How to cite this paper:

Shehab, M., Khader, A. T., & Laouchedi, M. (2018). A hybrid method based on cuckoo search algorithm for global optimization problems. *Journal of Information and Communication Technology*, *17* (3), 469-491.

A HYBRID METHOD BASED ON CUCKOO SEARCH ALGORITHM FOR GLOBAL OPTIMIZATION PROBLEMS

[1]Mohammad Shehab, [1]Ahamad Tajudin Khader & [2]Makhlouf Laouchedi

[1]School of Computer Sciences, Universiti Sains Malaysia, Malaysia
[2] Université des Sciences et de Technologies Houari Boumediene, Algeria

moh.shehab12@gmail.com; tajudin@usm.my; laoumakhl@yahoo.fr

ABSTRACT

Cuckoo search algorithm is considered one of the promising metaheuristic algorithms applied to solve numerous problems in different fields. However, it undergoes the premature convergence problem for high dimensional problems because the algorithm converges rapidly. Therefore, we proposed a robust approach to solve this issue by hybridizing optimization algorithm, which is a combination of Cuckoo search algorithmand Hill climbing called CSAHC discovers many local optimum traps by using local and global searches, although the local search method is trapped at the local minimum point. In other words, CSAHC has the ability to balance between the global exploration of the CSA and the deep exploitation of the HC method. The validation of the performance is determined by applying 13 benchmarks. The results of experimental simulations prove the improvement in the efficiency and the effect of the cooperation strategy and the promising of CSAHC.

Keywords: Cuckoo search algorithm, Hill climbing, optimization problems, slow convergence, exploration and exploitation.

Received: 3 August 2017 **Accepted**: 14 May 2018 **Published:** 12 June 2018

INTRODUCTION

Optimization resides in many domains, such as engineering, energy, economics, medical, and computer science (Mustaffa, Yusof, & Kamaruddin, 2013). It is mainly concerned with finding the optimal values for several decision variables to form a solution to problem optimization. This solution is optimally considered when the decision maker is satisfied with it. An optimization problem is the minimization or maximization of a suitable decision-making algorithm normally adapted to the approximation methods. The principle of decision making entails choosing between several alternatives. The result of this choice is the selection of the best decision from all choices (Mohammed, Khader, & Al-Betar, 2016).

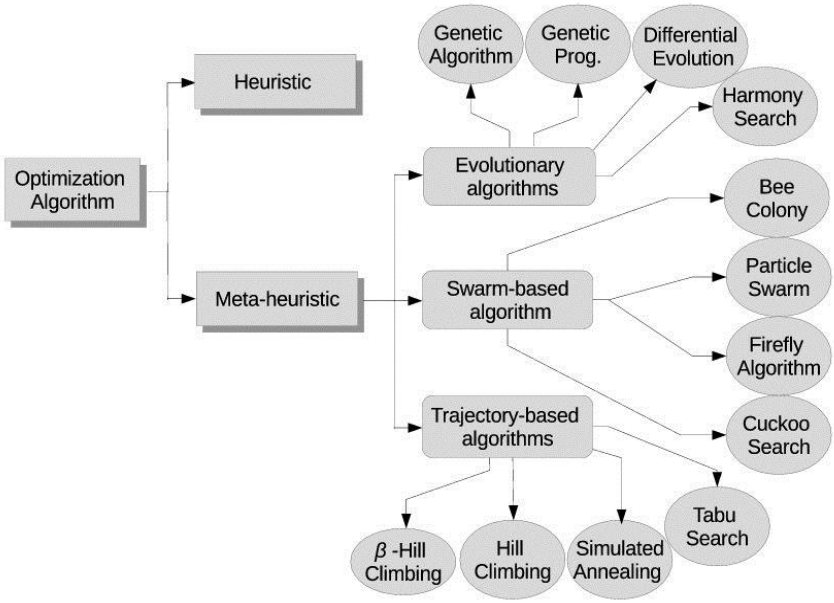

Figure 1. Optimization Algorithms (M. Shehab, Khader, & Al-Betar, 2017).

Optimization algorithms developed based on nature-inspired ideas deal with selecting the best alternative in the sense of the given objective function. The optimization algorithm can be either a heuristic or a metaheuristic approach. Heuristic approaches are problem-designed approaches where each optimization problem has its own heuristic methods that are not applicable for other kinds of optimization problems. The metaheuristic-based algorithm is also

a general solver template that can be adapted for various kinds of optimization problems by properly tweaking its operators and configuring its parameters (Hasan, Quo, & Shamsuddin, 2012). As shown in Figure 1, each optimization algorithm can be categorized into three classes: evolutionary algorithms (EAs), swarm-based algorithms, and trajectory-based algorithms. Examples of EAs include genetic algorithms (GAs) (Holland, 1975), genetic programming (GP) (Koza, 1994), and differential evolution (DE) (Storn & Price, 1996). Examples of swarm-based algorithms include artificial bee colony (ABC) (Karaboga, 2005), particle swarm optimization (PSO)(James & Russell, 1995), and cuckoo search algorithm (CSA) (Yang & Deb, 2009). Examples of trajectory- based algorithms includes tabu search (TS) (Glover, 1977), simulated annealing (SA) (Kirkpatrick, Gelatt, Vecchi, & et. al., 1983), hill climbing (Schaerf & Meisels, 1999).

The performance of the population-based algorithms is measured through checking its ability to establish a proper trade-off between exploration and exploitation. Where the algorithm has a weak balance between exploration and exploitation be more likely to the trapping in local optima, premature convergence and stagnation (Shehab, Khader, & Al-Betar, 2016).

Population-based search algorithm is normally very powerful in exploring several regions of the problem search space. However, it has difficulty in determining the local optima in each region. By contrast, deep searching of the local search-based algorithm is very efficient in a single search space region but not for several search space regions (McMinn, 2004). Thus, sometimes, it is very beneficial to hybridize a local and a population search-based method to complement their advantages in a single optimization framework. Based on the above suggestion and through hybridization, the search can strike a balance between the wide range of exploration and nearby exploitation of the problem search space. In this context, CSA has been hybridized with other local search-based algorithm to improve its performance in tackling complex optimization problems.

The linear least squares problem solved by hybridization algorithm between Newton method (NM) and CSA is called CSANM (Abdel-Baset & Hezam, 2016). The authors benefited from CSA for fast convergence and global search as well as from NM for the ability of strong local search. The experimental results showed the convergence efficiency and computational accuracy of the CSANM in comparison with the basic CSA and HS based on NM (HSNM).

A novel CSA base on the Gauss distribution (GCSA) was proposed by Zheng et al. (2012). In the basic CSA, although it finds the optimum solution, the search entirely depends on random walks. By contrast, fast convergence and

precision cannot be guaranteed. For this purpose, GCSA was introduced to solve the low convergence rate of the basic CSA. GCSA has been applied to solve the standard test functions and engineering design optimization problems. The obtained results showed that the GCSA proved its efficiency through achieving better solutions compared with basic CSA.

Wang et al. (2016) proposed a hybrid algorithm that combined CSA and a HS (HS/CSA) for continuous optimization problems. In the HS/CSA method, the pitch adjustment of HS was used to update the process of the CSA, which leads to the increase of population diversity. The improved elitism scheme was used to retain the best individuals in the cuckoo population as well. The performance of HS/CSA was evaluated by means of testing the set of benchmark functions. The obtained results showed that the HS/CSA achieved better outcomes in comparison with ACO, PSO, GA, HS, DE, and basic CSA.

Quadratic assignment problems (QAPs) are considered to be NP-hard problems, which cannot be easily solved by exact methods. Therefore, Dejam et al., (2012) proposed a hybrid algorithm combined with the CSA of TS (i.e., CSA-TS) to solve QAPs. In their research, the QAPs were initially tackled using CSA. Thereafter, these were combined with TS, which focused on the local search to increase the optimization precision. The experimental results indicated that the proposed algorithm performs better than ABC and GA.

In this work, a new hybrid optimization approach is developed by hybridizing the cuckoo search algorithm with hill climbing to solve global optimization problems. The proposed approach is evaluated on thirteen benchmark functions carefully selected from the literature. Experimental results demonstrate that the CSAHC performs better than Krill heard (KH) (Gandomi & Alavi, 2012), Harmony Search (HS) (Geem, Kim, & Loganathan, 2001), Bat Algorithm (BA) (Yang, 2010a), GA, and the basic CSA.

The paper is organized as follows. Next section describes the CSA and HC in brief. The Proposed Methodology section presents the CSAHC approach in details. Subsequently, our method is evaluated through 13 benchmarks and comparing with 5 methods in the Experimental Results Analysis section. Finally, the conclusion and future works are given in the last section.

PRELIMINARY

Cuckoo Search Algorithm

The use of CSA in the optimization context was proposed by Yang and Deb, (2009). To date, work on this algorithm has significantly increased, and the

CSA has succeeded in having its rightful place among other optimization methodologies (Fister Jr, Yang, Fister, & Fister, 2014). This algorithm is based on the obligate brood parasitic behavior found in some cuckoo species, in combination with the Levy flight behavior discovered in some birds and fruit flies. The CSA is an efficient metaheuristic swarm based algorithm that efficiently strikes a balance between local nearby exploitation and global-wide exploration in the search space problem (Shehab, Khader, & Laouchedi, 2017).

The cuckoo has a specific way of laying its eggs to distinguish it from the rest of the birds (Yang & Deb, 2014). The following three idealized rules clarify and describe the standard cuckoo search:

- Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs will be carried over to the next generations.
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $P\alpha \in (0,1)$. In this case, the host bird can either get rid of the egg or simply abandon the nest and build a completely new nest. In addition, probability $P\alpha$ can be used by the n host nest to replace the new nests.

```
1: Objective function f(X), X = (f(x1, x2, ..., xd)^T
2: Generate initial population of n host nests Xi (i=1, 2, …, n)
3: While t < Max_itertions do
4:       Get a cuckoo randomly by Levy flights
5:       Evaluate its quality/ fitness Fi
6:       Choose a nest among n (say, j) randomly
7:       If Fi > Fj then
8:           replace j by the new solution;
9:       End If
10:      A fraction (Pa) of worse nests are abandoned and
         new ones are built;
11:      Keep the best solutions
12:      Rank the solutions and find the current best
13: End While
14: Postprocess results and visualization
```

Figure 2. Pseudo code of the Cuckoo Search Algorithm

Figure 2 shows the pseudo code of the CSA search process. Similar to other swarm-based algorithms, the CSA starts with an initial population of *n* host nests. These initial host nests will be randomly attracted by the cuckoos with eggs and also by random Levy flights to lay the eggs. Thereafter, nest quality will be evaluated and compared with another random host nest. In case the host nest is better, it will replace the old host nests. This new solution has the egg laid by a cuckoo. If the host bird discovers the egg with a probability $P\alpha \in (0,1)$, the host either throws out the eggs, or abandons it and builds a new nest. This step is done by replacing the abundant solutions with the new random solutions.

Yang and Deb used a certain and simple representation of the implementation, with each *egg* representing a solution. As the *cuckoo* lays only one egg, it also represents one solution. The purpose is to increase the diversity of new, and probably better, cuckoos (solutions) and replace them instead with the worst solutions. By contrast, the CSA can be more complicated by using multiple eggs in each nest to represent a set of solutions.

The CSA, as a bat algorithm (Yang, 2010a) and an FA (Yang, 2010b), uses a balance between exploration and exploitation. The CSA is equiponderance to the integration of a Levy flights. When generating new solutions x^{t+1} for, say, a cuckoo *i*, a Levy flight is performed

$$x_i^{t+1} = x_i^t + \alpha \oplus le'vy\ (\lambda) \tag{1}$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interests. In most cases, we can use $\alpha = 1$. The x_i^t in the above equation represents the current location, which is the only way to determine the next location x_i^{t+1}. This is called random walk or Markov chain. The product $\oplus$ means entry wise multiplications. This entry wise product is similar to those used in PSO, but here the random walk via Levy flight is more efficient in exploring the search space as its step length is much longer in the long run. A global explorative random walk by using Levy flights can be expressed as follows:

$$le'vy \sim u = t^{-\lambda}, \quad 1 < \lambda \leq 3 \tag{2}$$

where λ is a parameter which is the mean or expectation of the occurrence of the event during a unit interval. Here the steps essentially form a random walk process with a power law step-length distribution with a heavy tail. Some of

the new solutions should be generated by Levy walk around the best solution obtained so far, this will speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization and whose locations should be far enough from the current best solution, this will make sure the system will not be trapped in a local optimum.

Hill Climbing

Hill Climbing (HC) is a mathematical optimization technique which belongs to the family of local search (Schaerf & Meisels, 1999). It searches for a better solution in the neighborhood through evaluating the current state. If it is also goal state, then return to it and quit. Otherwise, continue updating the current state, if possible. Then, loop until a solution is found or until there are no new operators left to be applied in the current state. Also, inside the loop there are two steps. The first step, select an operator that has not yet been applied to the current state and apply it to produce the new state. The second step, evaluate the new state. Figure 3 shows the pseudo-code of the HC algorithm, which proves the simplicity of hill climbing.

Based on the above, in HC the basic idea is to always head towards a state which is better than the current one. So, it always improves the quality of a solution (Burke & Newall, 2002).

```
1: i = initial solution
2: While f(s) ≤ f (i) s € Neighbours (i) do
3:        Generates an s € Neighbours (i);
4:        If fitness (s) > fitness (i) then
5:            Replace s with the i;
6:        End If
```

Figure 3. Pseudo code of the Hill Climbing method

HC has some advantages, such as it can easily be adjusted to the problem at hand. Almost any aspect of the algorithm may be changed and customized. For example, It can be used in conversions as well as discrete domains (Alajmi et al., 2011; Rubio & Gámez, 2011).

THE PROPOSED METHODOLOGY: CSA-HILL CLIMBING

Based on the introduction of CSA and HC in the previous sections, this section provides a detailed description of the proposed cuckoo search algorithm with hill climbing (CSAHC).

CSA based on the obligate brood parasitic behavior found in some cuckoo species, in combination with the Levy flight, which it is a type of random walk which has a power law step length distribution with a heavy tail. It is inspired from behavior discovered of some birds and fruit flies (Yang & Deb, 2009). Levy flight used for global exploration and proved its efficiency through achieving good results (Pavlyukevich, 2007; Yang & Deb, 2013). Thus, the CSA is considered as an efficient metaheuristic swarm-based algorithm that efficiently strikes a balance between local nearby exploitation and global wide exploration in the search space problem (Roy & Chaudhuri, 2013b). However, sometimes it exploits solutions poorly with slow convergence. For that reason, the proposed algorithm improves the search ability of the basic CSA through combining it with HC method for deepening exploitation; so-called CSAHC algorithm is used to optimize the benchmark functions (refer Figure 4).

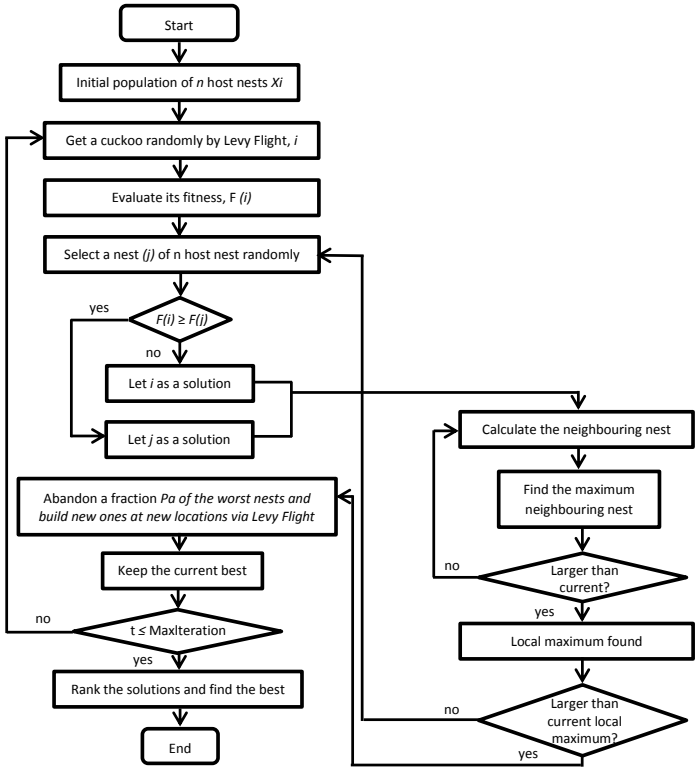

Figure 4. Flowchart of the CSAHC Algorithm.

CSAHC starts the search by applying the standard cuckoo search for the number of iterations. The best-obtained solution is then passed to the HC

to accelerate the search and overcome the slow convergence of the standard cuckoo search algorithm. HC is an iterative algorithm that starts with an arbitrary solution to a problem and subsequently attempts to determine a better solution by incrementally changing a single element of the solution. When the change produces a better solution, incremental change is performed on the new solution, which is repeated until no further improvements can be found. It then returns the solution to the CSA to check it through the fraction probability $P\alpha$.

THE EXPERIMENTAL RESULTS ANALYSIS

In this section, the proposed CSAHC was tested through an array of experiments. For testing purposes, we implemented the original version of CSA. We compared results of CSAHC with other methods. This comparison is shown in the tables within this section.

All the experiments are conducted using a computer with processor Intel(R) Core (TM) i7-6700K CPU 4.00 GHz with 16 GB of RAM and 64-bit for Microsoft Windows 10 Pro. The source code is implemented using MATLAB (R2015a).

Benchmark Functions

To test the performance of a CSAHC, 13 well-known benchmark functions are used for comparison. Table 1 describes these benchmark functions in terms of the optimum solution after a predefined number of iterations and the rate of convergence to the optimum solution. Further information about all the benchmark functions can be found in (Yao, Liu, & Lin, 1999; Simon, 2008; Jamil & Yang, 2013).

Table 1

Benchmark Functions

symbol	Function	Definition
F1	Ackley	$f(\bar{x}) = 20 + e - 20.e^{-0.2\sqrt{\frac{1}{2}\sum_{n=1}^{1} x \frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)}}$
F2	Griewank	$f(\bar{x}) = \sum_{i=1}^{n} \frac{x_1^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{xi}{\sqrt{i}}\right) + 1$

(continued)

symbol	Function	Definition
F3	Penalty #1	$f(\vec{x}) = \frac{\pi}{30}\left\{10sin^2(\pi yi) + \sum_{i=1}^{n-1}(yi-1)^2 \bullet [1+10sin^2(\pi yi+1)] + (yn-1)^2\right\}$ $\sum_{i+1}^{n} u\,(x_i, 10, 100, 4), yi = 1+0.25(x_i+1)$
F4	Penalty #2	$f(\vec{x}) = 0.1\left\{sin^2(3\pi x1) + \sum_{i=1}^{n-1}(xi-1)^2 \bullet [1+sin^2(3\pi xi+1)] + (x_n - 1)^2[1+sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F5	Quartic with noise	$f(\vec{x}) = \sum_{i=1}^{n}(i.x_i^4 + U(0,1))$
F6	Rastrigin	$f(\vec{x}) = 10.n\sum_{i=1}^{n}(x_1^2 - 10.\cos(2\pi xi))$
F7	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1}[100(xi+1-x_1^2) + (x_i-1)^2]$
F8	Schwefel 2.26	$f(\vec{x}) = 418.9829 \text{ x } D - \sum_{I=1}^{D} x_i \sin(\lvert x_i\rvert^{\frac{1}{2}})$
F9	Schwefel 1.2	$f(\vec{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} xj\right)2$
F10	Schwefel 2.22	$f(\vec{x}) = \sum_{i=1}^{n}\lvert xi\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
F11	Schwefel 2.21	$f(\vec{x}) = \underset{i}{max}\left\{\lvert x_i\rvert, 1 \le i \underline{<} n\right\}$
F12	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F13	Step	$f(\vec{x}) = 6.n + \sum_{i=1}^{n}\lvert x_i\rvert$

Experimental results and algorithms settings

A. Comparisons with other methods

CSAHC was initially compared with the global optimization problems of five optimization algorithms, namely, KH, HS, GA, BA, and CSA.

In our simulations, similar parameters for CSA have been used, the number of host nests $n = 20$ and probability of discovery $P\alpha = 0.25$. The tests have been run on 10, 25, 50, and 100 dimensions for a maximum of 100000 function evaluations. All tests have been run 100 times. Tables 2 and 3 show the different scales used to normalize the values to illustrate the differences of the six methods.

Table 2 shows that CSAHC performs the best on 11 of the 13 benchmarks which are F1-F4, F6-F10, and F12-F13. CSA is the second most effective, performing the best on the benchmarks F1-F2, F4-F5, and F13. Followed by GA, KH, BA, HS, respectively. Table 3 illustrated the average of results. Where, could be observed CSAHC method performs the most effective at determining objective function minimum on 10 of the 13 benchmarks F2-F4, F6-F9, and F11-F13. CSA and GA are the second most effective, performing best on the benchmarks F4-F5, F10, and F13 for the CSA. While, F2, F11-F12, and F13 for the GA. Followed by KH, BA, and HS, respectively.

Table 2

Best normalized optimization results

Fun	CSAHC	CAS	BA	GA	HS	KH
F1	**1.00**	**1.00**	8.22E_06	2.98E+02	4.29E+05	**1.00**
F2	**1.00**	**1.00**	3.09E+05	9.32E+02	6.15E+04	2.51E+04
F3	**1.00**	3.25E+02	8.64E+03	7.45E+01	5.08E+03	**1.00**
F4	**1.00**	**1.00**	2.85E+05	3.63E+02	3.85E+04	8.25E+04
F5	4.25E+00	**1.00**	7.66E+01	**1.00**	**1.00**	5.44E+00
F6	**1.00**	8.29E+03	1.25E+04	2.21E+02	8.22+05	4.36E+04
F7	**1.00**	5.02 E+02	8.68 E_05	6.78 E+01	1.52 E+05	1.27 E+05
F8	**1.00**	2.98E+03	5.43E+05	9.33E+02	7.69E+06	4.02E+05
F9	**1.00**	3.39E+02	7.09E+05	1.28E+02	7.63E+04	5.46E+02
F10	**1.00**	1.68E+03	4.21E+04	**1.00**	1.08E+03	3.31E+02
F11	7.54E+01	3.52E+01	8.59E+01	**1.00**	9.79E+01	7.25E+01
F12	**1.00**	8.29E+00	**1.00**	7.02E+02	5.93E+03	1.41E+04
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Total	**11**	**5**	**2**	**4**	**2**	**3**

Table 3

Mean Normalized Optimization Results

Fun	CSAHC	CAS	BA	GA	HS	KH
F1	2.15E-01	5.18E+01	5.32E+04	2.72E+03	6.96E+03	**1.00**
F2	**1.00**	4.38E+02	8.87E+05	**1.00**	2.36E+04	725E+03
F3	**1.00**	6.35E+01	2.41E+36	1.93E+03	2.02E+05	5.17E+02
F4	**1.00**	**1.00**	7.11E+06	2.11E+02	8.80E+04	5.67E+05
F5	8.51E+00	**1.00**	1.84E+05	4.22E+03	1.65E+05	1.89E+04
F6	**1.00**	5.75E+03	9.42E+03	5.01E+02	7.35E+04	9.78E+03
F7	**1.00**	9.89E+02	4.08E+05	9.85E+03	9.52E+06	2.74E+05
F8	**1.00**	1.75E+01	7.49E+05	9.15E+01	2.59E+04	7.35E+04
F9	**1.00**	7.07E+02	1.69E+06	3.46E+03	7.13E+05	1.28 E+05
F10	1.08E+00	**1.00**	3.28E+06	**1.00**	1.84E+04	1.30E+03
F11	**1.00**	4.26E+02	8.32E+06	**1.00**	8.63E+05	3.82 E+03
F12	**1.00**	4.73E+03	1.54E+03	8.55E+02	124E+06	1.68E+05
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Total	**10**	**4**	**1**	**4**	**1**	**2**

Further, the most representative convergent curves are provided (see Figure 5 - Figure 10). The values in the figures are the mean function optimum, which are the true values.

From Figure 5, apparently, CSAHC is well capable of finding the better solutions than all other methods. Here, HS converges sharply at the first search stage, however, soon it gets trapped into the sub-minima and the

global minimum decreases slightly. In addition, in this function, BA is closed to CSAHC in the first stage, but the difference is increasing in the second stage. Each of BA, CSA, GA, and KH have moved to the best solutions initially, while later CSA converges to the better minimum than the others and CSAHC is the best of all. Figure 6 shows that CSAHC has the best performance among the six methods, while CSA ranks second. GA has the third best performance with a relatively slow and stable convergence rate.

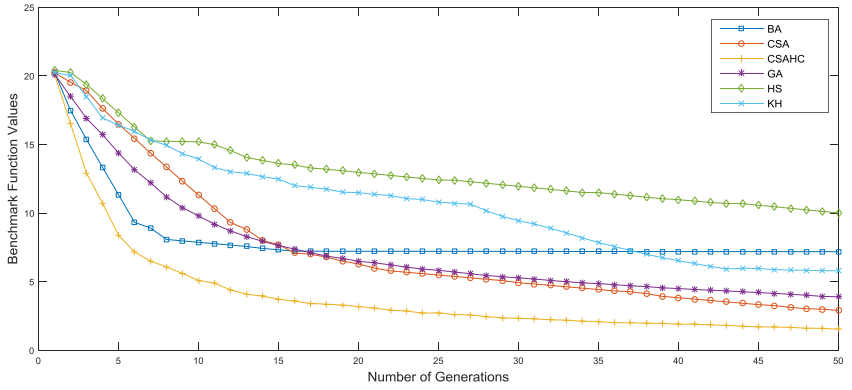

Figure 5. Performance comparison for the F1 Ackley function.

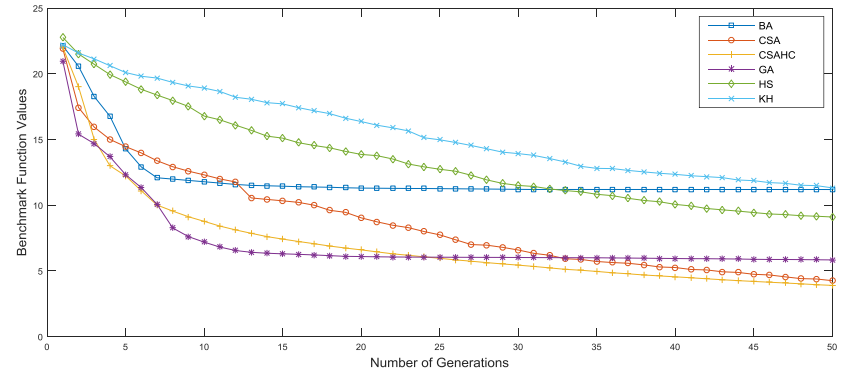

Figure 6. Performance comparison for the F3 Penalty 1.

Figures 7, 8, and 9 shows that CSAHC is capable of finding better solutions compared with all the other methods. In the Figure 7 CSA achieved best solutions from the beginning until 25th generation, and then GA got the best solutions from 26th generation until 43rd generation, followed by CSAHC with best solutions until the end. The results in Fig. 8 are almost same with the results achieved in Figure 7. But, in Fig. 8, the results of CSAHC, GA, CSA, and KH are close together with a preference for CSAHC. Figure 9, have the same ranking for Figure 7 and Figure 8. However, the CSAHC in Figure 9 has clear outperformed comparing with the other methods.

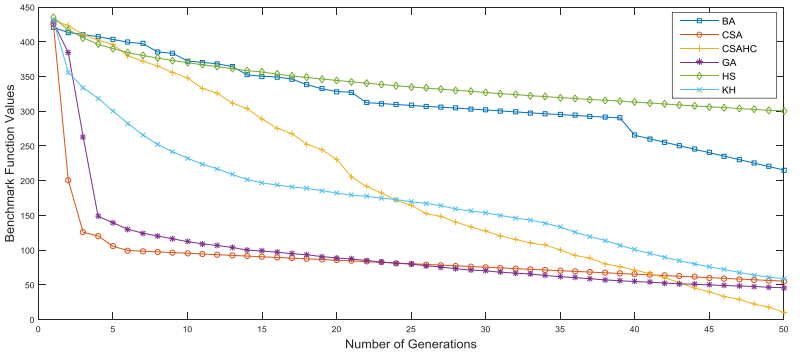

Figure 7. Performance comparison for the F6 Rastrigin function.

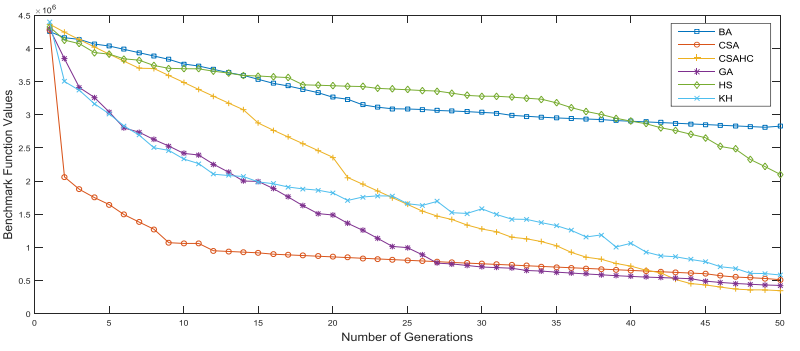

Figure 8. Performance comparison for the F9 Schwefel 1.2 function

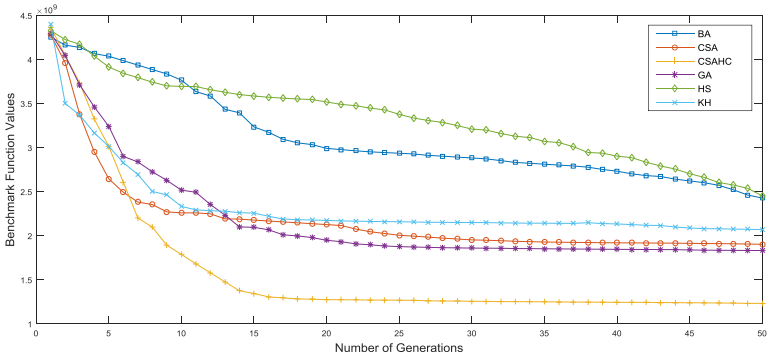

Figure 9. Performance comparison for the F12 Sphere function.

Figure 10 shows that CSAHC achieved the best solution in the especially in the first part of the results, with simple superiority for basic CSA. However, the GA outperforms both of basic CSA and CSAHC especially at the last part. An analysis of Figures 5 to 10 reveals that our proposed metaheuristic CSAHC method greatly outperforms the other methods.

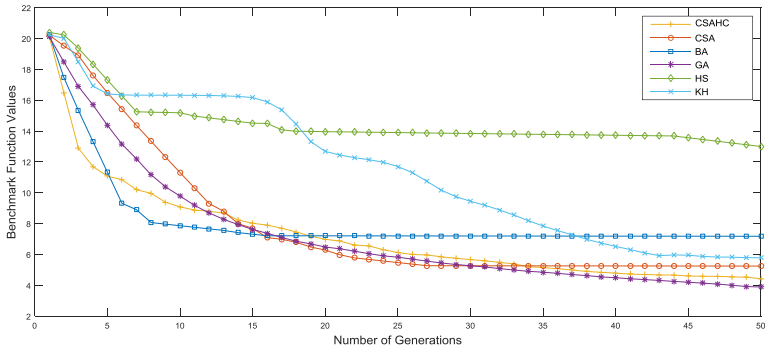

Figure 10. Performance comparison for the F13 Step function.

B. Influence of control parameter

Parameter setting plays an important role in the performance of metaheuristic methods when solving different problems. In this article are the number of host nests (population size **n**) and the probability of discovery ($P\alpha$) are thoroughly studied with 100 trials, which are implemented in the above problems to search for the best solution and mean as shown in Tables 4, 5, 6, and 7.

- **Population size *n***

The influence of n is investigated through an array of simulations with **n** = 5, 10, 15, 20, 50, 100, 150, 250, 500. $P\alpha$ = 0.25 (see Tables 4 and 5).

From Tables 4 and 5, we can see that the superior performance of CSAHC when the value of **n** = 20. While performance decreases as the value of n increases. This due to increasing the value of **n** that mean increase the search space, therefore the performance of CSAHC will decrease.

Table 4

Best Normalized Optimization Results with Different n

	Population size n								
	5	**10S**	**15**	**20**	**50**	**100**	**150**	**250**	**500**
F1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	26.07	41.29
F2	0.84	0.84	0.18	**1.00**	1.32	16.32	23.32	37.32	56.32
F3	**1.00**	**1.00**	**1.00**	**1.00**	1.16	1.00	72.12	82.23	29.16
F4	12.67	1.08	1.41	**1.00**	1.74	19.43	14.34	58.27	38.04
F5	10.25	39.02	6.83	**1.00**	**1.00**	18.24	26.34	24.51	47.70
F6	50.47	53.44	1.65	0.02	**1.00**	33.83	62.69	61.83	56.65
F7	**1.00**	**1.00**	**1.00**	1.00	1.13	51.40	21.45	19.33	31.60
F8	13.25	1.00	2.09	1.74	1.39	45.24	63.16	83.13	52.08
F9	44.86	67.99	35.67	17.2	1.02	91.54	88.07	91.00	64.17
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	57.09	39.81	33.92	38.91
F11	18.02	22.80	4.05	2.98	1.08	19.06	31.51	82.07	75.06
F12	15.20	22.24	10.07	**1.00**	1.14	28.20	47.09	73.30	95.10
F13	5.99	7.98	**1.00**	**1.00**	1.02	36.01	71.02	64.07	82.01
Total	**4**	**5**	**5**	**9**	**3**	**2**	**1**	**0**	**0**

- **Discovery rate** $P\alpha$

Firstly, the effect of the elitism parameter is studied in the benchmark problems with the elitism parameter $P\alpha$ = 0, 0.1, 0.2, ..., 0.8, 0.9, 1 and *n* = 20 (see Tables 6 and 7).

Table 5

Best normalized optimization results with different n

	Population size n								
	5	**10S**	**15**	**20**	**50**	**100**	**150**	**250**	**500**
F1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	2.85	11.77	25.63
F2	4.52	1.80	1.07	**1.00**	5.32	11.32	18.39	33.50	51.22
F3	6.03	1.17	**1.00**	**1.00**	7.08	17.21	22.02	47.13	55.27
F4	3.52	3.01	**1.00**	1.01	6.12	10.17	14.61	28.32	37.26
F5	2.49	1.89	1.22	**1.00**	3.74	21.55	31.76	69.08	92.15
F6	8.26	4.92	3.08	**1.00**	9.51	17.65	29.72	42.30	66.85
F7	1.67	1.01	**1.00**	**1.00**	5.27	9.53	16.17	24.12	48.06
F8	4.00	1.64	4.97	1.13	**1.00**	8.92	19.63	31.27	57.67
F9	**1.00**	**1.00**	2.51	**1.00**	2.99	13.37	21.61	35.53	60.01
F10	**1.00**	**1.00**	**1.00**	**1.00**	2.05	4.07	11.79	19.08	34.17
F11	38.30	25.73	12.59	3.73	**1.00**	41.26	47.84	67.11	82.65
F12	8.63	4.22	2.58	**1.00**	10.24	14.32	22.09	40.89	65.89
F13	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	9.01	15.28	21.01	43.05
Total	**4**	**4**	**6**	**10**	**4**	**1**	**0**	**0**	**0**

From Table 6, obviously, it can be seen that CSAHC performs the best when $P\alpha$ = 0.1 and 0.2. Especially, for the F1 until F5, CSAHC has the similar performance; that is, the elitism parameter $P\alpha$ has little influence on these three benchmark functions. Furthermore, when $P\alpha$ = 0 and from 0.3 until 0.8, CSAHC performs achieved almost same results. However, the worst results when $P\alpha$ = 0.9 and 1. In Table 7, there is a clear superiority for the CSAHC when the $P\alpha$ = 0.2, followed by $P\alpha$ = 0.1 and 0.3 almost the same results. Finally, all other values of are achieved nearby results. In short, CSAHC has the best performance when $P\alpha$ = 0.2.

Table 6

Best Normalised Optimization Results with Different Pα

	Discovery rate Pa				
Fun.	0	0.1	0.2	0.3	0.4
F1	2.93E+01	1.00	1.00	1.00	1.24E+02
F2	3.28E+02	1.00	1.00	5.36E+02	1.69E+01
F3	1.16E+01	1.00	1.00	7.30E+02	6.46E+01
F4	2.48E+00	1.00	1.00	6.22E+01	5.32E+02
F5	2.55E+01	1.00	1.00	2.14E+00	6.62E+02
F6	2.48E+01	1.21E+00	1.00	4.22E+01	9.43E+01
F7	2.55E+00	2.81E+02	1.00	6.24E+00	1.57E+02
F8	1.00	1.00	1.00	5.84E+02	3.89E+02
F9	1.00	2.22E+01	1.00	3.46E+01	8.33E+01
F10	1.01E+00	3.71E+02	1.00	1.00	6.31E+01
F11	2.12E+00	4.65E+00	1.17E+00	2.18E+01	1.00
F12	1.01E+00	9.39E+01	1.00	5.15E+02	2.31E+02
F13	2.12E+00	4.65E+01	1.17E+00	1.00	1.00
Total	2	6	11	3	2

	Discovery rate Pa					
Fun.	0.5	0.6	0.7	0.8	0.9	1
F1	3.67E+02	7.36E+03	6.89E+03	7.95E+02	4.28E+02	8.59E+03
F2	1.26E+03	4.23E+02	1.55E+03	4.47E+03	4.21E+02	4.22E+03
F3	5.68E+02	1.22E+03	1.00	2.14E+02	8.58E+03	6.91E+03
F4	1.90E+03	3.31E+02	1.73E+02	8.27E+02	3.38E+02	1.23E+01
F5	1.00	5.11E+02	3.01E+03	2.93E+01	1.42E+03	5.94E+02
F6	4.69E+02	7.01E+02	1.88E+02	3.78E+02	1.64E+02	1.23E+01
F7	7.63E+02	5.17E+01	7.39E+03	5.06E+03	1.15E+03	3.35E+03
F8	5.63E+03	5.39E+03	6.36E+01	4.85E+02	1.02E+03	5.18E+02
F9	8.40E+01	1.82E+02	2.54E+03	7.43E+02	1.11E+03	1.16E+01
F10	5.36E+03	1.00	5.20E+02	1.95E+03	1.08E+02	2.82E+01
F11	7.95E+02	1.72E+02	6.99E+03	1.00	1.17E+02	1.87E+01
F12	5.33E+02	1.00	1.13E+01	9.52E+02	1.06E+03	4.36E+01
F13	8.36E+03	8.39E+01	2.54E+02	4.65E+01	5.94E+03	7.92E+01
Total	1	2	1	1	0	0

Table 7

Mean normalised optimization results with different Pα

	Discovery rate Pa				
Fun.	0	0.1	0.2	0.3	0.4
F1	2.81E+01	6.84E+02	1.00	5.84E+02	6.24E+02
F2	1.00	1.00	106.E+00	3.46E+00	2.69E+02
F3	2.22E+02	1.00	1.00	1.00	8.46E+02
F4	3.71E+02	2.48E+01	1.00	2.18E+01	5.32E+01
F5	1.65E+02	2.55E+00	1.00	5.15E+00	1.00
F6	1.21E+00	2.48E+00	1.00	1.10E+01	7.43E+02
F7	1.08E+02	2.55E+01	1.00	6.24E+01	1.95E+02
F8	4.47E+01	1.00	1.00	1.00	1.00
F9	1.14E+02	1.00	1.00	1.26E+01	2.52E+03
F10	6.27E+00	1.00	1.00	1.00	6.89E+01
F11	2.93E+01	1.00	1.00	1.90E+00	5.63E+02
F12	6.27E+02	1.01E+01	1.00	1.00	1.40E+00
F13	2.93E+02	2.12E+01	1.00	1.90E+01	1.94E+01
Total	1	6	12	4	2

	Discovery rate Pa					
Fun.	0.5	0.6	0.7	0.8	0.9	1
F1	5.06E+02	1.06E+03	4.05E+03	6.89E+03	7.28E+03	8.89E+03
F2	4.85E+03	4.23E+03	8.59E+03	1.55E+04	8.35E+03	8.55E+03
F3	5.11E+02	7.22E+02	4.22E+01	1.00	8.58E+02	1.00
F4	7.01E+01	3.31E+01	6.91E+02	1.73E+03	3.38E+03	4.73E+03
F5	5.17E+02	1.15E+03	2.23E+03	3.01E+03	4.42E+03	4.61E+03
F6	5.39E+01	1.02E+02	1.00	1.88E+02	6.89E+02	8.12E+02
F7	1.82E+03	6.11E+03	7.23E+03	1.35E+04	1.55E+04	3.35E+04
F8	1.00	1.08E+01	5.94E+01	2.18E+02	1.00	5.18E+02
F9	5.81E+03	7.17E+03	6.89E+03	7.21E+03	1.73E+03	1.86E+03
F10	1.00	1.06E+02	4.14E+02	5.55E+02	6.01E+02	6.82E+02
F11	7.32E+03	1.00	1.22E+02	3.17E+02	1.88E+03	2.87E+03
F12	1.94E+02	1.00	1.14E+01	1.00	3.35E+01	1.01E+02
F13	7.95E+02	1.23E+03	3.42E+03	3.17E+03	9.18E+02	1.87E+03
Total	2	2	1	2	1	1

CONCLUSION AND FUTURE WORK

In the present work, a novel metaheuristic CSAHC method is proposed for solving global optimisation tasks. We improved the CSA by combining it with HC and evaluated the performance of the CSAHC on the 13 benchmark functions. The hybridization enhanced the exploration of basic CSA by using the HC which is capable of dealing with local searches. Furthermore, CSAHC is investigated on 13 benchmark functions. Results showed that comparing CSAHC with other search methods, such as the original CSA, original BA, GA, HS, and KH, improves its efficiency and effect. The CSAHC can be applied to more benchmark functions, including some real-world optimization problems for further examinations.

ACKNOWLEDGEMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for profit sectors.

REFERENCES

Abdel-Baset, M., & Hezam, I. M. (2016). Solving linear least squares problems based on improved cuckoo search algorithm. *Mathematical Sciences Letter, 5*(2), 199-202.

Alajmi, B. N.; Ahmed, K. H.; Finney, S. J. & Williams, B. W. (2011). Fuzzy-logic-control approach of a modified hill-climbing method for maximum power point in microgrid standalone photovoltaic system *IEEE Transactions on Power Electronics, IEEE, 26*, 1022-1030.

Burke, E. K., & Newall, J. P. (2002). Enhancing timetable solutions with local search methods. In *International Conference on the Practice and Theory of Automated Timetabling,* 2740, 195–206.

Dejam, S., Sadeghzadeh, M. & Mirabedini, S. J. (2012). Combining cuckoo and tabu algorithms for solving quadratic assignment problems. *Journal of Academic and Applied Studies*, *Liteseer*, *2*(12), 1-8.

Fister Jr, I., Yang, X.-S., Fister, D., & Fister, I. (2014). Cuckoo search: a brief literature review. In *Cuckoo Search and Firefly Algorithm*, *516*, 49–62.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, *17*(12), 4831–4845.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*(2), 60–68.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*(1), 156–166.

Hasan, S., Quo, T. S., & Shamsuddin, S. M. (2012). Artificial fish swarm optmization for multilayer network learning in classification problems. *Journal of Information & Communication Technology*, *11,* 37-53.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann, Anbor, USA: U Michigan Press.

James, K., & Russell, E. (1995). Particle swarm optimization. In *Proceedings of 1995 IEEE International Conference on Neural Networks*, *4*, 1942–1948.

Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*(2), 150–194.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report TR06. Technical Report TR06. Erciyes University, Engineering Faculty, Computer Engineering Department.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., & others. (1983). Optimization by simmulated annealing. *Science*, *220*(4598), 671–680.

Koza, J. R. (1994). Genetic programming II: Automatic discovery of reusable subprograms. Cambridge, MA, USA: MIT Press.

McMinn, P. (2004). Search-based software test data generation: A survey. *Software Testing, Verification and Reliability*, *14*(2), 105–156.

Mohammed, S. M. Z., Khader, A. T., & Al-Betar, M. A. (2016). 3-SAT using island-based genetic algorithm. *IEEJ Transactions on Electronics, Information and Systems*, *136*(12), 1694–1698.

Mustaffa, Z., Yusof, Y., & Kamaruddin, S. (2013). Enhanced Abc-Lssvm for Energy fuel price prediction. *Journal of Information and Communication Technology*, *12*, 73–101.

Rubio, A. & Gámez, J. A. (2011).Flexible learning of k-dependence Bayesian network classifiers. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation,* 1219-1226. Doi: 10.1145/2001576.2001741

Schaerf, A., & Meisels, A. (1999). Solving employee timetabling problems by generalized local search. In the 6th *Congress of the Italian Association for Artificial Intelligence.* Advancec in Artifical Intelligence, Bologna, Italy, LNAI, 1972, 380–389.

Shehab, M., Khader, A. T., & Al-Betar, M. A. (2017). A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing*, *61*, 1044-1069.

Shehab, M., Khader, A. T., & Laouchedi, M. (2017). Modified Cuckoo search algorithm for solving global optimization problems. Paper presented in the *International Conference of Reliable Information and Communication Technology*, *5*, 561–570.

Shehab, M. M., Khader, A. T., & Al-Betar, M. A. (2016). New selection schemes for particle swarm optimization. *IEEJ Transactions on Electronics, Information and Systems*, 136(12), 1706-1711.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, *12*(6), 702–713.

Storn, R., & Price, K. V. (1996). Minimizing the real functions of the ICEC'96 contest by differential evolution. In *International Conference on Evolutionary Computation*, 842–844. Doi: 10.11909/ICEC. 1996.

Yang, X.-S. (2010a). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization, 284,* 65-74. Springer.

Yang, X.-S. (2010b). Firefly algorithm. Nature-inspired metaheurictic algorithms. *Wiley Online Library,* 221–230. Doi 10.1002/9780470640425

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. Paper presented in *World Congress on Nature & Biologically Inspired Computing*, pp. 210–214. Doi: 10.1109/NABIC.2009.5393690

Yang, X.-S., & Deb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications*, *24*(1), 169–174.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, *3*(2), 82–102.

Wang, G.-G., Gandomi, A. H.; Zhao, X. & Chu, H. C. E. (2016). Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing*, *20*(1), 273-285

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