Bima Journal of Science and Technology, Vol. 4(2) Dec, 2020 ISSN: 2536-6041

EXPERIMENTAL ANALYSIS BASED ON TIME COMPLEXITY AND SOLUTION QUALITY OF SOME SWARM INTELLIGENCE ALGORITHMS

[1*]MOHAMMED AJUJI, [1]ALIYU ABUBAKAR, [2]DATTI USENI EMMANUEL AND [3]ISMAIL HARUNA

[1]Department of Computer Science, Faculty of Science, Gombe State University, Gombe 760214, Nigeria

[2]Department of Computer Science, School of Science, Federal College of Education, Plateau Nigeria

[3]Department of Computer Science, School of Science, Federal College of Education (Tech), Gombe Nigeria

Corresponding Author: mohammedajuji@gmail.com

ABSTRACT

The most popular tool nowadays for solving multidimensional optimization problems are nature inspired algorithms. These algorithms as the name implies are inspired by nature; nonetheless, they also have their own merit and demerit. Swarm intelligence-based optimization algorithms in recent-times are gaining more popularity for their resource efficiency and effectiveness. However, randomly picking an algorithm to help solve a problem or answer a question cannot guarantee quality of solution and or time efficiency. Therefore, this paper analyzes the time-complexity and efficacy of some nature-inspired algorithms which include Artificial Bee Colony (ABC), Bat Algorithm (BA), and Particle Swarm Optimization (PSO). For each algorithm, experiments were conducted several times and comparative analyses were made. It was found out that Artificial Bee Colony (ABC) outperformed both the other algorithms (BA and PSO) in terms of quality of the solution; Particle Swarm Optimization (PSO) is time-efficient while Artificial Bee Colony yields a worst-case scenario in terms of time complexity. Bat Algorithm on the other hand presents a worst case in terms of quality of solution and average case with respect to time complexity.

Keywords: Artificial Bee Colony, Bat, Swarm Intelligence, Optimization, And Opytimizer

INTRODUCTION

More discoveries and inventions by scientists are being inspired by nature. Many nature-inspired algorithms have been developed to solve complex optimization problems(Bujok *et al*., 2019). Generally, there are two main concepts developed in bio-inspired computation: Evolutionary algorithms and Swarm based algorithms (Greco and Vanzi, 2018). This research is concerned with the former. Algorithms that are inspired by nature are referred to as nature-inspired algorithm (Chiroma *et al*., 2020). For example, Artificial Bee Colony (ABC) Algorithm is a meta-heuristic optimization algorithm based on the intelligent behavior of honeybee swarm (Jr *et al*., 2013). Bat algorithm is inspired by echolocation features of microbats (Srivastava, 2019) and Particle Swarm Optimization (PSO) inspired by the behavior of birds flocking (Prabha and Yadav, 2019)(Wang *et al*., 2018). Swarm intelligence is the collective behavior of decentralized, self-organized systems, either

natural or artificial (Jr *et al*., 2013). The most well-known classes of swarm intelligence algorithms include Particle swarm optimization, Ant Colony Optimization, Artificial Bee Colony, Firefly algorithm, Cuckoo Search, and Bat algorithm(Dai *et al*., 2018). The complexity of an algorithm as regards to the solution quality (Johansson, 2016) produced and the time it takes to converge or complete is important in optimization.

Complex algorithms require powerful computation machines to execute within a required timeframe. As such, it is necessary to figure out the most efficacious algorithm to be utilized via the use of trial and error techniques. This is because these algorithms have been accepted all through optimization; machine learning, data mining, computational intelligence, and artificial intelligence (Chiroma *et al*., 2020). These algorithms are found to be very effective and efficient in solving real-world optimization problems better than the conventional algorithms because of their ability to effectively handle highly nonlinear and complex problems especially in science and engineering (Akay and Karaboga, 2012). Therefore, the study in this paper presents a comparative analysis of three randomly selected swarm based optimization algorithms: Artificial Bee Colony, Bat Algorithms, and Particle Swarm Optimization for both time complexity and quality of solution using Opytimizer (Rosa and Papa, 2020) framework. This is because Opytimizer is relatively new and no similar study was carried out yet that used it for similar analysis.

The paper is structured as follows: Section 2 presents the basic concepts of the algorithms under study; ABC, BA, and PSO. Section 3 discusses the methodology used to collect literature and conduct the experiment. Section 4 presents results and discussion and finally, the conclusion is drawn in section 5.

Basic Concept of the Algorithms

This section discusses the basic theoretical background of the ABC, BA, and PSO algorithms. (Chiroma *et al*., 2016) The soul of the computer is an algorithm; which is a sequence-of-steps or procedures designed to solve a problem or help answer a question. Yet, there is no universally accepted definition for an algorithm. But in simple terms, an algorithm is a set of procedure that generates results for the given input(s) (Greco and Vanzi, 2018).

Artificial Bee Colony

Artificial Bee Colony simulates the foraging process of natural honey bees (Saadi *et al*., 2016) (Figure 1 and 2). The bee colony family in ABC consists of three members: employed, onlooker, and scout bees. Scout bees' initiates searching of food sources randomly, once the potential food sources are identified by a scout, they become employed bees. Then food sources are exploited by employed bees that also share the information about the quality and quantity of food sources to the onlooker (bees resting at the hive and waiting for the information from employed bees). Specific waggle dance is performed to share food information(Ma and Wang, 2019) and (Alzaqebah and Abdullah, 2014).

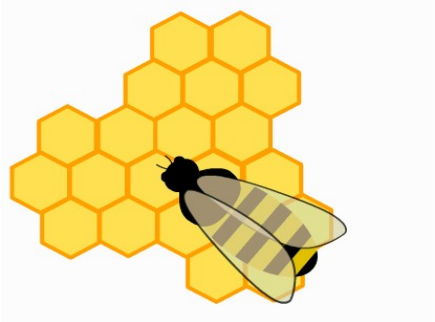

Figure 1: Bee colony

The structure of the ABC algorithm consists of steps as follows:

- Initialization of random food sources
- Employed bee process
- Onlooker bee process
- Scout bee process

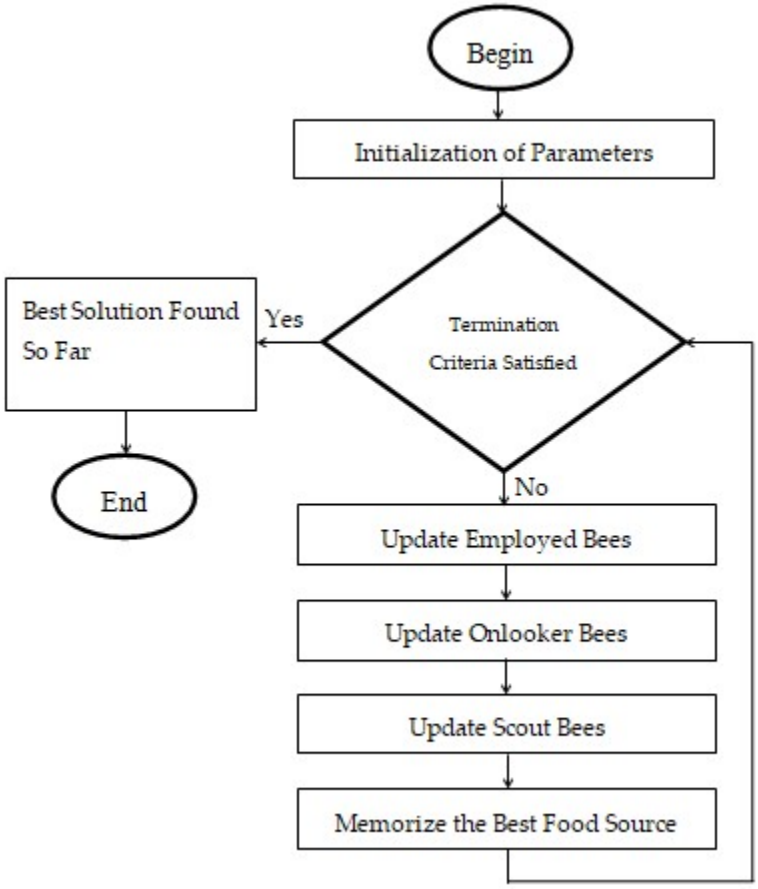

Figure 2: Flowchartof Artificial Bee Colony

Begin
Initialize the bat Population ()
While remain iterations **do**
Select sites for the local search.
Recruit bees for the selected sites and to evaluate fitness.
Select the bee with the best fitness.
Assign the remaining bees to looking for randomly.
Evaluate the fitness of remaining bees.
Update Optimum ()
EndWhile
Return Best Solution
End
Algorithm 1: Pseudocode of ABC

Bat Algorithm

Bat algorithm (BA) was developed based on the echolocation features of microbats (Srivastava, 2019) and (Nakamura *et al*., n.d.), and BA uses a frequency-tuning technique to increase the diversity of the solutions in the population, while at the same time, it uses the automatic zooming to try to balance exploration and exploitation during the search process by mimicking the variations of pulse emission rates and

loudness of bats when searching for prey. BA can deal with both continuous optimization and discrete optimization problems (Rosa and Papa, 2020). Bat algorithm has the advantage of simplicity and flexibility. BA is easy to implement, and such a simple algorithm can be very flexible to solve a wide range of problems (Yang and Karamanoglu, 2013).

The bat algorithm has three idealized rules:
1. All bats use echolocation to sense distance, and they also `know' the difference between food/prey and background barriers in some magical way;

2. Bats y randomly with velocity v_i at position x_i with a frequency f (or wavelength) and loudness A0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r 2 [0; 1], depending on the proximity of their target;

3. Although the loudness can vary in many ways, it was assumed that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{min.}$

Objective function Obj(X), X=[x 1 , x2 ,…,xnv]T

Begin
Initialize the bat population x i and v i (i=1,2,..,n)
Define pulse frequency of f i at x i
Initialize pulse rates r i and the loudness A i

While (t<maximum number of iterations)
Generate new solutions by adjusting frequency and update velocities and positions (equation 1, 2 and 3)
    If (rand >r i)
        Select a solution among the best solutions randomly;
        Generate a local solution around the selected best solution by a local random walk (equation 4)
    End if
    If (rand & lt;A i and f(xi) <f(xcgbest))
        Accept the new solution
        Increases r i and decrease a i
    End if
Rank the bats at each iteration and store their current global best xcg best
End while
    Post processing the results
End

Algorithm 2: Pseudocode of Bat algorithm

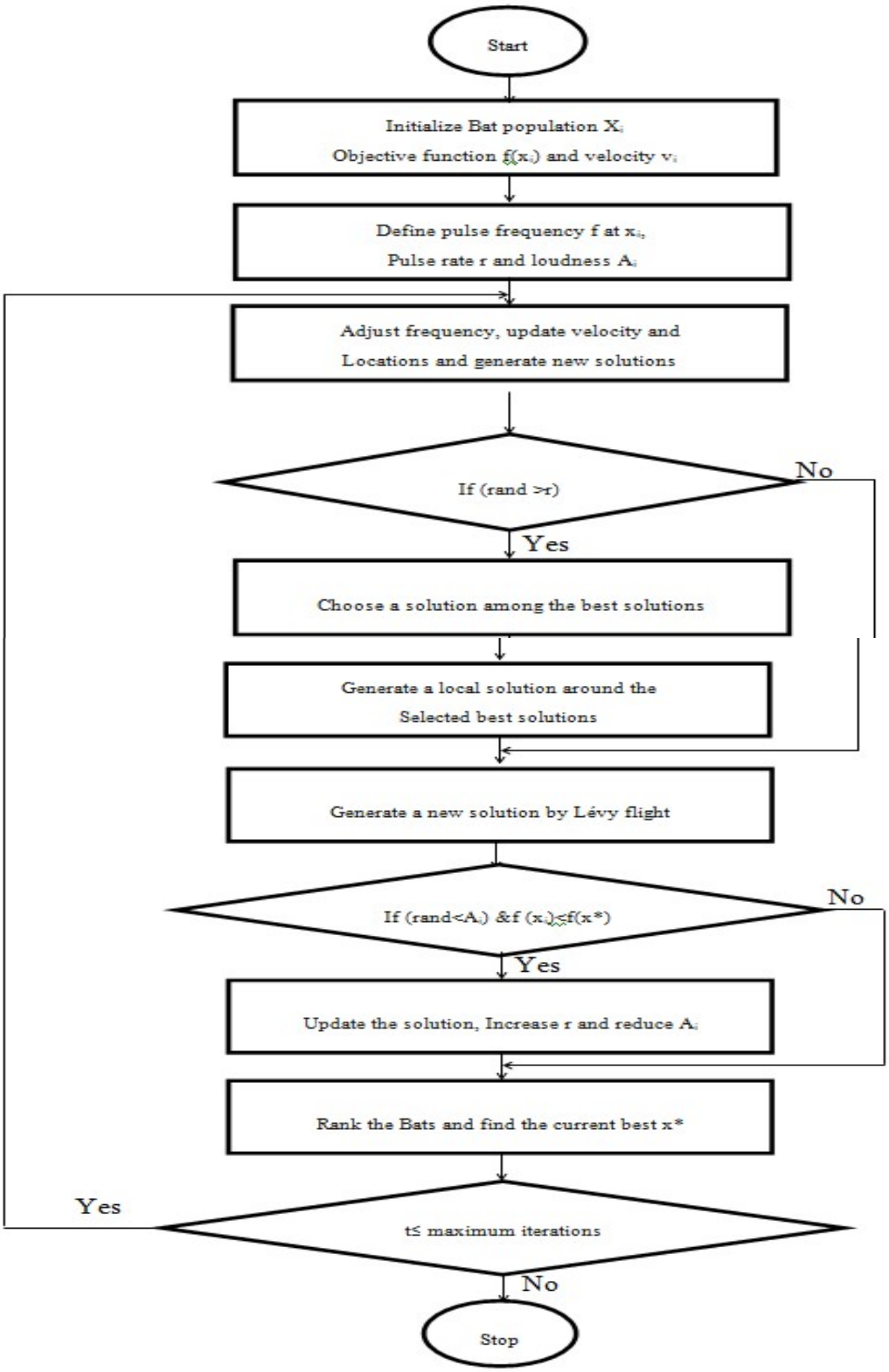

Figure 3: Flowchart of Bat Algorithm

Particle Swarm Optimization

Particle swarm optimization (PSO); it is originated from the analysis of the behavior of birds catching food (Kulkarni and Desai, 2016), American scholars Kennedy and Eberhart found during their analysis, that the flying birds often scattered, concentrated, or changed directions in an instant, adjusting their flight – usually unexpected fact. After summarizing the rules, they found that the flying pace of the whole flock of birds would generally keep consistent, and a proper distance was maintained between each bird. Through analyzing constantly the behavior of other social animals, such as birds, fishes, ants, and so on, they concluded that, in the behavior rules of social animals, there has been an invisible information sharing platform for those seemingly unstructured and dispersed biological groups. Inspired by this, scholars simulated the behavior of birds constantly and proposed the concept of optimization (Wang *et al*., 2018).

Particle swarm optimization has become a better-developed optimization, in recent years. It searches the optimal solution through continuous iteration, and it finally employs the size of the value of the objective function, or the function to be optimized (also known as the fitness function in the particle swarm), to evaluate the quality of the solution (Hudaib and Hwaitat, 2018).

To ease research, birds are considered as particles of life without mass and volume in the algorithm. The algorithm initializes the position of each particle into the solution of problems to be optimized. (Kulkarni and Desai, 2016)and (Dai *et al*., 2018) In the movement process of the particle swarm, information is conveyed between each influencing the others, and a particle's moving state is influenced by the speed and direction of its colleagues, and of the whole particle swarm, so that each particle adjusts its speed and direction according to the historical optimal positions of itself and its colleagues, and keeps flying and searching for the optimal position – the optimal solution. In the process of flying, particles update their position and direction according to their and external information; this has proved that the particle has the memory function, and particles with good positions and directions have the tendency to approach the optimal solution. As such, optimization is done through competition and cooperation between particles (Chiroma *et al*., 2016).

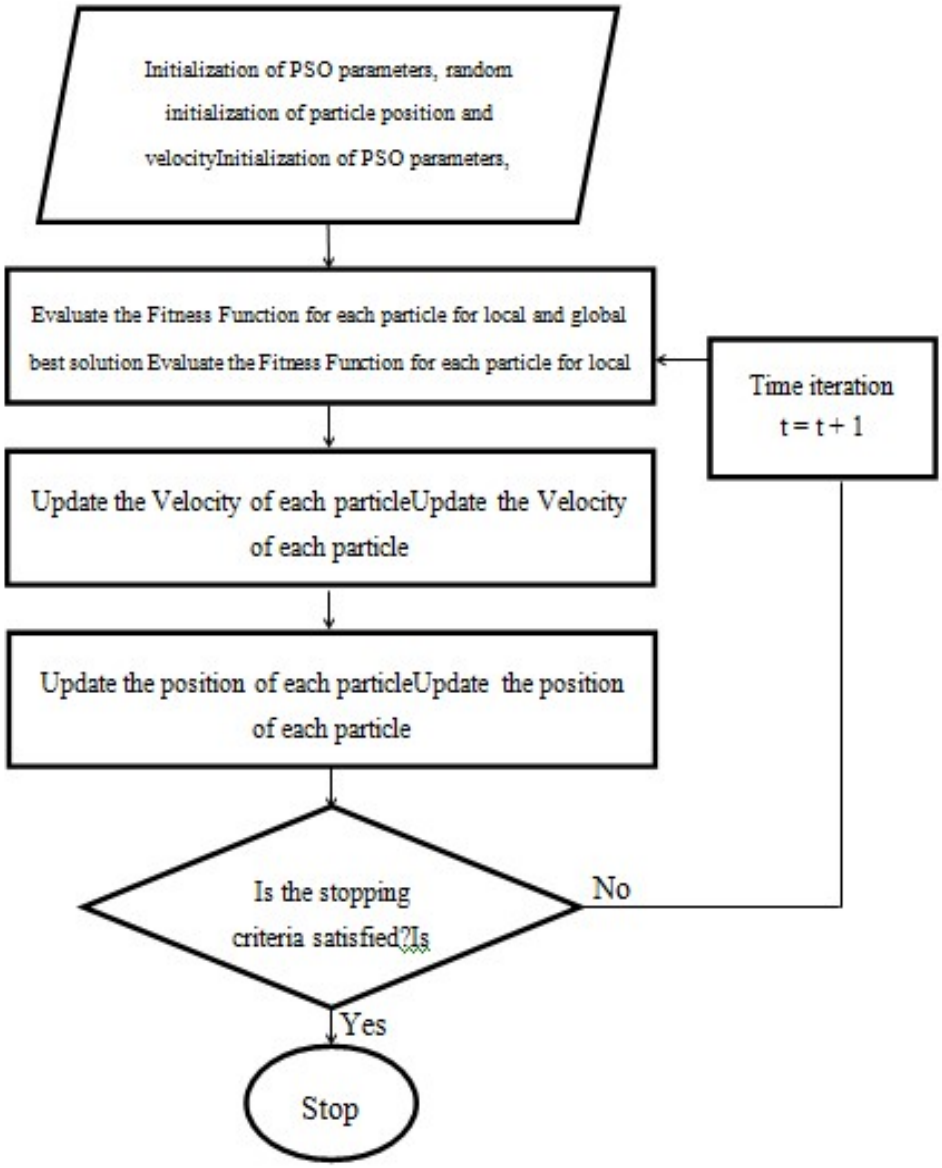

Figure 4: Flowchart of Particle Swarm Optimization

Algorithm

```
For each particle
{
   Initialize particle
}
Do until maximum iterations or minimum error criteria
{
   For each particle
   {
      Calculate Data fitness value
      If the fitness value is better than pBest
      {
         Set pBest = current fitness value
      }
      If pBest is better than gBest
      {
         Set gBest = pBest
      }
   }
      For each particle
   {
      Calculate particle Velocity
      Use gBest and Velocity to update particle Data
   }
```

MATERIALS AND METHODS

Opytimizer python micro-framework was downloaded and installed and run on visual studio code. A program was designed with a single objective function to test each algorithm separately. After running an algorithm for the required number of times we switched to another by changing only the function that calls a particular algorithm. The program is made up of the same number of agents, decision variables, the number of

iterations and upper and lower bounds to compose the search space, a mathematical function to be optimized, and an optimizer, which is the meta-heuristic technique used to perform the optimization process. Furthermore, they are bundled to an Optimizer class, which holds all the vital information about the optimization task. In the end, we run the program, and when it completes, it returns a history object which encodes valuable data which includes iteration count, fitness, position, and time taken to complete the experiment about the optimization procedure. The experiment is repeated twenty times with twenty iterations. The input variables and the number of iterations remain the same for all the algorithms throughout the experiments in other to easily compare the algorithms and get a better result.

For each iteration count in an experiment, the fitness and position are recorded in a table. The time taken for each experiment to complete is recorded for all the algorithms, which was used to analyze and determine the most time-efficient among the three algorithms. The results obtained from the procedure above are presented in the later section.

System configuration

The experiment was coded and implemented in a python micro-framework – Opytimizer. Visual Studio Code was used to code and run the experiments on a computer with the following configurations. Intel(R) Pentium(R), CPU N3540 @ 2.16GHz, RAM 4.00 GB, 64-bit operating system, x64-based processor. All experiments are performed on the same computer/machine.

RESULTS SND DISCUSSION

Results obtained from the experiments have been recorded, F1 – F20 represents the number of experiments from first to last. For each of the experiments; the convergence time, best, average, and worst cases were recorded, analyzed, and presented in figures (5, 6, 7 and 8); where x-axis and y-axis represents algorithms and experiments respectively.

The statistical summary of the result shows that with respect to time complexity; PSO outperforms BA and ABC. In other words, PSO converges faster.PSO can be used in time-sensitive applications where the quick result is expected (Tanveer *et al*., 2016)[35]. BA also has a good or average convergence time compare to ABC. ABC cannot be a good choice for time-sensitive applications.

In figure 6, the best cases of the algorithms were compared and it was found that the ABC algorithm which appeared to be time-inefficient has outperformed both PSO and BA as regards quality of solution. That means if application concern require quality solution than convergence speed then ABC is an excellent choice. BA cannot be used in the solution-quality-sensitive applications where the quality result is expected.

Both figures 7 and 8 compared the average and worst cases of the algorithms. The summary of the results revealed that ABC performs excellently well compare to the other two algorithms; BA and PSO. BA is worst whereas PSO is average.

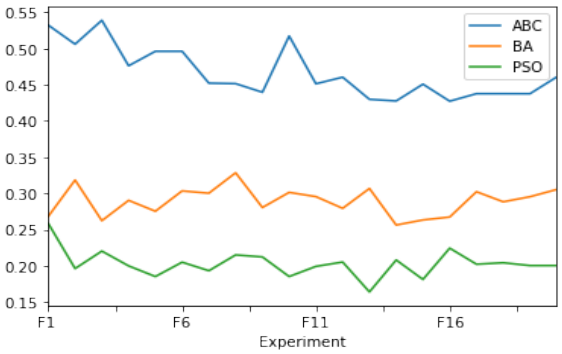

Figure 5: Running time

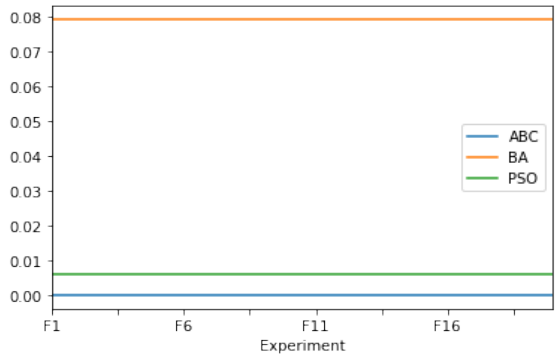

Figure 6: Comparison of best cases

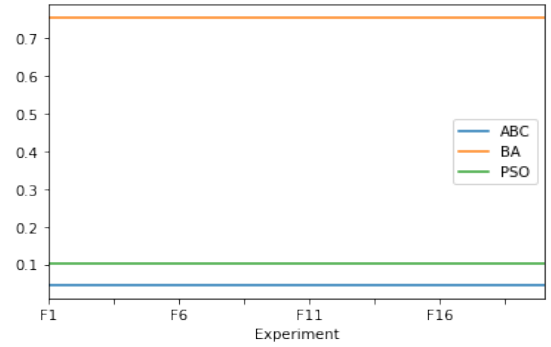

Figure 7: Comparison of average cases

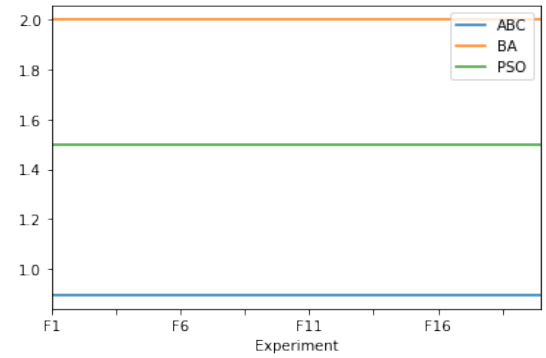

Figure 8: Comparison of worst cases

The above results are summarized and presented in the following tables.

Table 1: Summary of result (best, average, and worst cases)

	Best	**Average**	**Worst**
ABC	First	First	First
BA	Third	Third	Third
PSO	Second	Second	Second

Table 2: Time complexity

	Time complexity
ABC	Third
BA	Second
PSO	First

From the study we conducted, we found that ABC has the best solution quality even though it is worst in terms of time complexity. PSO appears to have the best time complexity but its solution has no equality with that of ABC concerning the quality of the solution. BA has the worst solution quality but it outperforms ABC in terms of convergence speed.

This study could be used to guide researchers who want to use any of the algorithms to choose easily amongst the three. Depending on the need for the research, some may be interested in the quality of the solution, not time complexity; hence ABC is the right choice whereas if speed is of much concern, then PSO is the best(Srivastava, 2019).

CONCLUSION

The performances of ABC, BA and PSO swarm based algorithms based on solution quality and time-complexity have been investigated using Opytimizer: A Nature-Inspired Python Optimizer to determine/measure the performance of three nature-inspired algorithms; particle swarm optimization, bat algorithm, and artificial

bee colony. The experiment was run several times and the convergence speed, mean, best, and worst cases were recorded for analysis. The study revealed that ABC has the best solution quality even though it is worst in terms of time complexity. PSO is time-efficient but its solution has no equality with that of ABC as regards quality of the solution. On the other hand, BA has the worst solution quality but it outperforms ABC in terms of faster convergence.

REFERENCES

Akay, B., and Karaboga, D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142.

Alzaqebah, M., and Abdullah, S. (2014). *Artificial bee colony search algorithm for examination timetabling problems*. *June*.

Bujok, P., Tvrdík, J., and Poláková, R. (2019). Comparison of nature-inspired population-based algorithms on continuous optimisation problems. *Swarm and Evolutionary Computation*, *50*(January), 100490.

Chiroma, H., Khan, A., Abubakar, A. I., Saadi, Y., Hamza, M. F., Shuib, L., Gital, A. Y., and Herawan, T. (2016). A new approach for forecasting OPEC petroleum consumption based on neural network train by using flower pollination algorithm. *Applied Soft Computing Journal*, *48*, 50–58.

Chiroma, H., Rana, N., and Muhammad, A. N. (2020). *Nature Inspired Meta-heuristic Algorithms for Deep Learning : Recent Progress and Novel Perspective Nature Inspired Meta-Heuristic Algorithms for Deep Learning : Recent Progress and Novel Perspective*. *May 2019*, 0–13.

Dai, H., Chen, D., and Zheng, Z. (2018). *Effects of Random Values for Particle Swarm Optimization Algorithm*. 1–20.

Greco, R., and Vanzi, I. (2018). ScienceDirect New few parameters differential evolution algorithm with application to structural identification. *Journal of Traffic and Transportation Engineering (English Edition)*, 1–14.

Hudaib, A. A., and Hwaitat, A. K. Al. (2018). *Movement Particle Swarm Optimization Algorithm*. *12*(1).

Johansson, E. (2016). *Understanding Solution Quality* (Issue 1769).

Jr, I. F., Yang, X., Fister, I., and Brest, J. (2013). *A Brief Review of Nature-Inspired Algorithms for*. *80*(3), 1–7.

J. N. Tripathi, J. Mukherjee, R. K. Nagpal, and R. Malik, "Application of nature-inspired algorithms in power delivery network design: An industrial case study,"*2014 IEEE 23rd Conference on Electrical Performance of Electronic Packaging and Systems*, Portland, OR, 2014, pp. 103-106.

Kulkarni, V., and Desai, V. V. (2016). *ABC and PSO : A Comparative Analysis*. *December 2017*.

Lones, M. A. (2020). Mitigating Metaphors : A Comprehensible Guide to Recent Nature - Inspired Algorithms. *SN Computer Science*, 1–12.

Ma, L., and Wang, X. (2019). *A novel artificial bee colony optimiser with dynamic population size for multi-level threshold image segmentation Hai Shen Min Huang*. *13*(1), 32–44.

Nakamura, R. Y. M., Pereira, L. A. M., Costa, K. A., Rodrigues, D., and Papa, J. P. (n.d.). *BBA : A Binary Bat Algorithm for Feature Selection*.

Prabha, S., and Yadav, R. (2019). Engineering Science and Technology , an International Journal Differential evolution with biological-based mutation operator. *Engineering Science*

and Technology, an International Journal, *xxxx*.

Rosa, G. H. De, and Papa, J. P. (2020). *O : a n -i p o*. 1–17.

Saadi, Y., Tri, I., Yanto, R., Herawan, T., and Balakrishnan, V. (2016). *Ringed Seal Search for Global Optimization via a Sensitive Search Model*. 1–31.

Srivastava, S. (2019). *Application of Bat Algorithm for Transport*. *2019*(Mc).

Tanveer, S. R., Islam, J., and Mah, A. (2016). *A Comparative Study on Prominent Swarm Intelligence Methods for Function Optimization Technology and Optimization*. *7*(3).

Wang, L., Liu, X., Sun, M., Qu, J., and Wei, Y. (2018). *A New Chaotic Starling Particle Swarm Optimization Algorithm*. *2018*.

Yang, X., and Karamanoglu, M. (2013). *1 Swarm Intelligence and Bio-Inspired Computation : An Overview*.