Comparison of Performances of Jaya Algorithm and Cuckoo Search Algorithm Using Benchmark Functions

Mashuk Ahmed, Abdullah B. Nasser, Kamal Z. Zamli and Sulistyo Heripracoyo

June 27, 2021

Comparison of Performances of Jaya Algorithm and Cuckoo Search Algorithm Using Benchmark Functions

Mashuk Ahmed[1], Abdullah B. Nasser[1], Kamal Z. Zamli[1], and Sulistyo Heripracoyo[2]

[1] Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, , Pekan 26600, Pahang, Malaysia
[2] School of Information Systems, Bina Nusantara University, Information Systems Department, Jakarta, 11480, Indonesia
MCS20004@student.ump.edu.my
abdullahnasser@ump.edu.my
kamalz@ump.edu.my
hpracoyo@binas.edu

Abstract. Nowadays, selecting the best possible solution among several solutions becomes an important skill for engineering and research. Therefore, engineers are turning to optimization methods as a complementary alternative strategy of exhaustive searching. Metaheuristic algorithms have been used successfully for solving different optimization problems. To help engineers select the best metaheuristic algorithms for their problems, there is a need to evaluate the performance of different metaheuristic algorithms against each other using common case studies. This paper aims to compare the performance of two metaheuristic algorithms which are Jaya Algorithm (JA) and Cuckoo Search (CS) using some common benchmark functions. CS and JA have implemented in the same platform (Intellij IDEA Community Edition 2020.2.3) using the same language (Java). The experimental results show that JA has better and consistent performance as compared to CS in most cases in terms of execution time and test suite size; however, the performance of JA is still within acceptable ranges.

Keywords: Jaya Algorithm, Cuckoo Search Algorithm, Metaheuristic Algorithm, Optimization, Execution Time.

1 Introduction

Besides the analysis, optimization has been indicated as one of the vital stages of engineering design [1]. Optimization is a vital part of software engineering which allows the engineer to find the optimum solution in the presence of design constraints and criteria. Whereas the analysis stage of the engineering design is the use of the mathematic model to predict the design results optimization method such as metaheuristic algorithms have been used successfully for solving different optimization problems such as robotic path planning, optimization of neural networks, minimizing weights of Truss Structures, task scheduling over cloudlets, optimization of integrated process planning, dynamic vehicle routing, susceptibility mapping of forest fire, etc. [2] Engineers and researchers have been working with the implementation of metaheuristic algorithms for a long time. As a result, metaheuristic algorithms such as Genetic Algorithm(GA), Flower Pollination Algorithm(FPA), Particle Swarm Optimization (PSO), Jaya Algorithm(JA), Hill Climbing(HC), Cuckoo Search (CS), have become very popular in the field of optimization. Metaheuristic algorithm based optimizations are also being mentioned as metaheuristic optimization now [3]. Metaheuristic based optimizations possess stochastic characteristics [4]. These algorithms have a wide range of applications. Every metaheuristic algorithm has unique characteristics. While performing an optimization, there may be a need for an algorithm with particular characteristics to solve the optimization problem. So, a specific algorithm with some characteristics may not be enough to solve a specific optimization problem.

Selecting the most suitable metaheuristic algorithm is vital in optimization. Also, it's quite a challenging task, especially when the optimization problem is complex [5]. So, it's necessary to have ideas about the comparison of different metaheuristic algorithms. The comparisons of the algorithms help to find out their unknown characteristics. For this reason, a comparison between two of the metaheuristic algorithms have been done, which are Jaya Algorithm (JA) and Cuckoo Search (CS) Algorithm. There has been a minimization process of some functions. The coding of the algorithms has been done in Java using Intellij IDEA Community Edition 2020.2.3. The result of the comparison will be helpful to determine which one is more suitable for a specific optimization problem.

The next sections will have a detailed discussion of the algorithms and experiments. In section 2, there will be an overview of JA and CS. Section 3 will discuss the related works and applications of JA and CS. Section 4 will discuss the methodologies. Section 5 will describe the tests and results, and related data. Section 6 will analyze the 2 algorithms based on the test results. Finally, section 7 will conclude the overall discussions with future works.

2 Background

Jaya Algorithm (JA) and Cuckoo Search (CS) Algorithm, both are efficient in generating test suite, trying to reach the best solutions and avoid worse solutions.

2.1 Jaya Algorithm

JA is a global algorithm in the field of optimization. It’s one of the recent metaheuristic algorithms. The algorithm is powerful and simple to implement. JA has some advantages. The potential solution for JA is based on finding the most optimum solution and avoiding the worst solution. JA doesn't require tuning [6]. It's because, mainly the common controlling parameters are required for JA. There’s no need for control parameters that are specific to the algorithm.

2.2 Cuckoo Search Algorithm

CS is an evolutionary optimization algorithm [7]; it's based on swarm intelligence. Yang and Deb, being inspired by cuckoos' natural behaviors, developed this algorithm. The algorithm mimics the behavior of some cuckoo species' obligate brood parasitism where they lay eggs in other host birds' nests [8]. Obligate brood parasitism is a special characteristics of cuckoos. As a bird, cuckoo's behavior and activity are fascinating. It's not only because of their beautiful sounds, but also due to their aggression in reproduction strategy. Mature cuckoos lay their eggs in the nests of other host birds. Basically, this algorithm is based on how cuckoos lay eggs and breed. If the host birds find out that the eggs aren't theirs, they will either throw away these unknown eggs or leave their nests and build up new nests somewhere else.

The CS follows 3 important rules:

1. Each of the cuckoos lays an egg at a time and drops the egg in a nest chosen randomly.
2. The nest having high-quality eggs is the best one. This nest carries over to the next stage or generation.
3. There's a fixed number of available nests of host birds. The probability Pa Ɛ [0, 1] determines whether the host bird will discover the cuckoo's laid egg or not [9].

3 Related Works and Applications

This section discusses the existing works of JA and CS. JA and CS have been implemented in various fields and applications.

3.1 Related Works and Applications of Jaya Algorithm

Jaya algorithm (JA) has a wide range of applications. Researchers have implemented the algorithm for adaptive control for the problem of surge tank indirectly. They have scrutinized the adaptive controls based on JA for the surge tanks' nonlinear models and components having nonlinearities. Usually, the formulations of the controllers are single objective optimizing problems with different controlling variables to find out the optimal solutions for satisfying JA's different constraints. This approach ensures improvement in randomness and performance of systems [10].

JA is an effective option to extract different PV model parameters [11]. This process controls and tracks maximum power point on photovoltaic systems, and also identifies reliable and accurate model parameters of PV modules and cells. This process uses an improved JA that quantifies the individual functioning in the population. Each individual selects various evolution strategies depending on probability. The strategies are designed for exploitation abilities and balancing exploration for the search process. The quantified performance is for constructing the searching direction by selecting the exemplar. This process ensures an improved population for exploring better solutions; for this, around the present best solution, it introduces a perturbation mechanism, which is self-adaptive and chaotic.

JA can be implemented for load balancing in the cloud. Cloud computing has various challenges like automated resource provisioning, server consolidation, event content dissemination, security, virtual machine migration, etc. For load balancing in the cloud, the challenges are decreasing response time, decreasing service request time of data center, improving the system's overall performance, etc. JA uses less controlling parameters and ensures a very good optimized result [12]. This approach has been proved to have great efficiency while comparing with other approaches.

Researchers have used the JA to perform fuzzy analysis using approaches based on pressure on benchmark networks [13]. Using fuzzy analysis, we can understand the uncertainty in different independent parameters of the network of water distribution. The parameters include pipe roughness values, nodal demands, pipe diameters, reservoir heads, etc. Obtaining the dependent parameters' membership functions are based on considering uncertain independent parameters' membership functions. According to the method of Impact Table, there's supposed to be a repetitive analysis due to the monotonous relationship between independent and dependent parameters. Methods based on optimization are more useful for fuzzy analysis when there's a non-monotonous relationship between independent and dependent parameters. JA has been found to be an efficient option for optimization. The analysis can be done by using a hydraulic model in EPANET, linked up with MATLAB for optimization.

JA can be implemented for tuning PID controllers for DC servo motor's position control [14]. Here, the unit step input's integral of squared error or ISE is the performance index. Using JA, ISE is minimized for obtaining the controller settings. Tuning based on JA ensures satisfactory response.

3.2 Related Works and Applications of Cuckoo Search Algorithm

Cuckoo Search (CS) based applications have shown very good efficiency in solving optimization problems. This algorithm provides better solutions as compared to many other algorithms.

It's possible to implement enhanced scatter search algorithms using CS [15]. An example is the problem of traveling salesman using improved and original scatter search. The improved edition of the scatter search algorithm performs better than the original one.

CS is an efficient algorithm for solving nurse schedule problems. Lim Huai Tein used this algorithm for nurse scheduling, which is very useful in healthcare institutions. Also, with the CS, we can solve problems in manufacturing optimization. CS has been effective in optimizing machine parameters in operations, and has provided better solutions as compared to other algorithms [16].

Quantum Inspired CS is an improved CS developed by A. Layeb [17]. This method is based on CS and Principles of Quantum Computing. The process includes defining the algorithm with a proper representation scheme, allowing the application of the algorithm on combinatorial optimizing tasks and some principles of quantum computing, such as measurement, qubit state superposition, interference, representation, etc. Representing the quantum solutions with a probability, it's possible to code the solutions. This improved algorithm ensures efficient and optimal solutions with the number of iterations and population size.

Another application of the CS is a flow of OP-AMP optimization assisted by a polynomial metamodel of 3 steps [18]. This improved algorithm provides solutions to the issues of inefficient system performance of optimized OP-AMPs. The CS provides the desired optimization results, ensuring a design flow for OP-AMP optimization. The process estimates the performance of OP-AMP; for this, it generates extremely accurate and ultra-fast polynomiameta models and facilitates quick time-domain simulating system of a metamacromodel of OP-AMP. These are integrated into a module of Verilog-AMS.

The particle approach based on CS helps to achieve energy efficiency in multimodal objective functions and wireless sensor networks [19]. This approach formulates network optimization. This process randomly deploys the nodes, and organizes those as static clusters using CS. The collection and aggregation of information are done after the selection of cluster heads. Then by the generalized algorithm of particle approach, this process forwards the information to the base station. CS helps to select cluster heads and form clusters among sensor nodes. This approach provides comparable results as compared to simulation results of LEACH protocols. Sensor network's longevity increases due to this protocol. Also, complications in chain formation reduces due to this approach.

4 Methodologies

4.1 Implementing Jaya Algorithm

While implementing the JA, the process is based on searching for the best solution and avoiding the worst solution for a certain problem. In the process, there are basic parameters, such as the size of the population, termination condition, number of design variables, etc. The termination condition is usually the maximum number of iteration. Maximizing or minimizing an objective function f(x) is the primary objective of this algorithm [20]. Suppose we have m design variables at i^{th} iteration. Also, there are n candidate solutions. The best-obtained candidate solution for f(x) from all candidate solutions is represented as f(x)best. Similarly, we represent the worst candidate solution as f(x)worst. We use $X_{j,k,i}$ to represent the j^{th} designing variable for k^{th} candidate at iteration number i. The modification of $X_{j,k,i}$ is $X'_{j,k,i}$. We do this modification by the following equation:

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i} (X_{j,best,i} - |X_{j,k,i}|) - r_{2,j,i} (X_{j,worst,i} - |X_{j,k,i}|) \quad (1)$$

In the equation, $X_{j,best,i}$ represents j variable's value for the best candidate, and $X_{j,worst,i}$ represents j variable's value for the worst candidate. $r_{1,j,i}$ and $r_{2,j,i}$ are two random numbers ranging from 0 to 1 [21]. With this equation, the search moves towards the best solution. Depending on which one is bigger between $X'_{j,k,i}$ and $X_{j,k,i}$, we update the solution. The process continues as a loop. As the iteration keeps moving forward, the solution becomes more and more optimum.

4.2 Implementing Cuckoo Search Algorithm

The CS has some stages. Usually, we begin with an objective function f(x). Then initial population is generated having n host nests xi (i=1,2,….,n). The search continues till the maximum generation. In each generation, a cuckoo is selected randomly by lévy flight. We evaluate the fitness or quality of the cuckoo. We choose a nest randomly. Then we compare the fitnesses of cuckoo and nest and replace the one having lower fitness [22].

While generating new solution $x^{(t+1)}$ for, suppose cuckoo i, we can use the following equation,

$$x^{(t+1)} = x^{(t)} + \alpha \oplus \text{Lévy}(\beta) \quad (2)$$

We use the product $\oplus$ meaning entry-wise walk during multiplications. There's esentially a random walk due to the Lévy flights [23]. We draw random steps of Lévy flights from a distribution of Lévy for large steps.

$$\text{Lévy} \sim u = t^{-1-\beta} (0 < \beta \leq 2) \quad (3)$$

This has an infinite variance, and with it, there's an infinite mean. The consecutive steps or jumps of a cuckoo develop a random walk process following a power-law and step-length distribution having a heavy tail. We abandon the worst nest's fraction Pa to help to build up new nests at new locations by randomly walking and mixing. The eggs or solutions get mixed by the random permutation depending on the difference or similarity to the host eggs. The step size sample generation isn't trivial using the Lévy flights. A simple scheme can be shown as:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy}(\beta) \sim 0.01 (u / |v|) (x_i^{(t)} - x_b^{(t)}) \quad (4)$$

Here, we have drawn u and v from normal distribution.

$$u = N(0, \sigma_u^2) \quad (5)$$

$$v = N(0, \sigma_v^2) \quad (6)$$

$$\sigma_u = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\frac{\Gamma\left[\frac{1+\beta}{2}\right]\beta 2(\beta-1)}{2}}\right\}^{1/\beta} \qquad (7)$$

$$\sigma_v = 1 \qquad (8)$$

Here, we have used Γ as the standard Gamma function.

5 Tests and Results

All tests of the functions have been done on Intellij IDEA Community Edition 2020.2.3. The tests were performed on Intel(R) Core(TM) i5-8250U (1.60 GHz, 3.4 GHz) with 8 GB of DDR4 RAM on Windows 10 operating system. We have done the minimization of the functions and updated the minimized values and time performances in milliseconds. Each test has been done with 1000 iterations.

For JA, there was no need for any fixed parameter values. After taking the initial population, the process had moved towards the next iterations, where the population was updated using equation (1). For CS, there were some fixed parameter values. Probability, Pa = 0.25. σ_u = 0.6969 was used based on previous study[24]. The parameters varied in the iterations.

We have used 11 functions for the tests as shown in table 1.

Table 1. Benchmark functions

No.	Functions	Conditions
1.	$f_1(x) = x_1^2 - x_1x_2 + x_2^2 + 2x_1 + 4x_2 + 3$	$-100 \le x_1, x_2 \le 100$
2.	$f_2(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$-100 \le x_1, x_2 \le 100$
3.	$f_3(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	$-5 \le x_1, x_2 \le 5$
4.	$f_4(x) = \lvert x_1^2 + x_2^2 - 2x_1x_2\rvert + \lvert\sin x_1\rvert + \lvert\cos x_2\rvert$	$-500 \le x_1, x_2 \le 500$
5.	$f_5(x) = 10x^4 - 8x^2 + 12x + 16$	$-100 \le x \le 100$
6.	$f_6(x) = 17x^5 - 11x^2 + 11x + 6$	$-100 \le x \le 100$
7.	$f_7(x) = 3x_1^2 + 5x_2^2 - 0.6\cos(3\pi x_1+4\pi x_2) + 4$	$-5 \le x_1,x_2 \le 5$
8.	$f_8(x) = 3x_1^2 + 7x_2^2 + 15(\sin^2 x_1+\sin^2 x_2)$	$-100 \le x_1,x_2 \le 100$
9.	$f_9(x) = (2.7 - x_1 + x_1x_2)^2 + (1.85 - x_1 + x_1x_2^2)^2 + (3.1 - x_1 + x_1x_2^3)^2$	$-100 \le x_1,x_2 \le 100$
10.	$f_{10}(x) = (x_1^2 + x_2 - 19)^2 + (3x_1 + x_2^2 - 16)^2$	$-100 \le x_1,x_2 \le 100$
11.	$f_{11}(x) = 10^4x_1^2 + x_2^2 - (x_1^2 + x_2^2) + \{10^{-4}(x_1^2 + x_2^2)\}$	$-100 \le x_1,x_2 \le 100$

Table 2. Results for Jaya Algorithm for minimized problems

No.	Best Results	Worst	Mean	Time (ms)
f_1	-6.33	33.2	-2.934	53.6
f_2	0	0	0	59
f_3	0	4.3	0.891	56.3
f_4	1.00	1.41	1.116	64
f_5	5.23	5.23	5.23	54.3
f_6	-1.699×10^{11}	-1.677×10^{11}	-1.692×10^{11}	74.9
f_7	3.4	3.45	3.407	63.46
f_8	0	0.01	0.001	65.55
f_9	0.68	2.88	2.265	83.67
f_{10}	0	1.02	0.104	59.1
f_{11}	0	0.84	0.107	44.1

Table 3. Results for Cuckoo Search Algorithm for minimized problems

No.	Best Results	Worst	Mean	Time (ms)
f_1	-6.33	-6.33	-6.33	33.5
f_2	0	0	0	35.1
f_3	0	0	0	32.9
f_4	1.00	1.15	1.055	151.5
f_5	5.23	5.23	5.23	29.6
f_6	-1.7×10^{11}	-1.7×10^{11}	-1.7×10^{11}	672391
f_7	3.4	3.4	3.4	40.85
f_8	0	0	0	37.5
f_9	0	0.68	0.612	99.3
f_{10}	0	0	0	40.5
f_{11}	0	0	0	26.5

Table 2 and Table 3 show the test obtained results. Each test was run 20 times and the best-minimized value, the worst minimized value, mean value, and the execution time were reported. The execution time of the two algorithms are shown in the tables. Also, **Fig. 1** and **Fig. 2** show the comparison of the two algorithms in terms of mean values and execution times for the functions.

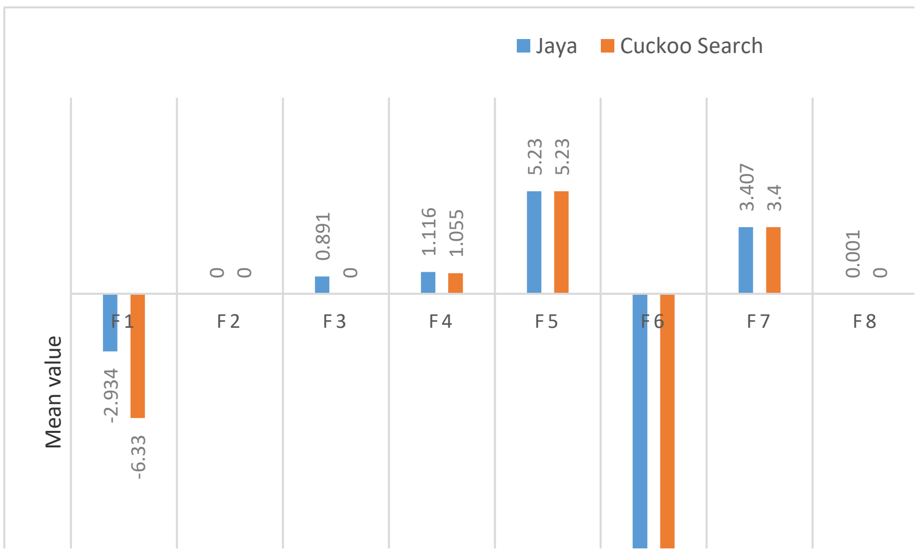


Fig. 1. Performance Comparison between Jaya algorithm and Cuckoo Search in term of test size

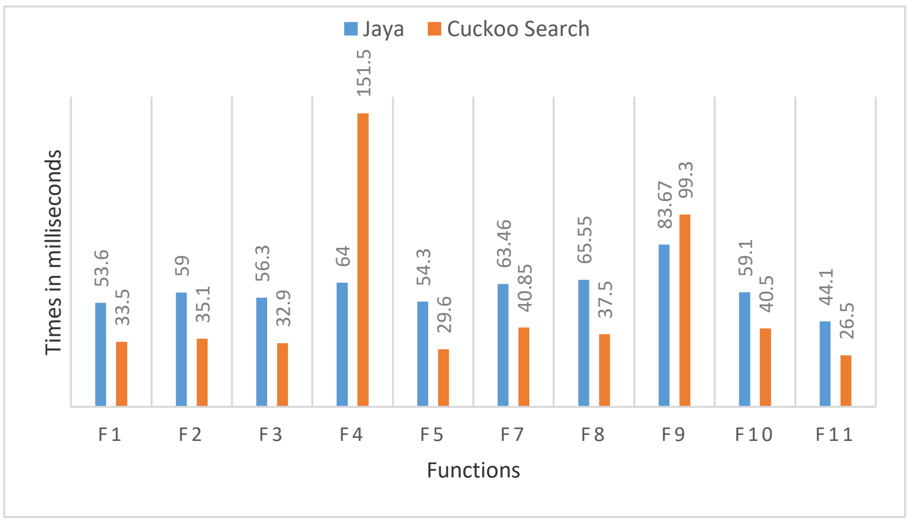
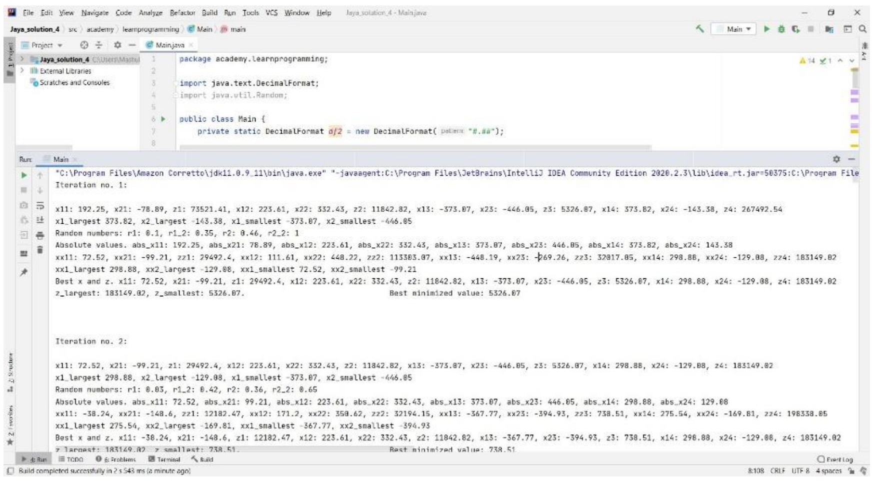


Fig. 2. Execution time comparison between Jaya algorithm and Cuckoo Search.

Fig. 3 shows the screenshot of a Cuckoo Search program done in Intellij IDEA Community Edition 2020.2.3. The coding of the algorithms has been done in Java.



```java
package academy.learnprogramming;

import java.text.DecimalFormat;
import java.util.Random;

public class Main {
    private static DecimalFormat df2 = new DecimalFormat( pattern: "#.##");
```

```
"C:\Program Files\Amazon Corretto\jdk11.0.9_11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.3\lib\idea_rt.jar=50375:C:\Program File
Iteration no. 1:

x11: 192.25, x21: -78.89, z1: 73521.41, x12: 223.61, x22: 332.43, z2: 11842.82, x13: -373.07, x23: -446.05, z3: 5326.07, x14: 373.82, x24: -143.38, z4: 267492.54
x1_largest 373.82, x2_largest -143.38, x1_smallest -373.07, x2_smallest -446.05
Random numbers: r1: 0.1, r1_2: 0.35, r2: 0.46, r2_2: 1
Absolute values. abs_x11: 192.25, abs_x21: 78.89, abs_x12: 223.61, abs_x22: 332.43, abs_x13: 373.07, abs_x23: 446.05, abs_x14: 373.82, abs_x24: 143.38
xx11: 72.52, xx21: -99.21, zz1: 29492.4, xx12: 111.61, xx22: 448.22, zz2: 113303.07, xx13: -448.19, xx23: -269.26, zz3: 32017.05, xx14: 298.88, xx24: -129.08, zz4: 183149.02
xx1_largest 298.88, xx2_largest -129.08, xx1_smallest 72.52, xx2_smallest -99.21
Best x and z. x11: 72.52, x21: -99.21, z1: 29492.4, x12: 223.61, x22: 332.43, z2: 11842.82, x13: -373.07, x23: -446.05, z3: 5326.07, x14: 298.88, x24: -129.08, z4: 183149.02
z_largest: 183149.02, z_smallest: 5326.07.                              Best minimized value: 5326.07



Iteration no. 2:

x11: 72.52, x21: -99.21, z1: 29492.4, x12: 223.61, x22: 332.43, z2: 11842.82, x13: -373.07, x23: -446.05, z3: 5326.07, x14: 298.88, x24: -129.08, z4: 183149.02
x1_largest 298.88, x2_largest -129.08, x1_smallest -373.07, x2_smallest -446.05
Random numbers: r1: 0.03, r1_2: 0.42, r2: 0.36, r2_2: 0.65
Absolute values. abs_x11: 72.52, abs_x21: 99.21, abs_x12: 223.61, abs_x22: 332.43, abs_x13: 373.07, abs_x23: 446.05, abs_x14: 298.88, abs_x24: 129.08
xx11: -38.24, xx21: -148.6, zz1: 12182.47, xx12: 171.2, xx22: 350.62, zz2: 32194.15, xx13: -367.77, xx23: -394.93, zz3: 738.51, xx14: 275.54, xx24: -169.81, zz4: 198338.05
xx1_largest 275.54, xx2_largest -169.81, xx1_smallest -367.77, xx2_smallest -394.93
Best x and z. x11: -38.24, x21: -148.6, z1: 12182.47, x12: 223.61, x22: 332.43, z2: 11842.82, x13: -367.77, x23: -394.93, z3: 738.51, x14: 298.88, x24: -129.08, z4: 183149.02
z_largest: 183149.02, z_smallest: 738.51.                               Best minimized value: 738.51
```

Build completed successfully in 2 s 543 ms (a minute ago)

Fig. 3. Screenshot of the part of a program of the Cuckoo Search based function solution with output in Intellij IDEA Community Edition 2020.2.3

6 Discussions

As observed from the tests, JA has performed well consistently for all the 11 functions. CS hasn't been consistent like JA, but the mean values determined from the tests of the CS were nearer to the best-minimized values as compared to JA for many functions. As shown in table 2 and 3, for 8 of the 11 functions, CS required less execution time as compared to that of JA but the differences weren't much. For 3 functions, which are function 4 (f_4), function 6 (f_6), and function 9 (f_9), JA took less execution time as compared to that of CS. But in these cases, the time differences were larger as compared to those of other functions. In the case of function 6 (f_6), the average execution time of JA was 74.9 ms, but the average execution time of CS algorithm was 672391 ms. Here, the difference is huge. For function 4 (f4), CS takes 151.5 ms, where JA takes only 64 ms. But when JA takes larger execution time as compared to CS, the difference isn't much. There has been no case in the tests where JA took equal to or more than twice the execution time of CS, according to Fig 2. So, it indicates that CS may take much larger execution time for some functions as compared to JA. Considering this case, JA is more consistent.

CS showed better performance for 8 of the 11 functions in terms of mean values and execution times as shown in table 2 and 3, but lacked consistency as compared to JA. For function 9 (f_9), CS was better than JA, but the exection time was a little higher. Regarding execution time, in **Fig. 2**, CS's performance isn't always efficient for all functions. There are some functions, where CS performs at a much slower speed. For some functions, CS can take extremely large execution time, as shown in the case of function 6 (f_6). Regarding these issues, JA is a simpler and more consistent algorithm, providing solutions to most functions with an acceptable execution time. But regarding mean values, JA performs well but is less optimum as compared to CS. From the overall scenario, JA is more consistent due to its performances in acceptable ranges for most functions.

7 Conclusion and Future Works

This paper evaluated the performances of CS and JA using the benchmark functions in terms of mean values and execution times. Each of the algorithms has shown impressive results in many cases. There has also been complexities for both algorithms regarding execution time and mean values in some cases. In the overall scenario, JA has turned out to be a more consistent algorithm. The analysis of these 2 algorithms will help engineers to select the right algorithm for specific problems.

As for the future works, we can make a detail analysis explaining for which functions CS works well with acceptable execution times and for which functions, CS struggles to maintain a faster execution time. Also, there are works to be done with the functions, for which, CS takes much longer execution time.

ACKNOWLEDGMENTS

This research is funded by Universiti Malaysia Pahang (UMP) under grant: “Prioritized T-way Test Suite Generation Strategy Based on Chaotic Flower Pollination Algorithm”, Grant no: RDU190372. We thank UMP for the contribution and supports.

References

1. Kelley, T.R.: Optimization, an Important Stage of Engineering Design. The Technology Teacher **69**(5), 18-23 (2010).
2. Mansour, N.: Search Algorithms and Applications. (2011)
3. Yang, X.-S.: Metaheuristic Optimization: Algorithm Analysis and Open Problems. Paper presented at the International Symposium on Experimental Algorithms, Berlin. Heidelberg,
4. WK Wong, C.I.M.: A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications. Paper presented at the 2019 7th International Conference on Smart Computing & Communications (ICSCC),
5. Y. A. Alsariera, H.S.A., A. M. Nasser, M. A. Majid and K. Z. Zamli: Comparative Performance Analysis of Bat Algorithm and Bacterial Foraging Optimization Algorithm using Standard Benchmark Functions. Paper presented at the 2014 8th. Malaysian Software Engineering Conference (MySEC), Langkawi, Malaysia,
6. Willa Ariella Syafruddin, M.K., Brahim Benaissa: Does the Jaya Algorithm Really Need No Parameters? . Paper presented at the 10th International Joint Conference on Computational Intelligence,

7. Pandey, H.M.: Jaya a Novel Optimization Algorithm: What, How and Why? Paper presented at the 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence),
8. Swati Sharma, B.B.: Adaptive Control using Jaya Algorithm. In: AIP Conference Proceedings 2136, 020001 2019
9. Kunjie Yua, B.Q., Caitong Yuea, Shilei Gea, Xu Chenc, Jing Lianga: A performance-guided JAYA algorithm for parameters identification of photovoltaic cell and module. Applied Energy, Elsevier **237**, 241-257 (2019).
10. Subhadarshini Mohanty, P.K.P., Mitrabinda Ray, Subasish Mohapatra: An Approach for Load Balancing in Cloud Computing Using JAYA Algorithm. International Journal of Information Technology and Web Engineering **14**(1), 27-41 (2019).
11. Sreethu Subrahmanian, R.G.: Fuzzy node flow analysis of water distribution networks using Jaya algorithm. Paper presented at the IOP Conference Series Earth and Environmental Science 491:012010
12. ANKIT K. SAHU, J.K.B., V.P. SINGH, S.P. SINGH: JAYA ALGORITHM BASED TUNING OF PID CONTROLLER. International Journal of Industrial Electronics and Electrical Engineering **4**(12) (2016).
13. Azizah Mohamad, A.M.Z., Nor Erne Nazira Bazin, Amirmudin Udin: Cuckoo Search Algorithm for Optimization Problems - A Literature Review. Applied Mechanics and Materials **421**, 502-506 (2013).
14. Iztok Fister Jr.*, D.F., Iztok Fister: A comprehensive review of cuckoo search: variants and hybrids. International Journal Mathematical Modelling and Numerical Optimisation **4**(4), 387-409 (2013).
15. Azizah Binti Mohamad, A.M.Z., Nor Erne Nazira Bazin: Cuckoo Search Algorithm for Optimization Problems—A Literature Review and its Applications. Applied Artificial Intelligence: An International Journal **28**(5), 419-448 (2014).
16. Al-Obaidi, A.T.S.: Improved Scatter Search Using Cuckoo Search. International Journal of Advanced Research in Artificial Intelligence **2**(2) (2013).
17. Lim Huai Tein, R.R.: Recent Advancements of Nurse Scheduling Models and A Potential Path. In: Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA2010), Kedah, Malaysia 2010
18. Yildiz, A.R.: Cuckoo search algorithm for the selection of optimal machine parameters in milling operations. The International Journal of Advanced Manufacturing Technology **64**, 55-61 (2012).
19. Layeb, A.: A novel quantum-inspired cuckoo search for Knapsack problems. International Journal of Bio-Inspired Computation **3**(5) (2011).
20. G. Zheng, S.P.M.a.E.K.: Metamodel-Assisted Fast and Accurate Optimization of an OP-AMP for Biomedical Applications. Paper presented at the 2012 IEEE Computer Society Annual Symposium on VLSI, Amherst, MA, USA,
21. Al-Hakam Ayad Salih, A.H.A., Nada Yousif Hashim: Jaya: An Evolutionary Optimization Technique for Obtaining the Optimal Dthr Value of Evolving

Clustering Method (ECM). International Journal of Engineering Research and Technology **11**(12), 1901-1912 (2018).
22. Dhivya Manian, M.S.: Energy Efficient Computation of Data Fusion in Wireless Sensor Networks Using Cuckoo Based Particle Approach (CBPA). International Journal of Communications, Network and System Sciences **4**(4), 249-255 (2011).
23. Hongqing ZHENG, Y.Z.: A Novel Cuckoo Search Optimization Algorithm Base on Gauss Distribution. Journal of Computational Information Systems **8**(10), 4193-4200 (2012).
24. Nasser, A.B., Alsewari, A.R.A., Zamli, K.Z.: Tuning of cuckoo search based strategy for t-way testing. In: International Conference on Electrical and Electronic Engineering 2015, vol. 9, p. 8948. Journal of Engineering and Applied Sciences