{b_1, \ldots, b_q\}$, $b_i \in [0, 1]$ and $q < \infty$. Finally, in the S-model, $B = [0, 1]$ (Thathachar and Sastry 2003). The framework for the learning process of a FALA with a pursuit learning scheme in a stationary environment is shown in Algorithm 2.

Algorithm 2. Framework for the learning process of FALA with a pursuit learning scheme in a random unknown environment

1. **begin**
2. **Let** $A = \{\alpha_1, \ldots, \alpha_n\}$ be the set of finite actions, where n is the number of actions;
3. **Let** $P = [p_1, \ldots, p_n]$ be the action probability vector of the LA;
4. **while** (the automaton converges to one of its actions) **do**
5. The learning automaton chooses one of its actions based on the probability distribution P;
6. The environment evaluates the action and calculates a reinforcement signal;
7. The environment sends a feedback to the learning automaton;
8. The automaton updates its probability vector using Equation (14);
9. **end-while**
10. **End**

5. Support vector regression

Support vector machine (Xuegong 2000) is a classification method developed for pattern recognition problems. In 1997, Vapnik *et al.* (1996) proposed a modified version of the support vector machine for function approximation called SVR. Given a data set of N training points as $\{(x_1, y_1), \ldots, (x_N, y_N)\} \subset \aleph \times \Re$, where $\aleph$ denotes the space of the input data, SVR attempts to find a function for which the approximation of target value y_i is in its radius ε. For the case of a linear function with the following formula:

$$f(x) = \langle \omega, x \rangle + b, \quad \omega \in \aleph \quad \text{and} \quad b \in \mathcal{R} \tag{16}$$

where ω and b are the weight vector and interception of the line at the origin, respectively, the problem of finding a regression model for the function given in Equation (16) could be expressed as:

$$\text{minimize} \quad \frac{1}{2}||\omega||^2 \tag{17}$$

$$\text{subject to} \quad \begin{cases} \langle \omega, x \rangle + b - y_i \leq \epsilon \\ y_i - \langle \omega, x \rangle - b \leq \epsilon \end{cases} \tag{18}$$

In many cases, there is no such linear function which approximates the target values without any error. Therefore, the problem of Equation (17) is converted to the following optimization problem:

$$\text{minimize} \quad \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \tag{19}$$

$$\text{subject to} \quad \begin{cases} \langle \omega, x \rangle + b - y_i \leq \epsilon + \xi_i \\ y_i - \langle \omega, x \rangle - b \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{20}$$

where C is called the soft margin constant, which controls the penalty that is applied when an approximated output for a data point violates the constraints. The parameters ξ_i and ξ_i^* are slack variables which determine the amount that approximated values of data point x_i deviate from the target value. To solve the constrained optimization problem in Equation (19), the Lagrangian of the original problem is formed in the following manner:

$$\begin{aligned} L = \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) - \sum_{i=1}^{N} \alpha_i (\langle \omega, x \rangle + b - y_i + \epsilon + \xi_i) \\ - \sum_{i=1}^{N} \alpha_i^* (y_i - \langle \omega, x \rangle - b + \epsilon + \xi_i^*) - \sum_{i=1}^{N} (\eta_i \xi_i + \eta_i^* \xi_i^*) \end{aligned} \tag{21}$$

Solving the above equation gives:

$$\omega = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) x_i \tag{22}$$

$$b = y_i - \langle \omega, x_i \rangle - \epsilon \tag{23}$$

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \langle \omega, x_i \rangle + b \tag{24}$$

The main advantage of SVR is that in the process of calculation, $\langle ., . \rangle$ can be replaced with a kernel function $K(x, y)$ that has the same result as the dot product, but can be used to map the initial data points to a space with higher dimensions.

Kernels are similarity measures that are used to quantify the similarity between data points. Some popular kernels, such as sigmoid and radial basis functions, have been frequently used in the literature. In this article, the radial basis function was used, which is defined as follows:

$$k(x, y) = \exp(-\gamma ||x - y||^2) \tag{25}$$

where γ is the width of the radial basis function.

The values of C and γ have a great effect on the performance of SVR. As mentioned earlier, the constant C in SVR is a soft constant margin that governs the generality of the model. There is a trade-off between the generality and training accuracy of the model that is influenced by this parameter. On the one hand, a large value of C results in overfitting of the model, and on the other hand, a small value of C causes the model to have low accuracy on the training data.

6. Related works

This section presents six types of approach for improving the performance of CS.

6.1. Changing the initialization pattern of cuckoo search

It has been confirmed that the initialization method in population-based algorithms has a great influence on the exploration behaviour of the algorithms in the early iterations of the execution (Rahnamayan, Tizhoosh, and Salama 2007). In the field of CS, Shatnawi and Nasrudin (2011) used centroidal Voronoi tessellations to generate the initial positions of the nests in CS. Their experimental results show a significant improvement over the standard CS.

6.2. Hybridizing cuckoo search with other methods

Various studies have tried to combine the desirable features of CS with other optimization methods. For example, Wang, Guo, *et al.* (2012) proposed a hybrid algorithm of differential evolution and CS, which they called DE/CS. In DE/CS, the mutation and crossover of DE are applied to replace the process of cuckoo selection in the standard CS. In this way, DE/CS can combine the exploration ability of DE with the exploitation ability of CS. Their experimental results indicate that DE/CS could outperform the standard CS in the uninhabited combat air vehicle path planning problem.

6.3. Parallel search

With the aim of speeding up the search process of CS, some authors have introduced parallelization into the standard CS. Xu *et al.* (2014) presented a parallel CS using MapReduce. In this method, they divided the search space into equally sized cells. The search process of CS in the cells is followed simultaneously. Their results show that parallelizing the search process of CS can significantly improve the running time of the algorithm.

6.4. Adjusting the size of the Levy flight step size

Another approach for improving the performance of the standard CS is to control the exploration and exploitation ability of CS by adjusting the size of the Levy flight step size. For example, Walton *et al.* (2011) proposed a modified cuckoo search (MCS), with a time-varying Levy flight step size. In this model, the value of α is decreased according to the following equation:

$$\alpha = \frac{A}{\sqrt{G}} \tag{26}$$

where A is the initial value of the Levy flight step size, and G is the generation number. This scheme encourages the exploitation ability of the CS at the later stages of the search process.

6.5. Adding information exchange between eggs

In the standard CS, there is no information exchange between individuals. In other words, the search process of each individual is performed separately. A group of studies has focused on establishing an information network between eggs. Walton *et al.* (2011) proposed a modification of CS in which the eggs are divided into two groups: a fraction of eggs with the best fitness is marked as top eggs, and the remaining eggs are normal eggs. For each egg in the top group, a second egg from the top group is selected randomly. A new egg is then generated on the line connecting these two top eggs. Simulations on a set of well-known functions show that exchanging information between eggs when generating new one can improve the performance of CS.

6.6. Changing the distribution of the steps

The standard CS algorithm uses the Levy distribution to generate new solutions. While the Levy distribution can increase the exploration ability of CS, the exploitation capacity of this algorithm remains poor. Therefore, researchers have tried to enhance the local search capability of CS by changing the distribution of space or adding some local search methods. For example, Zheng and Zhou (2012) proposed a CS based on the Gaussian distribution. In this method, the Levy flight was replaced with a Gaussian distribution, as follows:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \alpha \oplus \sigma_0(-\mu k) \tag{27}$$

where σ_0 and μ are constants, and k is the current generation.

Nguyen *et al.* (2014) applied CS with three different distributions, *i.e.* Levy, Cauchy and Gaussian, to solve a short-term hydrothermal scheduling problem considering cascaded hydropower plants.

7. Proposed modified cuckoo search algorithms

This section provides a complete description of the proposed methods for solving the following continuous global optimization problem:

$$\begin{aligned} &\text{Minimize:} \quad f(\vec{x}), \vec{x} = (x_1, x_2, \ldots, x_D) \in s \\ &\text{Subject to:} \quad x_i \in [lb_i, ub_i] \end{aligned} \tag{28}$$

where $f(\vec{x})$ is a continuous real-valued objective function to be minimized, and s is the solution space. Finally, lb_i and ub_i are the box constraints corresponding to ith dimension of the search space, *i.e.* $\forall i \in \{1, \ldots, D\}, -\infty < l_i < u_i < \infty$.

7.1. Cuckoo search with composite flight operator

In the first scheme for generating the candidate solutions, a memetic algorithm is proposed that uses three cases of Levy distribution. In this modification, candidate solutions are generated by standard Cauchy distribution (Levy distribution with $\beta = 1$), Levy distribution with $\beta = 1.5$ and standard Gaussian distribution (Levy distribution with $\beta = 2$) and the current previous solutions are replaced with the best newly generated candidate solution. The first two operators (Cauchy and Levy flights) use heavy-tailed distributions which improve the global search, and the last operator (Gauss flight) uses a light-tailed distribution that could improve the local search of the original CS.

7.1.1. Initialization

The proposed CSCF starts with a population of NP randomly generated eggs in a D-dimensional search space. Each egg is a potential solution to an optimization problem which is represented by $\vec{X}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. The initial population of eggs is simply randomized into the boundary of the search space according to a uniform distribution, as follows:

$$\vec{X}_i = lb_j + rand_j[0,1] \times (ub_j - lb_j) \tag{29}$$

where $i \epsilon [1, 2, \ldots, NP]$ is the index of the ith egg of the population, $j \epsilon [1, 2, \ldots, D]$ represents the jth dimension of the search space, and $rand_j[0, 1]$ is a uniformly distributed random number corresponding to the jth dimension. Finally, lb_j and ub_j are the lower and upper bounds, respectively, of the search space corresponding to the jth dimension of the search space.

7.1.2. Composite flight operator

After initialization of the cuckoos in the search space, the composite flight operator is performed on each cuckoo as follows:

(1) Three eggs are generated using Levy, Cauchy and Gaussian distributions.
(2) The fitness of the generated eggs is evaluated.
(3) The fittest egg is transferred to the next generation, if it is fitter than a randomly chosen nest j.

Algorithm 3 shows the general procedure of CSCF.

Algorithm 3. CSCF algorithm

```
1. begin
2.     Objective function f(x),  x = (x1,  x2,  ...,  xn)^T;
3.     Generate initial population of n host nests xi(i = 1,  2,  ...,  n);
4.     while (e < MaxEvals) or (Stopping criteria not met) do
5.         Get a cuckoo randomly (say, i) and generate three new solutions by Levy, Cauchy and
           Gauss flight;
6.         Evaluate the fitness of the new solutions F_Levy, F_Cauchy and F_Gauss;
7.         Choose a nest among n (say, j) randomly;
8.         if  max{F_Levy,  F_Cauchy,  F_Gauss} > Fj  then
9.             Replace j by the new solution;
10.        end-if
11.    A fraction (pa) of worst nests are abandoned and new ones are built;
12.    Keep the best solutions (or nests with quality solutions);
13.    Rank the solutions and find the current best;
14.  end-while
15. end
```

7.2. Adaptive cuckoo search

The basis for the second adaptation is to choose the distribution of steps for each cuckoo adaptively. In the proposed ACS, each cuckoo is equipped with a three-action learning automaton. The automaton of each cuckoo is responsible for choosing the distribution of the flight according to the feedback received from the environment. Therefore, ACS differs from CSCF in that the distribution of steps for each cuckoo is chosen adaptively, based on its search progress.

In ACS, a three-action learning automaton ($\mathrm{LA_{fly}}$) is associated with each cuckoo of the population, which is responsible for choosing the proper distribution for the fly operation, $|\mathrm{LA_{fly}}| = |popsize| = 30$. The actions of $\mathrm{LA_{fly}}$ correspond to the Levy distribution, Cauchy distribution and Gaussian distribution. The initial selection probability of each of these three actions is

set to 1/3. At each generation, the LA_{fly} chooses a distribution for the fly operation according to Algorithm 4.

Algorithm 4. Pseudo-code for selecting the distribution by LA

```
1. begin
2.   rnd : = rand();
3.   sum : = 0;
4.   selectedDistribution : = Null;
5.   for each action j∈[1, 2, 3] of LAdo
6.     if rnd < sum + p_j then
7.     selectedDistribution : = j;
8.     break;
9.     end-if
10.  sum : = sum + p_j;
11.  end
12. end
```

Afterwards, the cuckoo lays an egg with the selected distribution. After the new egg has been generated, the automata update their probability vectors based on the reinforcement signal they receive from the environment. Based on the received reinforcement signals, the automata can determine whether their actions were right or wrong, and update their probability vector accordingly. In this work, the reinforcement signal is generated as follows:

$$\text{Reinforcement signal} = \begin{cases} 0 & \text{if } F(\vec{X}_{i,G}) < F(\vec{X}_{i,G+1}) \\ 1 & \text{otherwise} \end{cases} \tag{30}$$

where $F(\vec{X}_{i,G})$ is the fitness value of the ith cuckoo at the Gth generation, and $F(\vec{X}_{i,G+1})$ is the fitness value of the generated egg by the ith cuckoo after evolving by the selected distribution at generation $G+1$.

After generating the reinforcement signal β by the environment, the corresponding probability vector of the LA for each cuckoo is modified based on the learning algorithm of automata using Algorithm 5.

Algorithm 5. *updateProbability* (automata LA, reinforcement signal β)

```
1. begin
2.     Z_i(n + 1) = Z_i(n) + β;
3.     η_i(n + 1) = η_i(n) + 1;
4.     d_i(n + 1) = Z_i(n+1) / η_i(n+1);
5.     P(n + 1) = P(n) + λ(e_M(k) − P(k));
6. end
```

In Algorithm 5, $Z_i(n)$ is the number of times that action α_i has been rewarded up to time n, $\eta_i(n)$ is the number of times that action α_i has been chosen up to time n, and $d_i(n+1)$ refers to the average reward value of the ith action. The pseudo-code for ACS is presented in Algorithm 6.

8. Experimental study

8.1. Numerical optimization problems

8.1.1. Benchmark functions

The performance of the proposed method is compared with other contestants on all 25 instances of the CEC 2005 (Suganthan *et al.* 2005). This test suite includes five unimodal functions (f_1–f_5), seven

Algorithm 6. ACS algorithm

1. **begin**
2. Objective function $f(x)$, $x = (x_1, x_2, \ldots, x_n)^T$;
3. Generate initial population of n host nests $x_i (i = 1, 2, \ldots, n)$;
4. Initialize a three-action automation with action set $\alpha = \{F_L, F_C, F_G\}$, and action probability vector $p = \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}$;
5. **while** ($e <$ *MaxEvals) or* (*Stopping criteria not met*) **do**
6. Get a cuckoo randomly (say, i) and generate a new solution using the distribution chosen by the learning automata according to Algorithm 4;
7. Evaluate the fitness of the new solution F_i;
8. Choose a nest among n (say, j) randomly;
9. **if** $F_i > F_j$ **then**
10. Replace j by the new solution;
11. **end-if**
12. Update the probability of choosing the actions according to Algorithm 5;
13. A fraction (p_a) of worst nests are abandoned and new ones are built;
14. Keep the best solutions (or nests with quality solutions);
15. Rank the solutions and find the current best;
16. **end-while**
17. **End**

multimodal functions (f_6–f_{12}), two expanded multimodal functions (f_{13} and f_{14}) and 11 hybrid composition functions (f_{15}–f_{25}). A complete description of the functions can be found in Suganthan *et al.* (2005).

8.1.2. Parameter settings for proposed modifications

Many experiments were carried out to examine the effect of different parameter settings on the performance of the proposed modifications. The following settings are adopted in the experiments of this article. Population size is a major factor in all nature-inspired algorithms. This parameter has a direct influence on the function evaluations (FEs). As the population size increases, the diversity of the population increases too. However, this can be at the expense of wasting precious FEs. In this study, the population size for all CS variants, except for CSCF, is set to 30 cuckoos. To make a fair comparison, the population size for CSCF is set to 10 cuckoos. The reason for this is that three eggs are generated for each cuckoo at each generation in CSCF, which results in $10 \times 3 = 30$ FEs. In each generation, about p_a per cent of worst candidate solutions are replaced with some new ones. This parameter introduces a mechanism for forgetting the worst candidate solutions in each generation. p_a varies from 0 to 1, where low values of p_a tend to make the algorithm keep a relatively long memory of the past, and high values of p_a result in forgetting much of the information about the past populations. In this study, p_a is set to 0.75 to increase the ability of the algorithm to escape from local optima.

For the LA in ACS, a learning rate of 0.003 is used. Table 1 summarizes the parameter settings used in this article.

Table 1. Parameter settings for different cuckoo search (CS) variants.

Parameter	Values
Population size for CSCF	10
Population size for other CS variants	30
p_a	0.75
Learning rate	0.003
Initial probability vector of LA	{1/3, 1/3, 1/3}

Note: CSCF = cuckoo search with composite flight; LA = learning automata.

8.1.3. Simulation settings and results

The following settings are adopted for all experiments on CEC 2005 benchmark functions. For a fair comparison among different methods, the maximum number of FEs was set to 300,000. All experiments on each function were run 30 times. All of the algorithms were implemented in MATLAB® release 2012 and executed on a computer with a Core i5 3210M processor and 6 GB of RAM, in the Ubuntu LTS 14.04 operating system.

The average and standard deviation of the function error values $f(\vec{x}_{best}) - f(\vec{x}_{opt})$ among 30 independent runs were recorded for each benchmark function, where $f(\vec{x}_{best})$ is the best solution found by the algorithm in a typical run and $f(\vec{x}_{opt})$ is the optimum value of the test function.

8.1.4. Comparison with standard cuckoo search

Table 2 presents the results obtained for different CS variants. In Table 2, the improvement rate (% Imp) between the proposed modifications and the standard CS for problem instance f_i is calculated as follows:

$$\%Imp_i = 100 \times \left(1 - \frac{e_{i,MCS}}{e_{i,\,CS}}\right) \tag{31}$$

where $e_{i,MCS}$ and $e_{i,CS}$ are the average function error value of the proposed modified CS algorithms and the average function error value of the standard CS for problem instance f_i, respectively.

Table 2. Mean and standard deviation of the function error values of standard cuckoo search (CS), the proposed cuckoo search with composite flight (CSCF) and the proposed adaptive cuckoo search (ACS) over 30 independent runs on 25 benchmark functions at 30D, after 300,000 function evaluations.

		CS	CSCF		ACS	
Property	F	Mean ± Std	Mean ± Std	% Imp	Mean ± Std	% Imp
Unimodal	f_1	3.09E − 02 ± 1.27E − 02	5.68E − 14 ± 1.49E − 14 $^+$	100.00	5.89E − 14 ± 1.09E − 14 $^+$	100.00
	f_2	1.54E + 03 ± 2.33E + 02	4.65E − 02 ± 5.42E − 02 $^+$	99.99	9.11E + 00 ± 6.99E + 00 $^+$	99.40
	f_3	3.81E + 06 ± 6.45E + 05	1.77E + 06 ± 4.96E + 05 $^+$	53.54	3.94E + 06 ± 9.04E + 05 $^\approx$	0.00*
	f_4	3.30E + 04 ± 5.63E + 03	1.23E + 03 ± 6.94E + 02 $^+$	96.27	1.34E + 04 ± 6.44E + 03 $^+$	59.39
	f_5	6.77E + 03 ± 4.71E + 02	2.77E + 03 ± 5.14E + 02 $^+$	59.08	2.67E + 03 ± 3.68E + 02 $^+$	60.56
Basic multimodal	f_6	1.29E + 03 ± 8.28E + 02	7.09E + 01 ± 3.95E + 01 $^+$	94.50	1.15E + 02 ± 6.37E + 01 $^+$	91.08
	f_7	2.04E + 01 ± 5.16E + 00	2.32E − 02 ± 1.48E − 02 $^+$	99.88	2.76E − 02 ± 1.34E − 02 $^+$	99.86
	f_8	2.09E + 01 ± 5.06E − 02	2.14E + 01 ± 8.09E − 02 $^-$	0.00	2.14E + 01 ± 8.16E − 02 $^-$	0.00
	f_9	9.36E + 01 ± 6.22E + 00	8.84E + 00 ± 1.73E + 00 $^+$	90.55	1.54E + 01 ± 2.50E + 00 $^+$	83.54
	f_{10}	5.48E + 02 ± 4.73E + 01	1.64E + 02 ± 2.24E + 01 $^+$	70.07	2.13E + 02 ± 2.29E + 01 $^+$	61.13
	f_{11}	2.37E + 01 ± 1.09E + 00	2.85E + 01 ± 1.31E + 00 $^-$	0.00	3.34E + 01 ± 1.53E + 00 $^-$	0.00
	f_{12}	3.70E + 04 ± 7.05E + 03	2.51E + 04 ± 8.19E + 03 $^+$	32.16	4.26E + 04 ± 8.33E + 03 $^-$	0.00
Expanded multimodal	f_{13}	9.99E + 00 ± 8.67E − 01	1.17E + 01 ± 2.24E + 00 $^-$	0.00	1.52E + 01 ± 1.37E + 00 $^-$	0.00
	f_{14}	1.33E + 01 ± 1.40E − 01	1.45E + 01 ± 2.13E − 01 $^-$	0.00	1.45E + 01 ± 1.39E − 01 $^-$	0.00
Hybrid composition	f_{15}	3.17E + 02 ± 6.20E + 01	3.23E + 02 ± 1.05E + 02 $^\approx$	0.00*	4.27E + 02 ± 5.75E + 01 $^-$	0.00
	f_{16}	4.08E + 02 ± 2.69E + 01	2.91E + 02 ± 1.00E + 02 $^+$	28.67	2.81E + 02 ± 4.69E + 01 $^+$	31.12
	f_{17}	4.67E + 02 ± 7.32E + 01	3.59E + 02 ± 1.08E + 02 $^+$	23.12	4.70E + 02 ± 7.11E + 01 $^\approx$	0.00*
	f_{18}	9.86E + 02 ± 1.44E + 01	9.09E + 02 ± 1.95E + 00 $^+$	7.80	9.05E + 02 ± 1.99E + 01 $^+$	8.21
	f_{19}	9.88E + 02 ± 1.88E + 01	9.09E + 02 ± 1.88E + 00 $^+$	7.99	9.10E + 02 ± 2.84E + 00 $^+$	7.89
	f_{20}	1.00E + 03 ± 3.39E + 01	9.08E + 02 ± 2.06E + 00 $^+$	9.20	9.08E + 02 ± 2.00E + 01 $^+$	9.20
	f_{21}	1.13E + 03 ± 1.22E + 02	5.11E + 02 ± 5.74E + 01 $^+$	54.77	5.03E + 02 ± 2.43E + 00 $^+$	55.48
	f_{22}	1.08E + 03 ± 3.05E + 01	9.09E + 02 ± 1.06E + 01 $^+$	15.83	9.35E + 02 ± 1.51E + 01 $^+$	13.42
	f_{23}	1.04E + 03 ± 1.27E + 02	5.35E + 02 ± 4.84E + 00 $^+$	48.55	5.38E + 02 ± 1.02E + 01 $^+$	48.26
	f_{24}	1.24E + 03 ± 3.47E + 01	2.00E + 02 ± 1.37E + 00 $^+$	83.87	2.00E + 02 ± 6.40E − 13 $^+$	83.87
	f_{25}	1.74E + 03 ± 1.24E + 01	1.62E + 03 ± 3.30E + 00 $^+$	6.89	1.71E + 03 ± 1.49E + 01 $^+$	1.72
		+	20		17	
		−	4		6	
		≈	1		2	

Note: '+' and '−' indicate a 0.05 level of significance by Friedman multiple comparison test. '+', '−' and '≈' denote that the performance of the corresponding modification is better than, worse than and similar to that of standard CS, respectively. The improvement rate (% Imp) values with an asterisk symbol are not statistically significant.

In situations where the results of the proposed modifications show deterioration in the performance of the CS improvement over the standard CS, the improvement rate of '0.00' is printed in the corresponding column.

Considering the results of Table 2, it is observed that CSCF is the best performing algorithm. The reason for this can be attributed to the memetic behaviour of the CSCF. The existence of three different probability distributions, *i.e.* Cauchy, Levy and Gaussian distributions, with different features is favourable to the performance of CSCF. From Table 2, it can also be concluded that both proposed modifications can significantly improve the performance of standard CS.

8.1.5. Comparison between cuckoo search with composite flight, with Gaussian distribution and with Cauchy distribution

In this experiment, the results of CSCF are compared with those of CS with Gaussian distribution (CS-Gauss) and CS with Cauchy distribution (CS-Cauchy). Table 3 summarizes the experimental results provided by different CS variants on 25 benchmark functions.

Considering Table 3, the overall performance of CSCF is better than that of CS-Gauss and CS-Cauchy. On unimodal functions, CSCF is obviously the best performing algorithm. It outperformed CS-Gauss and CS-Cauchy on all five unimodal functions. On basic multimodal functions, CSCF shows better performance than CS-Gauss and CS-Cauchy. It performed better than CS-Gauss and CS-Cauchy on five out of seven and four out of seven basic multimodal functions. CS-Cauchy is the best performing algorithm on expanded multimodal functions. Part of the reason for this can be

Table 3. Mean and standard deviation of the function error values of cuckoo search with Gaussian distribution (CS-Gauss), cuckoo search with Cauchy distribution (CS-Cauchy) and the proposed cuckoo search with composite flight (CSCF) over 30 independent runs on 25 benchmark functions at 30D, after 300,000 function evaluations.

		CS-Gauss	CS-Cauchy	CSCF
Property	F	Mean ± Std	Mean ± Std	Mean ± Std
Unimodal	f_1	6.76E − 12 ± 1.16E − 11 $^+$	2.21E − 13 ± 2.95E − 14 $^+$	5.68E − 14 ± 1.49E − 14
	f_2	3.59E + 03 ± 1.31E + 03 $^+$	7.60E + 02 ± 1.69E + 02 $^+$	4.65E − 02 ± 5.42E − 02
	f_3	6.52E + 07 ± 2.80E + 07 $^+$	4.89E + 06 ± 9.60E + 05 $^+$	1.77E + 06 ± 4.96E + 05
	f_4	7.76E + 03 ± 4.02E + 03 $^+$	1.21E + 04 ± 3.23E + 03 $^+$	1.23E + 03 ± 6.94E + 02
	f_5	2.97E + 03 ± 1.01E + 03 $^+$	3.99E + 03 ± 4.86E + 02 $^+$	2.77E + 03 ± 5.14E + 02
Basic multimodal	f_6	2.30E + 02 ± 2.41E + 02 $^+$	8.28E + 01 ± 4.37E + 01 $^\approx$	7.09E + 01 ± 3.95E + 01
	f_7	1.53E + 00 ± 1.26E + 00 $^+$	1.08E + 00 ± 1.67E − 01 $^+$	2.32E − 02 ± 1.48E − 02
	f_8	2.14E + 01 ± 7.38E − 02 $^\approx$	2.09E + 01 ± 3.78E − 02 $^-$	2.14E + 01 ± 8.09E − 02
	f_9	2.23E + 01 ± 5.33E + 00 $^+$	2.97E + 01 ± 3.10E + 00 $^+$	8.84E + 00 ± 1.73E + 00
	f_{10}	2.44E + 02 ± 2.28E + 01 $^+$	2.24E + 02 ± 1.66E + 01 $^+$	1.64E + 02 ± 2.24E + 01
	f_{11}	3.42E + 01 ± 9.33E + 00 $^+$	2.66E + 01 ± 8.24E − 01 $^-$	2.85E + 01 ± 1.31E + 00
	f_{12}	2.25E + 04 ± 1.03E + 04 $^\approx$	3.42E + 04 ± 4.60E + 03 $^+$	2.51E + 04 ± 8.19E + 03
Expanded multimodal	f_{13}	2.58E + 01 ± 2.63E + 00 $^+$	1.04E + 01 ± 4.76E − 01 $^-$	1.17E + 01 ± 2.24E + 00
	f_{14}	1.44E + 01 ± 1.60E − 01 $^\approx$	1.35E + 01 ± 1.03E − 01 $^-$	1.45E + 01 ± 2.13E − 01
Hybrid composition	f_{15}	3.88E + 02 ± 1.20E + 02 $^+$	3.20E + 02 ± 4.63E + 01 $^\approx$	3.23E + 02 ± 1.05E + 02
	f_{16}	3.28E + 02 ± 5.59E + 01 $^\approx$	2.30E + 02 ± 1.71+01 $^-$	2.91E + 02 ± 1.00E + 02
	f_{17}	4.35E + 02 ± 1.04E + 02 $^+$	2.94E + 02 ± 5.95E + 01 $^-$	3.59E + 02 ± 1.08E + 02
	f_{18}	9.06E + 02 ± 2.15E + 00 $^-$	8.82E + 03 ± 4.64E + 01 $^+$	9.09E + 02 ± 1.95E + 00
	f_{19}	9.06E + 02 ± 1.41E + 00 $^-$	8.80E + 03 ± 4.79E + 01 $^+$	9.09E + 02 ± 1.88E + 00
	f_{20}	9.07E + 02 ± 1.55E + 00 $^-$	8.88E + 02 ± 4.24E + 01 $^-$	9.08E + 02 ± 2.06E + 00
	f_{21}	5.10E + 02 ± 5.87E + 01 $^\approx$	5.01E + 02 ± 2.43E + 00 $^\approx$	5.11E + 02 ± 5.74E + 01
	f_{22}	9.33E + 02 ± 1.23E + 01 $^+$	9.16E + 02 ± 7.79E + 00 $^+$	9.09E + 02 ± 1.06E + 01
	f_{23}	5.47E + 02 ± 7.44E + 01 $^\approx$	5.34E + 02 ± 2.10E − 03 $^\approx$	5.35E + 02 ± 4.84E + 00
	f_{24}	2.00E + 02 ± 4.33E − 10 $^\approx$	2.00E + 02 ± 1.29E − 07 $^\approx$	2.00E + 02 ± 1.37E + 00
	f_{25}	1.68E + 03 ± 1.25E + 01 $^+$	1.66E + 03 ± 5.18E + 00 $^-$	1.62E + 03 ± 3.30E + 00
	+	15	12	
	−	3	8	
	≈	7	5	

Note: '+' and '−' indicate a 0.05 level of significance by Friedman multiple comparison test. '+', '−' and '≈' denote that the performance of CSCF is better than, worse than and similar to that of the contestant CS variant, respectively.

directly attributed to the nature of the problems. When the search space is large compared to the size of the population and the problems are multi-modal, it is expected that an algorithm with relatively large jumps will show better performance. Here, the population size of CS-Cauchy is three times larger than that of CSCF; and at each generation, one of the FEs is devoted to the flight/jump with Gaussian distribution, which is a relatively small jump. Finally, CS-Gauss, CS-Cauchy and CSCF showed almost the same performance on hybrid composition functions.

8.1.6. Comparison between cuckoo search with composite flight and modified cuckoo search

In this section, the performance of the proposed CSCF was compared with MCS[1] (Walton *et al.* 2011). Table 4 shows the experimental results obtained by CSCF and MCS.

To justify the reason for the superiority of CSCF over MCS, one should pay attention to the process of generating new eggs by the two algorithms. In MCS, in the early generations of the optimization process, a relatively high value of alpha causes the CS to have more exploration ability and search a vast area of the search space. As the optimization process progresses, the value of alpha is decreased, which causes MCS to tend towards more exploitation. Therefore, if MCS fails to locate the area of the optima in the early generations, it is unlikely that MCS can achieve an acceptable performance. Comparing CSCF to MCS, it is concluded that the composite flight operator in CSCF can establish a steady balance between the exploration and exploitation of the CS.

To further investigate the performance of CSCF, it is compared to MCS with respect to the execution time on functions f_1 and f_6, whose dimensionality D is set to {10, 30, 50, 100, 150}. Figure 2 shows the average computation times consumed by CSCF and MCS over 30 independent runs when solving f_1 and f_6 for different numbers of dimensions. As can be observed in Figure 2, the execution

Table 4. Mean and standard deviation of the function error values of modified cuckoo search (MCS) and the proposed cuckoo search with composite flight (CSCF) over 30 independent runs on 25 benchmark functions at 30D, after 300,000 function evaluations.

Property	F	MCS	S	CSCF
Unimodal	f_1	5.05E − 02 ± 4.70E − 02	+	5.68E − 14 ± 1.49E − 14
	f_2	1.42E + 03 ± 5.64E + 02	+	4.65E − 02 ± 5.42E − 02
	f_3	1.16E + 07 ± 3.21E + 06	+	1.77E + 06 ± 4.96E + 05
	f_4	2.47E + 04 ± 6.13E + 03	+	1.23E + 03 ± 6.94E + 02
	f_5	1.22E + 04 ± 2.79E + 03	+	2.77E + 03 ± 5.14E + 02
Basic multimodal	f_6	2.17E + 03 ± 3.01E + 03	+	7.09E + 01 ± 3.95E + 01
	f_7	8.02E + 03 ± 4.98E + 02	+	2.32E − 02 ± 1.48E − 02
	f_8	2.09E + 01 ± 5.67E − 02	−	2.14E + 01 ± 8.09E − 02
	f_9	4.50E + 01 ± 2.02E + 01	+	8.84E + 00 ± 1.73E + 00
	f_{10}	2.41E + 02 ± 5.22E + 01	+	1.64E + 02 ± 2.24E + 01
	f_{11}	3.00E + 01 ± 2.99E + 00	+	2.85E + 01 ± 1.31E + 00
	f_{12}	2.58E + 04 ± 1.39E + 04	≈	2.51E + 04 ± 8.19E + 03
Expanded multimodal	f_{13}	1.22E + 01 ± 5.12E + 00	+	1.17E + 01 ± 2.24E + 00
	f_{14}	1.20E + 01 ± 5.18E − 01	−	1.45E + 01 ± 2.13E − 01
Hybrid composition	f_{15}	3.19E + 02 ± 1.27E + 02	≈	3.23E + 02 ± 1.05E + 02
	f_{16}	3.98E + 02 ± 1.00E + 02	+	2.91E + 02 ± 1.00E + 02
	f_{17}	5.53E + 02 ± 1.38E + 02	+	3.59E + 02 ± 1.08E + 02
	f_{18}	1.08E + 03 ± 6.67E + 01	+	9.09E + 02 ± 1.95E + 00
	f_{19}	1.07E + 03 ± 5.30E + 01	+	9.09E + 02 ± 1.88E + 00
	f_{20}	1.09E + 03 ± 4.59E + 01	+	9.08E + 02 ± 2.06E + 00
	f_{21}	1.19E + 03 ± 1.91E + 02	+	5.11E + 02 ± 5.74E + 01
	f_{22}	1.19E + 03 ± 5.24E + 01	+	9.09E + 02 ± 1.06E + 01
	f_{23}	1.17E + 03 ± 1.66E + 02	+	5.35E + 02 ± 4.84E + 00
	f_{24}	1.31E + 03 ± 4.09E + 01	+	2.00E + 02 ± 1.37E + 00
	f_{25}	1.83E + 03 ± 2.21E + 01	+	1.62E + 03 ± 3.30E + 00
	+		21	
	−		2	
	≈		2	

Note: '+' and '−' indicate a 0.05 level of significance by Friedman multiple comparison test. '+', '−' and '≈' denote that the performance of CSCF is better than, worse than and similar to that of MCS, respectively.

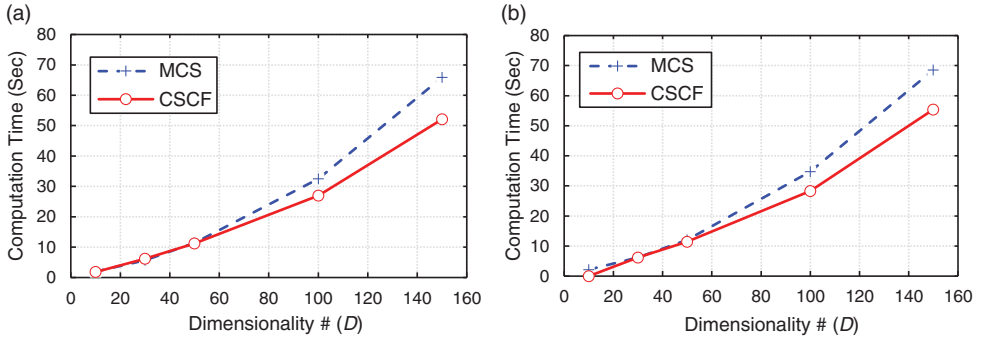

Figure 2. Computation time consumed by modified cuckoo search (MCS) and the proposed cuckoo search with composite flight (CSCF) for varying number of dimensions on benchmark functions f_1 and f_6.

time of CSCF is better than that of MCS. Regarding Figure 2(a) and 2(b), each line can be divided into two parts. When $10 \leq D \leq 50$, the computation times of MCS and CSCF are almost the same. However, when $50 \leq D \leq 150$, the computation time consumed by CSCF is less than that of MCS. It can be observed that the increasing rate of CSCF is significantly smaller than the increasing rate obtained by MCS. For this part, CSCF reduces the computation time by 13.75% on average compared with MCS.

8.2. Hard rock tunnel-boring machine rate of penetration

In this section, the SVR-CSCF is proposed and its effectiveness is studied on predicting TBM performance in hard rock conditions.

8.2.1. Problem description

Hard rock penetration rate is an important factor in the economic planning of many underground tunnelling projects. Two sets of features are used to predict the penetration rate of hard rock TBMs. The first set consists of parameters corresponding to intact and mass rock properties such as uniaxial compressive strength, brittleness index, Poisson ratio, porosity, rock quality designation, rock mass rating, orientation of discontinuities and distance between planes of weakness. The second set of parameters is the set of design parameters of TBM and operational parameters such as thrust, torque, rotational velocity and number of cutting blades (Hassanpour, Rostami, and Zhao 2011).

8.2.2. Data set

The data set used in this study obtained from Yagiz (2008). This data set was gathered from the New York City Water Tunnel No. 3, Stage 2, project in 1998. The project aimed at improving the freshwater distribution throughout the city of New York, USA. The tunnel is about 7.5 km long, has a diameter

Table 5. Description of the data set (153 data points).

Standard deviation	Maximum	Average	Minimum	Parameter (units)
22.09	199.70	149.88	118.30	UCS (MPa)
0.86	11.40	9.54	6.70	BTS (MPa)
8.40	58.00	35.00	25.00	BI (kN/mm)
0.64	2.00	1.02	0.05	DPW (m)
23.00	89.00	45.00	2.00	α (°)
0.40	3.07	2.05	1.27	Measured ROP (m/h)

Note: UCS = uniaxial compressive strength; BTS = Brazilian tensile strength; BI = brittleness index; DPW = distance between planes of weakness; α = angle of internal friction; ROP = rate of penetration.

of about 7 m, and was excavated beneath the boroughs of Brooklyn and Queens at an average depth of 200 m below sea level using power TBM (Merguerian and Ozdemir 2003). This data set is used to evaluate the performance of SVR-CSCF in comparison with other state-of-the-art methods in the literature. The data and parameters of the data set are listed in Table 5.

8.2.3. Performance criteria

Root mean squared error (RMSE) is used to measure the effectiveness of the proposed method in the prediction of TBM performance in hard rock conditions. RMSE is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\mathrm{predicted}} - y_{\mathrm{true}})^2} \tag{32}$$

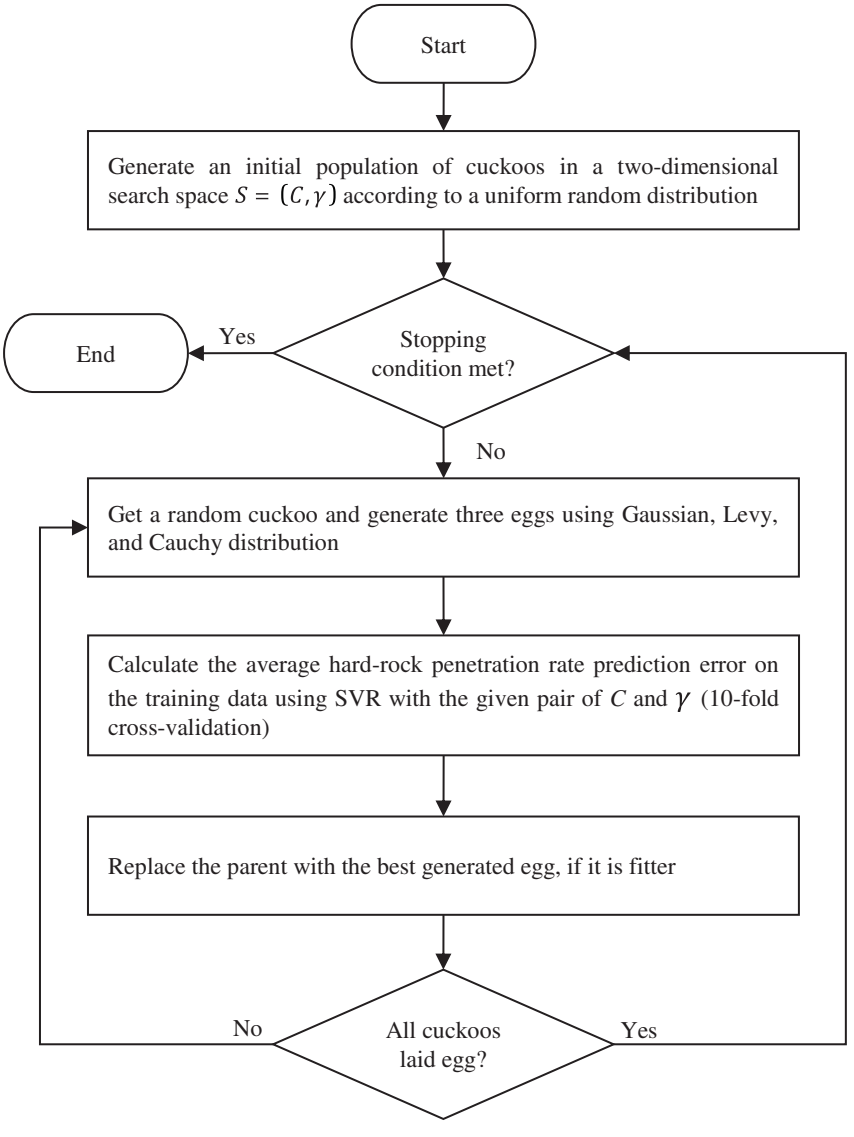

Figure 3. Schematic view of support vector regression–cuckoo search with composite flight (SVR-CSCF) for hard rock tunnel-boring machine penetration rate prediction.

where n is the number of data values, and $y_{\text{predicted}}$ and y_{true} are the predicted value and target value, respectively. This measure varies from 0 to ∞, with values closer to zero showing better prediction.

8.2.4. Applying cuckoo search with composite flight for adjusting the parameters of support vector regression

To find the best parameters of SVR for the given data set, CSCF is used as an optimization algorithm to determine the best combination of C and γ. Here, the objective function $f(x)$ is defined as the mean of RMSE over a 10-fold cross-validation process on the training data, which should be minimized. The general procedure of SVR-CSCF is given in Figure 3.

The maximum iteration for CSCF is set to 240 FEs and a population of 10 cuckoos is used. The parameter p_a is set to 0.75. Finally, 75% of the data was used as training set data, and the remaining 25% of the data was used as test set data.

8.2.5. Results and discussion

Table 6 summarizes the optimization results of SVR-CSCF, linear multivariate regression (Yagiz 2008), PSO (Yagiz and Karahan 2011) and nonlinear multivariate regression (Yagiz *et al.* 2009) on the hard rock TBM penetration rate prediction problem. The comparison of the predicted versus true values is also visualized in Figure 4. Regarding the reported results, it is concluded that SVR-CSCF gave the best results, which shows the applicability of the proposed method to real-world optimization problems.

Table 6. Optimization results of linear multivariate regression (LMR), particle swarm optimization (PSO), nonlinear multivariate regression (NLMR) and the proposed support vector regression–cuckoo search with composite flight (SVR-CSCF) for the hard rock tunnel-boring machine penetration rate prediction problem.

LMR		PSO		NLMR		SVR-CSCF	
RMSE	RMSE*	RMSE	RMSE*	RMSE	RMSE*	RMSE	RMSE*
0.2221	0.1950	0.2111	0.1942	1.4757	1.5086	0.0100	0.0100
0.2162	0.2136	0.2083	0.2026	1.5003	1.4361	0.0100	0.0100
0.2170	0.2112	0.2136	0.1861	1.4911	1.4636	0.0099	0.0100
0.2150	0.2171	0.2116	0.1925	1.5090	1.4092	0.0100	0.0100
0.2207	0.1996	0.2089	0.2009	1.4687	1.5285	0.0099	0.0098
0.2132	0.2222	0.2036	0.2163	1.4852	1.4811	0.0099	0.0100
0.2129	0.2230	0.2042	0.2146	1.4908	1.4647	0.0099	0.0098
0.2174	0.2099	0.2067	0.2075	1.4730	1.5164	0.0099	0.0100
0.2054	0.2428	0.1981	0.2308	1.4871	1.4756	0.0099	0.0098
0.2117	0.2265	0.2031	0.2177	1.4735	1.5149	0.0100	0.0100
0.2128	0.2233	0.2037	0.2160	1.4739	1.5139	0.0100	0.0100
0.2216	0.1966	0.2159	0.1779	1.5003	1.4359	0.0099	0.0101
0.2198	0.2027	0.2122	0.1905	1.4869	1.4761	0.0099	0.0100
0.2154	0.2161	0.2076	0.2047	1.4866	1.4771	0.0100	0.0100
0.2189	0.2056	0.2114	0.1932	1.4882	1.4722	0.0099	0.0100
0.2151	0.2169	0.2078	0.2043	1.4871	1.4756	0.0099	0.0100
0.2190	0.2050	0.2067	0.2075	1.4706	1.5230	0.0098	0.0098
0.2033	0.2478	0.1974	0.2324	1.4961	1.4487	0.0098	0.0100
0.2168	0.2118	0.2073	0.2058	1.4856	1.4800	0.0099	0.0100
0.2150	0.2171	0.2116	0.1925	1.5090	1.4092	0.0100	0.0100
0.2188	0.2057	0.2069	0.2070	1.4744	1.5123	0.0099	0.0099
0.2156	0.2154	0.2079	0.2040	1.4853	1.4809	0.0098	0.0100
0.2119	0.2260	0.2046	0.2134	1.4884	1.4716	0.0098	0.0101
0.2184	0.2070	0.2068	0.2072	1.4857	1.4797	0.0098	0.0099
0.2033	0.2478	0.1974	0.2324	1.4961	1.4487	0.0097	0.0100
0.2191	0.2048	0.2065	0.2079	1.4601	1.5523	0.0098	0.0100
0.2128	0.2233	0.2037	0.2160	1.4739	1.5139	0.0100	0.0100
0.2054	0.2428	0.2012	0.2227	1.5025	1.4291	0.0099	0.0100
0.2168	0.2118	0.2073	0.2058	1.4856	1.4800	0.0100	0.0099
0.2182	0.2077	0.2133	0.1869	1.4973	1.4450	0.0098	0.0100

Note: The columns 'RMSE' and 'RMSE*' denote the root mean squared error on the training and test sets, respectively.

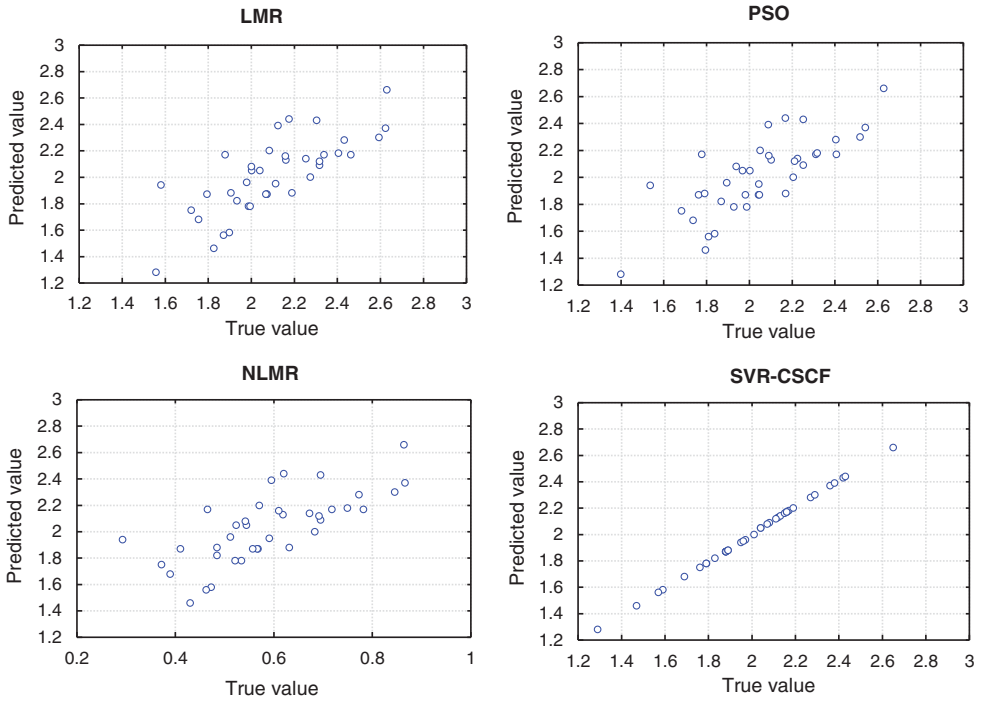

Figure 4. True values versus predicted values of linear multivariate regression (LMR), particle swarm optimization (PSO), nonlinear multivariate regression (NLMR), and the proposed support vector regression–cuckoo search with composite flight (SVR-CSCF) for the hard rock tunnel-boring machine rate of penetration problem.

9. Conclusion

Two modifications to the standard CS have been presented in this article. Both proposed modifications aimed at improving the CS by changing the distribution of steps when generating new eggs. The first extension, *i.e.* CSCF, is a memetic algorithm which adopts a new flight operator. The second extension, *i.e.* ACS, employs the concept of LA with a pursuit algorithm into CS to govern the suitable distribution of step for each cuckoo based on its progress in the search space.

The experimental results in Section 8 clearly demonstrated the superiority of the proposed methods. Comparisons were made with other well-known CS variants in terms of solution accuracy and computation time. The results showed the effectiveness of the proposed methods.

To examine the applicability of CSCF on practical problems, a new model based on SVR and CSCF, *i.e.* SVR-CSCF, was further developed in which CSCF was used to determine the optimal value for the parameters of SVR (*i.e.* C and γ). SVR-CSCF was applied to solve a TBM penetration rate prediction problem in hard rock conditions. The obtained results proved the great performance of SVR-CSCF in terms of accuracy and efficiency.

Future research may focus on incorporating different adaptation mechanisms with CSCF. It could also be interesting to use other learning schemes for choosing the proper distribution in ACS.

Finally, the source code of the proposed algorithms is openly available to researchers and can be requested from the corresponding author for future research.

Note

1. The source code for MCS is publicly available at: https://code.google.com/archive/p/modified-cs/

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Akbari Torkestani, J. 2012. "An Adaptive Learning Automata-Based Ranking Function Discovery Algorithm." *Journal of Intelligent Information Systems* 39 (2): 441–459. doi:10.1007/s10844-012-0197-4.

Akbari Torkestani, J., and M. R. Meybodi. 2011. "A Cellular Learning Automata-Based Algorithm for Solving the Vertex Coloring Problem." *Expert Systems with Applications* 38 (8): 9237–9247. doi:10.1016/j.eswa.2011.01.098.

Barzegar, S., M. Davoudpour, M. R. Meybodi, A. Sadeghian, and M. Tirandazian. 2011. "Formalized Learning Automata with Adaptive Fuzzy Coloured Petri Net; An Application Specific to Managing Traffic Signals." *Scientia Iranica* 18 (3): 554–565. doi:10.1016/j.scient.2011.04.007.

Bhargava, V., S. E. K. Fateen, and A. Bonilla-Petriciolet. 2013. "Cuckoo Search: A New Nature-Inspired Optimization Method for Phase Equilibrium Calculations." *Fluid Phase Equilibria* 337: 191–200. doi:10.1016/j.fluid.2012.09.018.

Chandrasekaran, K., and S. P. Simon. 2012. "Multi-objective Scheduling Problem: Hybrid Approach Using Fuzzy Assisted Cuckoo Search Algorithm." *Swarm and Evolutionary Computation* 5: 1–16. doi:10.1016/j.swevo.2012.01.001.

Civicioglu, P., and E. Besdok. 2013. "A Conceptual Comparison of the Cuckoo-Search, Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony Algorithms." *Artificial Intelligence Review* 39 (4): 315–346. doi:10.1007/s10462-011-9276-0.

Esnaashari, M., and M. R. Meybodi. 2011. "A Cellular Learning Automata-Based Deployment Strategy for Mobile Wireless Sensor Networks." *Journal of Parallel and Distributed Computing* 71 (7): 988–1001. doi:10.1016/j.jpdc.2010.10.015.

Hashemi, A. B., and M. R. Meybodi. 2011. "A Note on the Learning Automata Based Algorithms for Adaptive Parameter Selection in PSO." *Applied Soft Computing* 11 (1): 689–705. doi:10.1016/j.asoc.2009.12.030.

Hassanpour, J., J. Rostami, and J. Zhao. 2011. "A New Hard Rock TBM Performance Prediction Model for Project Planning." *Tunnelling and Underground Space Technology* 26 (5): 595–603. doi:10.1016/j.tust.2011.04.004.

Kordestani, J. K., A. Ahmadi, and M. R. Meybodi. 2014. "An Improved Differential Evolution Algorithm using Learning Automata and Population Topologies." *Applied Intelligence* 41 (4): 1150–1169.

Lee, Chang-Yong and Yao, Xin. 2004. "Evolutionary Programming using Mutations Based on the Levy Probability Distribution." *IEEE Transactions on Evolutionary Computation* 8 (1): 1–13. doi:10.1109/TEVC.2003.816583.

Merguerian, C., and L. Ozdemir. 2003. "Rock Mass Properties and Hard Rock TBM Penetration Rate Investigations, Queens Tunnel Complex, NYC Water Tunnel# 3, Stage 2." *Proceedings of Rapid Excavation and Tunneling Conferences* (pp. 1019–1036).

Moghiss, V., M. R. Meybodi, and M. Esnaashari. 2010. "An Intelligent Protocol to Channel Assignment in Wireless Sensor Networks: Learning Automata Approach." *International Conference on Information Networking and Automation* (Vol. 1, pp. V1–338–V1–343).

Narendra, K. S., and M. A. Thathachar. 1974. "Learning Automata—A Survey." *IEEE Transactions on Systems, Man and Cybernetics* 4: 323–334. doi:0.1109/TSMC.1974.5408453.

Nguyen, Thang Trung, Dieu Ngoc Vo, and Tam Thanh Dao. 2014. "Cuckoo Search Algorithm using Different Distributions for Short-Term Hydrothermal Scheduling with Cascaded Hydropower Plants (pp. 1–6)." Presented at the TENCON 2014 - 2014 IEEE Region 10 Conference. doi:10.1109/TENCON.2014.7022454.

Piechocki, J., D. Ambroziak, A. Palkowski, and G. Redlarski. 2014. "Use of Modified Cuckoo Search Algorithm in the Design Process of Integrated Power Systems for Modern and Energy Self-Sufficient Farms." *Applied Energy* 114 (0): 901–908. doi:10.1016/j.apenergy.2013.07.057.

Rahnamayan, S., H. R. Tizhoosh, and M. M. A. Salama. 2007. "A Novel Population Initialization Method for Accelerating Evolutionary Algorithms." *Computers & Mathematics with Applications* 53 (10): 1605–1614. doi:10.1016/j.camwa.2006.07.013.

Rezvanian, A., and M. Meybodi. 2010. "LACAIS: Learning Automata Based Cooperative Artificial Immune System for Function Optimization." In *Contemporary Computing, Communications in Computer and Information Science*, edited by S. Ranka, A. Banerjee, K. Biswas, S. Dua, P. Mishra, R. Moona, S.-H. Poon, et al., Vol. 94, 64–75. Berlin: Springer. doi:10.1007/978-3-642-14834-7_7.

Shatnawi, M., and M. F. Nasrudin. 2011. "Starting Configuration of Cuckoo Search Algorithm using Centroidal Voronoi Tessellations." *International Conference on Hybrid Intelligent Systems* (pp. 40–45). Presented at the HIS 2011, Melacca, Malaysia: IEEE. doi:10.1109/HIS.2011.6122077.

Suganthan, P. N., N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. 2005. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-parameter Optimization* (No. #2005005). Nanyang Technol. Univ., Singapore, IIT Kanpur, Kanpur, India, #2005005.

Thathachar, M. A. L., and P. S. Sastry. 2002. "Varieties of Learning Automata: An Overview." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 32 (6): 711–722. doi:10.1109/TSMCB.2002.1049606.

Thathachar, M. A. L., and P. S. Sastry. 2003. *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. 1st ed. New York: Springer.

Vapnik, V., S. E. Golowich, and A. Smola. 1996. "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing." *Advances in Neural Information Processing Systems* 9: 281–287.

Walton, S., O. Hassan, K. Morgan, and M. R. Brown. 2011. "Modified Cuckoo Search: A New Gradient Free Optimisation Algorithm." *Chaos, Solitons & Fractals* 44 (9): 710–718. doi:10.1016/j.chaos.2011.06.004.

Wang, Y., Z. Cai, and Q. Zhang. 2011. "Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters." *IEEE Transactions on Evolutionary Computation* 15 (1): 55–66. doi:10.1109/TEVC.2010.2087271.

Wang, G., L. Guo, H. Duan, L. Liu, H. Wang, and B. Wang. 2012. "A Hybrid Meta-Heuristic DE/CS Algorithm for UCAV Path Planning." *Journal of Information and Computational Science* 9 (16): 4811–4818.

Wang, F., X. S. He, Y. Wang, and S. M. Yang. 2012. "Markov Model and Convergence Analysis Based on Cuckoo Search Algorithm." *Computer Engineering* 38 (11): 180–185.

Xu, X., Z. Ji, F. Yuan, and X. Liu. 2014. "A Novel Parallel Approach of Cuckoo Search using MapReduce." *2014 International Conference on Computer, Communications and Information Technology (CCIT 2014)*. Atlantis Press.

Xuegong, Z. 2000. "Introduction to Statistical Learning Theory and Support Vector Machines." *Acta Automatica Sinica* 26 (1): 32–42.

Yagiz, S. 2008. "Utilizing Rock Mass Properties for Predicting TBM Performance in Hard Rock Condition." *Tunnelling and Underground Space Technology* 23 (3): 326–339. doi:10.1016/j.tust.2007.04.011.

Yagiz, S., C. Gokceoglu, E. Sezer, and S. Iplikci. 2009. "Application of Two Non-linear Prediction Tools to the Estimation of Tunnel Boring Machine Performance." *Engineering Applications of Artificial Intelligence* 22 (4): 808–814.

Yagiz, S., and H. Karahan. 2011. "Prediction of Hard Rock TBM Penetration Rate using Particle Swarm Optimization." *International Journal of Rock Mechanics and Mining Sciences* 48 (3): 427–433.

Yang, Xin-She, and S. Deb. 2009. "Cuckoo Search via Lévy Flights." *World Congress on Nature & Biologically Inspired Computing. NaBIC 2009* (pp. 210–214). IEEE. doi:10.1109/NABIC.2009.5393690.

Yang, X.-S., and S. Deb. 2013. "Multiobjective Cuckoo Search for Design Optimization." *Emergent Nature Inspired Algorithms for Multi-objective Optimization* 40 (6): 1616–1624. doi:10.1016/j.cor.2011.09.026.

Yildiz, A. 2013. "Cuckoo Search Algorithm for the Selection of Optimal Machining Parameters in Milling Operations." *The International Journal of Advanced Manufacturing Technology* 64 (1–4): 55–61. doi:10.1007/s00170-012-4013-7.

Zheng, H., and Y. Zhou. 2012. "A Novel Cuckoo Search Optimization Algorithm Based on Gauss Distribution." *Journal of Computational Information Systems* 8 (10): 4193–4200.