Cuckoo Search with Composite Flight Operator for Numerical Optimization Problems and Its Application in Tunneling

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

[a] *Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran*

[b] *Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran Iran*

abedi.hossein@aut.ac.ir, Javidan.kazemi@gmail.com*, mmeybodi@aut.ac.ir

Abstract

This paper presents two modifications of the cuckoo search (CS) algorithm for numerical optimization problems. The first modified algorithm is CS 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 composite flight operator is to allow generating a new cuckoo egg 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 tunnel-boring machine. The experimental results show that the proposed modifications can significantly improve the performance of the standard CS.

Keywords: Cuckoo search, Global numerical optimization, Composite flight, Support vector regression, Tunnel-boring machine

1. Introduction

Nature-inspired algorithms have gained a 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; Xin-She Yang & Deb, 2009) which imitates the parasitism behavior of cuckoo bird. Theoretical studies have shown that CS is an efficient algorithm that can guarantee the global convergence (F. Wang, He, Wang, & Yang, 2012). A study by Civicioglu and Besdok (Civicioglu & Besdok, 2013) revealed that CS can perform better than particle swarm optimization (PSO) and artificial bee colony in solving numerical optimization problems. However, we believe that there is still a room for improving the performance of CS.

In this paper, two modifications to the CS are suggested. The first suggested modification is a CS with composite flight (CSCF). The primary objective of the composite flight is to generate a new egg through competing 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 for generating a new individual has already achieved good results in composite differential evolution (CoDE) (Y. Wang, Cai, & Zhang, 2011). However, the composition in the present study has a significant difference with 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 setting from the parameter candidate pool, but CSCF utilizes three types of random walk to compose. The second modification is an adaptive CS (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 tunnel-boring machine (TBM) in hard rock condition.

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

2. Cuckoo Search

CS is a stochastic population-based algorithm which was recently proposed by Yang and Deb (2009), motivated by the breeding strategy of some cuckoos. These birds show an extraordinary behavior 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 of their chicks. This behavior is the main inspiration of the CS algorithm.

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

- Each cuckoo only lays one egg at a time and dumps it in a randomly chosen nest.

- The best nests with high quality of 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 simply 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 $f(x),\ x = (x_1, x_2, \dots, x_n)^T$;
3.    Generate initial population of $n$ host nests $x_i (i = 1,2, \dots, n)$;
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, & Bonilla-Petriciolet, 2013), design of power systems for energy self-sufficient farms (Piechocki, Ambroziak, Palkowski, & Redlarski, 2014), selection of optimal machining parameters in milling operations (Yildiz, 2013), multi-objective optimization problems (Chandrasekaran & Simon, 2012; Yang & Deb, 2013), etc.

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 (F. Wang et al., 2012):

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

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 Levy distribution. Figure 1 shows the importance of β parameter in above equation.

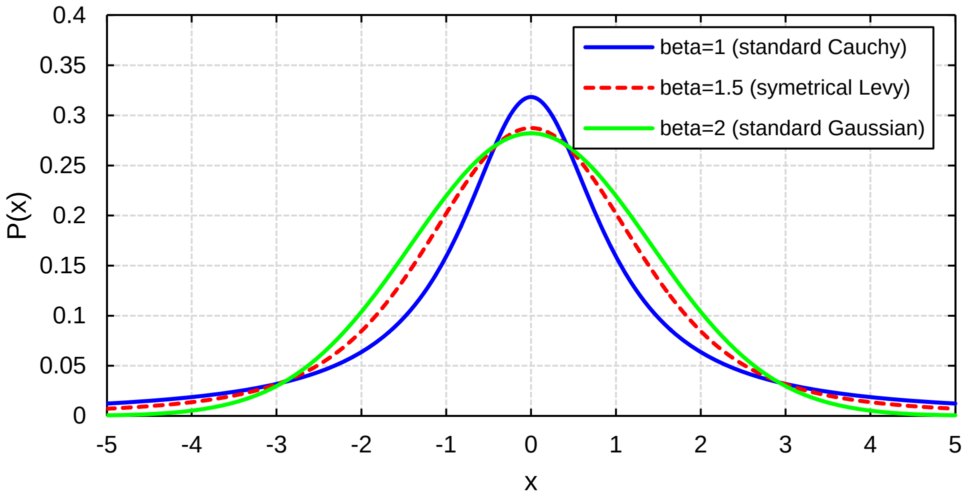

Figure 1. Levy probability distribution for different values of β.

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

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

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

where $\frac{1}{\pi(1+x^2)}$ is the famous probability density function of Cauchy distribution. $l(x;\beta)$ and its cumulative probability distribution $Pr_{l(x;\beta)}$ have the following limit properties (Chang-Yong Lee & Xin Yao, 2004):

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

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

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.

In order 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:

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

This function enables us to further analyze 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 $x = 10000$, 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 $\frac{Pr_{\beta=1}\{X>10000\}}{Pr_{\beta=2}\{X>10000\}} = 10000$. The same analysis could be done on the left tail function with similar results.

The above analysis show 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 & Thathachar, 1974; Thathachar & 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 toward selecting the optimal action.

To just name a few, LA have a wide variety of applications in parameter control (Rezvanian & Meybodi, 2010; Hashemi & Meybodi, 2011; Kazemi Kordestani, Ahmadi, & Meybodi, 2014), wireless sensor networks (Moghis, Meybodi, & Esnaashari, 2010; Esnaashari & Meybodi, 2011), vertex coloring problem (Akbari Torkestani & Meybodi, 2011), traffic signal control (Barzegar, Davoudpour, Meybodi, Sadeghian, & Tirandazian, 2011), information retrieval (Akbari Torkestani, 2012), etc.

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 $A = \{\alpha_1, \dots, \alpha_n\}$ is the set of finite actions, $P(k) = [p_1(k), \dots, p_n(k)]$ denotes the vector of action probabilities at time step k, which its elements should satisfy the following conditions:

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

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

B is the set of all possible outputs received from the environment. Finally, T is 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 paper, a learning algorithm called *pursuit algorithm* (Thathachar & Sastry, 2003) was applied that uses the histories of actions to update $P(k)$.

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

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

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 where its position is determined as follows:

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

One of the interesting aspects 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 modeled using B. In the P-model, $B = \{0,1\}$ in which the value “0” corresponds to unfavorable response or punishment and “1” corresponds to favorable response or reward. In the Q-model, $B = \{b_1, \dots, b_q\}, b_i \in [0,1]\ and\ q < \infty$. Finally, in the S-model, $B = [0,1]$ (Thathachar & Sastry, 2003).The framework for the learning process of a FALA with pursuit learning scheme in a stationary environment is shown in Algorithm 2.

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

```
1.  begin
2.      Let $A = \{\alpha_1, \dots, \alpha_n\}$ be the set of finite actions, where $n$ is the number of actions;
3.      Let $P = [p_1, \dots, p_n]$ 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 Eq. (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, Vapnic et al. (1997) proposed a modified version of the support vector machine for function approximation called support vector regression. Given a dataset of N training points as $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \aleph \times \Re$, where $\aleph$ denotes the space of the input data, SVR attempt 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:

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

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 function given in Eq. (16) could be expressed as:

$$minimize\ \frac{1}{2}\|\omega\|^2 \qquad (17)$$

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

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

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

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

where C is called 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 value of data point x_i deviate from the target value. In order to solve the constrained optimization problem in Eq. (19) the Lagrangian of the original problem is formed in the following manner:

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

By solving the above equation, we have:

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

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

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

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 function have been frequently used in the literature. In this paper, radial basis function was used which is defined as follows:

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

where γ is the width of radial basis function.

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

6. Related Works

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

6.1. Changing the Initialization Pattern of Cuckoo Search

It has been confirmed that the initialization method in the population-based algorithms has a great influence on the exploration behavior of the algorithms on the early iterations of the execution (Rahnamayan, Tizhoosh, & Salama, 2007). In the field of CS, Shatnawi and Nasrudin (2011) used centroidal voronoi tessellations for generating 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 exist that have tried to combine the desirable features of CS with other optimization methods. For example, Wang et al. (2012) proposed a hybrid algorithm of differential evolution and CS, which they have called DE/CS. In DE/CS, the mutation and crossover of DE is 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 exploitation ability of CS. Their experimental results indicate that DE/CS could outperform standard CS over uninhabited combat air vehicle path planning problem.

6.3. Parallel Search

With the aim to speed 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 have divided the search space into equally-sized cells. The search process of CS in the cells is followed simultaneously. Their results shows 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 CS, called 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 been 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*) a fraction of eggs with the best fitness is marked as top eggs, (*b*) 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 ones, can improve the performance of CS.

6.6. Changing the Distribution of the Steps

Standard CS algorithm uses Levy distribution for generating new solutions. While Levy distribution can increase the exploration ability of CS, the exploitation capacity of this algorithm remains poor. Therefore, a group of 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 Gauss distribution. In this method, the Levy flight was replaced with Gaussian distribution as follows:

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

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

Nguyen et al. (2014) applied CS with three different distribution including, Levy, Cauchy, and Gaussian for solving short-term hydrothermal scheduling problem considering cascaded hydropower plants.

7. Proposed Modified CS Algorithms

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

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

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 i^th dimension of the search space, that is, $\forall\, i \in \{1, \dots, D\}, -\infty < l_i < u_i < \infty$.

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 in which improves the global search and the latter 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 $\vec{X}_i = (x_{i1}, x_{i2}, \dots, x_{id})$. The initial population of eggs is simply randomized into the boundary of the search space according to a uniform distribution as follows:

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

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

(2) The fitness of the generated eggs is evaluated.

(3) The most fitted 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 $f(x),\ x = (x_1, x_2, \dots, x_n)^T$;
3.    Generate initial population of $n$ host nests $x_i (i = 1,2,\dots,n)$;
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 $max\{F_{Levy}, F_{Cauchy}, F_{Gauss}\} > F_j$ 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
```

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 distribution of step for each cuckoo is chosen based on its search progress, adaptively.

In ACS, a 3-action learning automaton (LA_{fly}) is associated with each cuckoo of the population which is responsible for choosing the proper distribution for fly operation, $|LA_{fly}|=|popsize|=30$. The actions of LA_{fly} correspond to *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 the 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 $j \epsilon [1, 2, 3]$ of LA do
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 age was 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 action were right or wrong, and update their probability vector accordingly. In this work, the reinforcement signal is generated as follows:

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

In above equation, $f(\vec{X}_{i,G})$ is the fitness value of the i^{th} cuckoo at G^{th} generation, and $f(\vec{X}_{i,G+1})$ is the fitness value of the generated egg by i^{th} cuckoo after evolving by selected distribution at generation $G+1$.

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

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

```
1. begin
2.     $Z_i(n+1) = Z_i(n) + \beta$;
3.     $\eta_i(n+1) = \eta_i(n) + 1$;
4.     $d_i(n+1) = \frac{Z_i(n+1)}{\eta_i(n+1)}$;
5.     $P(n+1) = P(n) + \lambda\left(e_{M(k)} - P(k)\right)$;
6. end
```

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

Algorithm 6. ACS Algorithm

```
1.  begin
2.      Objective function $f(x)$,  $x = (x_1, x_2, \dots, x_n)^T$;
3.      Generate initial population of n host nests $x_i (i = 1,2,\dots,n)$;
4.      Initialize a 3-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
```

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 suit include five unimodal functions (f_1-f_5), five multimodal functions (f_6-f_{12}), two expanded multimodal functions (f_{13}-f_{14}), and ten hybrid composition functions (f_{15}-f_{25}). The complete description of the functions can be found in (Suganthan et al., 2005).

8.1.2. Parameters settings for proposed modifications

Many experiments were carried out to examine the effect of different parameter settings on the performance of proposed modifications. The following settings are adopted in the experiments of this paper. 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. In order to have a fair comparison, the population size for CSCF is set to 10 cuckoos. The reason is that three eggs are generated for each cuckoo at each generation in CSCF, which results in 10×3=30 FEs. In each generation, about p_a percent 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 tends to make the algorithm to keep a relatively long memory of the past, and high values of p_a results 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 the local optima.

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

Table 1. Parameter settings for different 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}

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 300000. All experiments on each function were run 30 times. All the algorithms were implemented in Matlab 2012 and executed on a computer with Core i5 3210M Processor and 6GB of RAM in 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 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 obtained results for different CS variants. In Table 2, the improvement rate (%Imp) between proposed modifications and 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.

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 the “0.00” is printed in the corresponding column.

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 twenty five benchmark functions at 30D, after 300000 FEs.

Property	F	CS	CSCF		ACS	
		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	

‘‘+’’ 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 % Imp values with asterisk symbol are not statically significant.

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

8.1.5. Comparison between CSCF, CS with Gaussian distribution, and CS 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.

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 twenty five benchmark functions at 30D, after 300000 FEs.

Property	F	CS-Gauss	CS-Cauchy	CSCF
		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	

‘‘+’’ 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.

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 function. On basic multimodal functions, CSCF shows a better performance compared to CS-Gauss and CS-Cauchy. It could perform better than CS-Gauss and CS-Cauchy on 5 out of 7 and 4 out of 7 basic multimodal functions. CS-Cauchy is the best-performing algorithm on expanded multimodal functions. Part of the reason can be 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 larger jumps shows a better performance. Here, (*a*) the population size of the CS-Cauchy is three times larger than that of CSCF, and (*b*) 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 CSCF and MCS

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

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 twenty five benchmark functions at 30D, after 300000 FEs.

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

[1] Source code for MCS is publicly available at: https://code.google.com/archive/p/modified-cs/

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	

‘‘+’’ 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.

To justify the reason for the superiority of CSCFA over MCS, one should pay attention to the process of generating new eggs by the two algorithms. In MCS, at 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 the MCS to tend toward more exploitation. Therefore, if MCS fails to locate the area of the optima at the early generations, it is unlikely that MCS can reach to an acceptable performance. Comparing CSCF to MCS, it is concluded that composite flight operator in CSCF can establish a steady balance between the exploration and exploitation of the CS.

In order 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 consumed by CSCF and MCS over 30 independent runs when solving f_1 and f_6 for different number of dimension. As can be observed in Figure 2, the execution time of CSCF is better than that of MCS. Regarding Figures 2a and 2b, each line can be divided into two parts. When $10 \leq D \leq 50$, the computation time of MCS and CSCF are almost the same. However, when $50 \leq D \leq 150$, 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 saves the computation time by 13.75% on average compared with MCS.

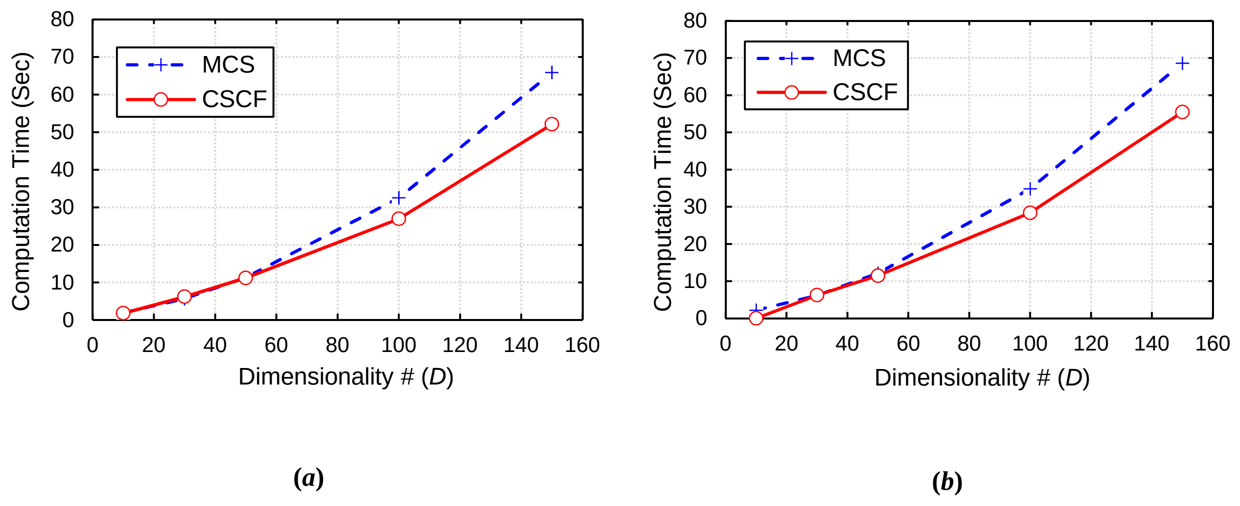

Figure 2. Computation time consumed by modified cuckoo search (MCS) and the proposed cuckoo search with composite flight (CSCF) for varying number of dimensions on benchmark functions f_1 and f_6.

8.2. Hard Rock Tunnel-Boring Machine Rate of Penetration

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

8.2.1. Problem description

Hard rock penetration rate is an important factor in economical planning of many underground tunneling 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 ones such as thrust, torque, rotational velocity and number of cutting blades (Hassanpour, Rostami, & Zhao, 2011).

8.2.2. Dataset

The dataset used in this study obtained from (Yagiz, 2008). This dataset was gathered from the Queens Water Tunnel # 3, Stage 2, project in 1998. The project aimed at improving the fresh water distribution throughout the City of New York, USA. The tunnel is about 7.5 km long, has a diameter about 7 meters, and was excavated beneath the Brooklyn and Queens at an average depth of 200 meters below the sea level using power TBM (Merguerian & Ozdemir, 2003). This dataset is used to evaluate the performance of SVR-CSCF in comparison with other state-of-the-art methods in the literature. The description of the data and parameters of the dataset is listed in Table 5.

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

standard deviation	maximum	average	minimum	parameter (unit)
22.09	199.70	149.88	118.30	$USC(M\ P_a)$
0.86	11.40	9.54	6.70	$BTS(M\ P_a)$
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	$\alpha(°)$
0.40	3.07	2.05	1.27	$Measured\ ROP(m/h)$

8.2.3. Performance criteria

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

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

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

8.2.4. Applying CSCF for adjusting the parameters of SVR

In order to find the best parameters of SVR for the given dataset, 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.

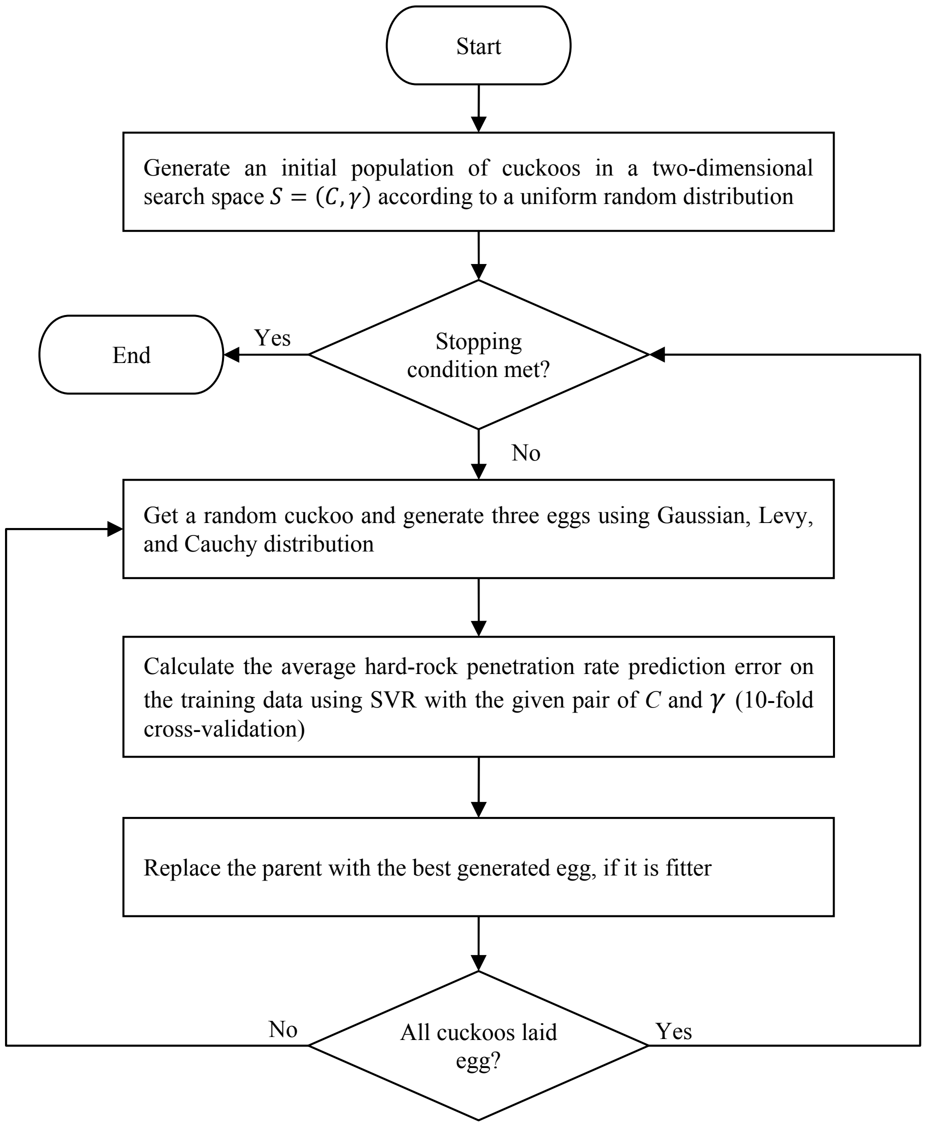
Start
Generate an initial population of cuckoos in a two-dimensional search space $S = (C, \gamma)$ according to a uniform random distribution
Stopping condition met?
Yes
End
No
Get a random cuckoo and generate three eggs using Gaussian, Levy, and Cauchy distribution
Calculate the average hard-rock penetration rate prediction error on the training data using SVR with the given pair of C and γ (10-fold cross-validation)
Replace the parent with the best generated egg, if it is fitter
All cuckoos laid egg?
No
Yes

Figure 3. Schematic view of SVR-CSCF for hard-rock TBM penetration rate prediction.

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, the 75% of the data was used as train set data, and remaining 25% 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 & Karahan, 2011), and non-linear multivariate regression (Yagiz, Gokceoglu, Sezer, & Iplikci, 2009) on the hard-rock TBM penetration rate prediction problem. The comparison of the predicted vs. true values is also visualized in Figure 4. Regarding the reported results, it is concluded that SVR-CSCF can provide the best results which shows the applicability of the proposed method for real-world optimization problems.

Table 6. Optimization results of linear multivariate regression (LMR), particle swarm optimization (PSO), non-linear multivariate regression (NLMR), and the proposed SVR-CSCF for hard-rock TBM 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

The columns "*RMSE*" and "*RMSE*[*]" denote the root mean squared error on the train and test sets, respectively.

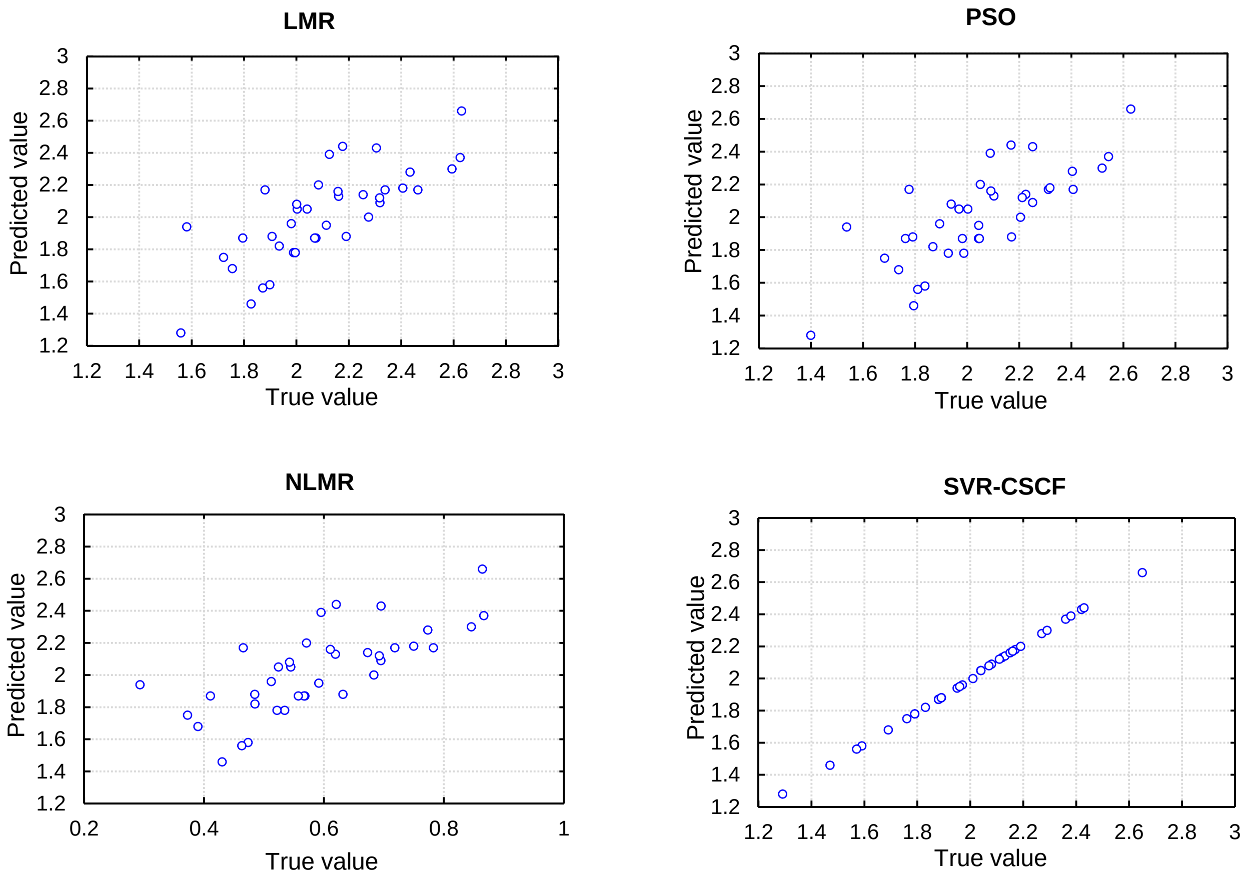

Figure 4. True values versus predicted values of linear multivariate regression (LMR), particle swarm optimization (PSO), non-linear multivariate regression (NLMR), and the proposed SVR-CSCF for hard rock TBM rate of penetration problem.

9. Conclusion

Two modifications to the standard CS have been presented in this paper. Both proposed modifications aimed at improving the CS by changing the distribution of steps when generating new eggs. First extension, i.e. CSCF, is a memetic algorithm which adopts a new flight operator. Second extension, i.e. ACS, employs the concept of LA with 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. The 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 TBM penetration rate prediction problem in hard-rock condition. The obtained results proved the great performance of SVR-CSCF in terms of accuracy and efficiency.

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

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

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., & Meybodi, M. R. (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., Davoudpour, M., Meybodi, M. R., Sadeghian, A., & Tirandazian, M. (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., Fateen, S. E. K., & Bonilla-Petriciolet, A. (2013). Cuckoo Search: A new nature-inspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria*, *337*(0), 191–200. doi:10.1016/j.fluid.2012.09.018

Chandrasekaran, K., & Simon, S. P. (2012). Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm. *Swarm and Evolutionary Computation*, *5*(0), 1–16. doi:10.1016/j.swevo.2012.01.001

Chang-Yong Lee, & Xin Yao. (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

Civicioglu, P., & Besdok, E. (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., & Meybodi, M. R. (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., & Meybodi, M. R. (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., Rostami, J., & Zhao, J. (2011). A new hard rock TBM performance prediction model for project planning. *Tunnelling and Underground Space Technology*, *26*(5), 595 – 603. doi:http://dx.doi.org/10.1016/j.tust.2011.04.004

Kordestani, J. K., Ahmadi, A., & Meybodi, M. R. (2014). An improved Differential Evolution algorithm using learning automata and population topologies. *Applied Intelligence*, *41*(4), 1150–1169.

Merguerian, C., & Ozdemir, L. (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., Meybodi, M. R., & Esnaashari, M. (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., & Thathachar, M. A. (1974). Learning automata-a survey. *IEEE Transactions on Systems, Man and Cybernetics*, (4), 323–334. doi:0.1109/TSMC.1974.5408453

Piechocki, J., Ambroziak, D., Palkowski, A., & Redlarski, G. (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., Tizhoosh, H. R., & Salama, M. M. A. (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., & Meybodi, M. (2010). LACAIS: Learning Automata Based Cooperative Artificial Immune System for Function Optimization. In S. Ranka, A. Banerjee, K. Biswas, S. Dua, P. Mishra, R. Moona, S.-H. Poon, et al. (Eds.), *Contemporary Computing*, Communications in Computer and Information Science (Vol. 94, pp. 64–75). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-14834-7_7

Shatnawi, M., & Nasrudin, M. F. (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., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (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.

Thang Trung Nguyen, Dieu Ngoc Vo, & 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

Thathachar, M. A., & Sastry, P. S. (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., & Sastry, P. S. (2003). *Networks of learning automata: Techniques for online stochastic optimization*. Springer Science & Business Media.

Vapnik, V., Golowich, S. E., & Smola, A. (1997). Support vector method for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems*, 281–287.

Walton, S., Hassan, O., Morgan, K., & Brown, M. R. (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, F., He, X. S., Wang, Y., & Yang, S. M. (2012). Markov model and convergence analysis based on cuckoo search algorithm. *Computer Engineering*, *11*, 055.

Wang, G., Guo, L., Duan, H., Liu, L., Wang, H., & Wang, B. (2012). A hybrid meta-heuristic DE/CS algorithm for UCAV path planning. *Journal of Information and Computational Science*, *9*(16), 4811–4818.

Wang, Y., Cai, Z., & Zhang, Q. (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

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

Xu, X., Ji, Z., Yuan, F., & Liu, X. (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:http://dx.doi.org/10.1016/j.tust.2007.04.011

Yagiz, S., Gokceoglu, C., Sezer, E., & Iplikci, S. (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., & Karahan, H. (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, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on* (pp. 210–214). IEEE.

Yang, X.-S., & Deb, S. (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., & Zhou, Y. (2012). A novel cuckoo search optimization algorithm based on Gauss distribution. *Journal of Computational Information Systems*, *8*(10), 4193–4200.