Article

Experimental analysis of time complexity and solution quality of swarm intelligence algorithm

Mohammed Ajuji [1]*, Aliyu Abubakar [1] and Datti, Useni Emmanuel [2]

[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

aaliyu@gsu.edu.ng (A.A.), duedatti@yahoo.com (D.U.E.)

* Correspondence: mohammedajuji@gmail.com, majuji@gsu.edu.ng; Tel.: +2347065539970

Abstract: Nature-inspired algorithms are very popular tools for solving optimization problems inspired by nature. However, there is no guarantee that optimal solution can be obtained using a randomly selected algorithm. As such, the problem can be addressed using trial and error via the use of different optimization algorithms. Therefore, the proposed study in this paper analyzes the time-complexity and efficacy of some nature-inspired algorithms which includes Artificial Bee Colony, Bat Algorithm and Particle Swarm Optimization. For each algorithm used, experiments were conducted several times with iterations and comparative analysis was made. The result obtained shows that Artificial Bee Colony outperformed other algorithms in terms of the quality of the solution, Particle Swarm Optimization is time efficient while Artificial Bee Colony yield a worst case scenario in terms of time complexity.

Keywords: artificial bee colony, bat, particle swarm, optimization and Opytimizer

1. Introduction

Recently, more discoveries and inventions by scientists are been inspired by nature. Many nature-inspired algorithms have been developed to solve complex optimization problems. Generally, there are two main concepts developed in bio-inspired computation: Evolutionary algorithms and Swarm based algorithms. This research is concerned with the former. Algorithms that are inspired by nature are referred to as nature-inspired algorithm [1]. For example Artificial Bee Colony (ABC) Algorithm is a meta-heuristic optimization algorithm based on the intelligent behavior of honeybee swarm [2]. Bat algorithm is inspired by echolocation features of microbats [3] and Particle Swarm Optimization (PSO) inspired by the behavior of birds flocking [4, 5].

Swarm intelligence is the collective behavior of decentralized, self-organized systems, either natural or artificial [2]. 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. The complexity of each algorithm with regards to quality of the solution produced and the time is important consideration. Algorithms that are more complex 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 technique. This is because these algorithms have been accepted all through optimization; machine learning, data mining, computational intelligence and artificial intelligence. The 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 [6]. Therefore, the study in this paper presents a comparative analysis of three randomly selected algorithms: Artificial Bee Colony, Bat Algorithms and Particle Swarm Optimization for both time complexity and quality of solution.

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 method used to collect literature and conduct experiment. Section 4 present results and discussion and finally conclusion is drawn in section 5.

2. Basic concept of the algorithms

This section discusses the basic theoretical background of the ABC, BA and PSO algorithms. The soul of computer they say is algorithm; it is a sequence-of-steps or procedure deigned to solve a problem or help answer a question. Yet, there is no universally accepted definition for algorithm. Mathematically speaking, an algorithm is a procedure to generate outputs for given inputs. From the optimization point of view, an optimization algorithm generates a new solution xt11 to a given problem from a known solution xt at iteration or time t [10].

2.1. Artificial Bee Colony

Artificial Bee Colony simulates the foraging process of natural honey bees. 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 scout, they become employed bees. Then food sources are exploited by employed bees that also shares the information about the quality and quantity of food sources to the onlooker (bees resting at hive and waiting for the information from employed bees). A specific *waggle dance* is performed to share food information.

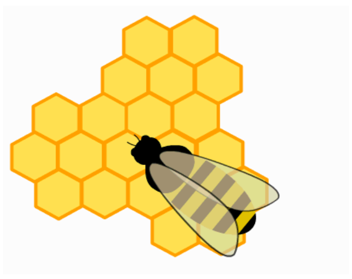

Figure 1. Bee colony

The ABC algorithm is presented below:

- **Initialization of random food sources**

The random food sources (FS) are generated in the search space using following Eq. (1):

$$x_{ij} = max_j + rand(0,1) \times (1)(max_j - min_j) \quad (1)$$

where i represents the FS and j denotes the jth dimension. max and min denote the upper and lower bounds.

- **Employed bee process**

The search equation involved in this phase and also performs the global search by introducing new food sources $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ corresponding to $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$is discussed below:

$$v_{ij} = x_{ij} + rand(-1,1) \times (x_{ij} - x_{kj}) \quad (2)$$

where k is selected randomly and distinct from i. Greedy selection mechanism is performed to select the population to store in a trail vector. In case v_{ij} fails corresponding to boundary constraints then they are handled using following Eq. (3):

$$v_{ij} = \begin{cases} max_j \ if \ v_{ij} > max_{ij} \\ min_j \quad if \ v_{ij} < min_j \end{cases} \tag{3}$$

From the Eq. (3), new solution may be generated then there will be greedy selection in Eqs. (3) and (4)

$$x_i = \begin{cases} v_i \ if \quad fit(v_i) > fit(x_i) \\ \ x_i \qquad\qquad otherwise \end{cases} \tag{4}$$

where $fit()$ represents the fitness value which is defined in Eq. (5) (for minimization case):

$$fit(x_i) = \begin{cases} \frac{1}{(1+f(x_i)} \qquad if \ f(x_i) > 0 \\ 1 + abs\, f(x_i) \ if \ f(x_i) \leq 0 \end{cases} \tag{5}$$

where $f()$ represents the objective function value.

- **Onlooker bee process**

Onlooker bee carry out local search in the region of the food sources shared by employed bee. Equation (6) is used to choose the food source by Onlooker bee from a set of FS solutions. Probability Pi is used to choose the food source (solution).

$$p_i = fit(x_i)/\sum_{i=1}^{FS} fit(x_i) \tag{6}$$

Onlooker bee chooses the food source having better probability, then Eq. (2) is used to exploit the food source and new food source is generated. After this a greedy process is followed using Eq. (4).

- **Scout bee process**

If the food source does not improve in the fix number of trials (limit a control parameter) then employed bees turns into scout bees and randomly forage for the new food sources. Initially ABC was designed to handle unconstrained optimization problems. Later ABC was modified to handle and solve COPs [22] by adding one more parameter, modification rate (MR) in employed and onlooker phase, constraints were handled using Deb's rule (Deb 2000) and thirdly another control parameter named SPP is added along with limit in scout bee phase that controls the abandoned food source, if it exceeds limit. If it exceeds limit then a scout production process is carried out. SPP ensures that the new food source randomly generated by the scout bee replace the proposed food source. Deb's rule suggests that:

a) Feasible solution is selected over infeasible solution.
b) In case two feasible solutions are there then the solution having best objective function value would be considered.
c) If both the solutions are infeasible then the one violating minimum number of constraints would be preferred.

Following Eq. (7) is used by the employed and onlooker bees to generate the new food source.

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij}\left(x_{ki} - x_{ij}\right) \ if \ R_j < MR \\ x_{ij} \qquad\qquad\qquad\qquad Otherwise \end{cases} \tag{7}$$

Where φ_{ij} is a random number in the range [−1,1] and MR controls the modification in xij and R ∈ [0,1].

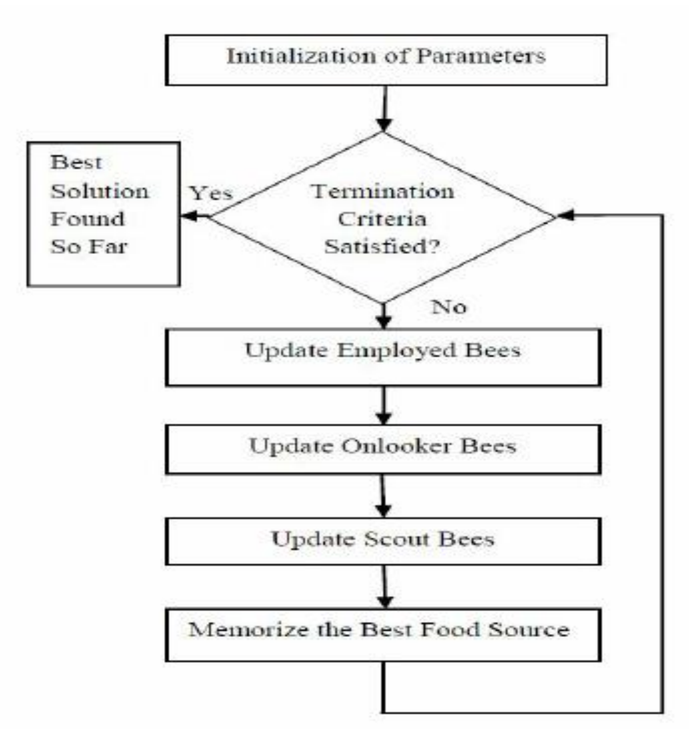
Initialization of Parameters
Best Solution Found So Far
Yes
Termination Criteria Satisfied?
No
Update Employed Bees
Update Onlooker Bees
Update Scout Bees
Memorize the Best Food Source

Figure 2. Flowchart of Artificial Bee Colony

2.2 Bat Algorithm

Bat algorithm (BA) was developed based on the echolocation features of microbats [3], 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 [7]. 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 [9].

[9] developed the bat algorithm with the following 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 vi at position xi 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 assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

2.3. Particle Swarm Optimization

Particle swarm optimization (PSO); it is originated from the analysis of behavior of birds catching food [13], 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 – fact, which is usually unexpected. 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 individual 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 [13][14].

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 objective function, or the function to be optimized (also known as the fitness function in the particle swarm), in order to evaluate the quality of the solution [15].

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. In the movement process of the particle swarm, information is conveyed between each individual 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 own 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 [13-15].

3. Materials and Methods

Program designed for each algorithm is made up of the same number of agents, decision variables, 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 Opytimizer class, which holds all the vital information about the optimization task. Finally the task is started, and when it finishes, it returns a history object which encodes valuable data which include 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 in other to easily compare and obtain a better result.
For each iteration count in an experiment, the fitness and position are recorded in a table which gives room for easy understanding and analysis. The time taken for every 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.

3. Results

The results obtained from the experiments are as follows. For each of the experiment, the best, average and worst cases were recorded and are shown in table 2, 3 and 4. Table 1 compare the running time of the experiments for each algorithm while tables 2, 3 and 4 also contain comparison of best, average and worst solutions of the algorithms for each experiment.

In terms of time complexity, particle swarm optimization has less running time or converge to the global optimum faster that both the other algorithms, closely followed by Bat algorithm and then Artificial Bee Colony. Figure 3: present visual representation of the time analysis.

Each algorithm results were compared to one another for easy analysis. Table ii compared the best solutions of the algorithm and we found out that ABC outperformed the other algorithms. That is; it has the best solution among best solutions as it can be seen in Fig. 4., followed closely by PSO and lastly BA. Fig. 5. compared the average cases of the algorithms. Also ABC proved to be more efficient than the other two. Worst case solutions were compared in Fig. 6., with BA being very worst compared to both PSO and ABC.

Table 1. Comparison of running time of each experiment

Experiment	**ABC**	**BA**	**PSO**
F1	0.532167196	0.268395424	0.258379221
F2	0.50595665	0.318498611	0.196287155
F3	0.538910627	0.262340546	0.220400572
F4	0.476151228	0.290428162	0.199978352
F5	0.496158838	0.275404453	0.185274363
F6	0.496167421	0.303462029	0.20508337
F7	0.452147245	0.300301075	0.19328475
F8	0.45152235	0.328483105	0.215129614
F9	0.439612389	0.280530691	0.212312937
F10	0.517238855	0.301443338	0.185272694
F11	0.451301098	0.295578718	0.19929266
F12	0.460430384	0.279411554	0.205303669
F13	0.429872274	0.306813955	0.164041996
F14	0.427607536	0.256379604	0.208134413
F15	0.450779915	0.263383389	0.181266546
F16	0.427275181	0.267396212	0.224447727
F17	0.437671423	0.302340031	0.202298164
F18	0.437671423	0.288424969	0.204300165
F19	0.437602043	0.295433283	0.200441837
F20	0.46074605	0.305450439	0.200441837

Table 2. Comparison of best solutions obtained from the experiments

Experiment	ABC	BA	PSO
F1	9.88E-07	0.079218	0.005835
F2	9.88E-07	0.079218	0.005835
F3	9.88E-07	0.079218	0.005835
F4	9.88E-07	0.079218	0.005835
F5	9.88E-07	0.079218	0.005835
F6	9.88E-07	0.079218	0.005835
F7	9.88E-07	0.079218	0.005835
F8	9.88E-07	0.079218	0.005835
F9	9.88E-07	0.079218	0.005835
F10	9.88E-07	0.079218	0.005835

Table 3. Comparison of average solutions obtained from the experiments

Experiment	ABC	BA	PSO
F1	4.50E-02	0.753815	0.10348
F2	4.50E-02	0.753815	0.10348
F3	4.50E-02	0.753815	0.10348
F4	4.50E-02	0.753815	0.10348
F5	4.50E-02	0.753815	0.10348
F6	4.50E-02	0.753815	0.10348
F7	4.50E-02	0.753815	0.10348
F8	4.50E-02	0.753815	0.10348
F9	4.50E-02	0.753815	0.10348
F10	4.50E-02	0.753815	0.10348

Table 4. Comparison of worst solutions obtained from the experiments

Experiment	ABC	BA	PSO
F1	8.96E-01	1.998503	1.499843
F2	8.96E-01	1.998503	1.499843
F3	8.96E-01	1.998503	1.499843
F4	8.96E-01	1.998503	1.499843
F5	8.96E-01	1.998503	1.499843
F6	8.96E-01	1.998503	1.499843
F7	8.96E-01	1.998503	1.499843
F8	8.96E-01	1.998503	1.499843
F9	8.96E-01	1.998503	1.499843
F10	8.96E-01	1.998503	1.499843

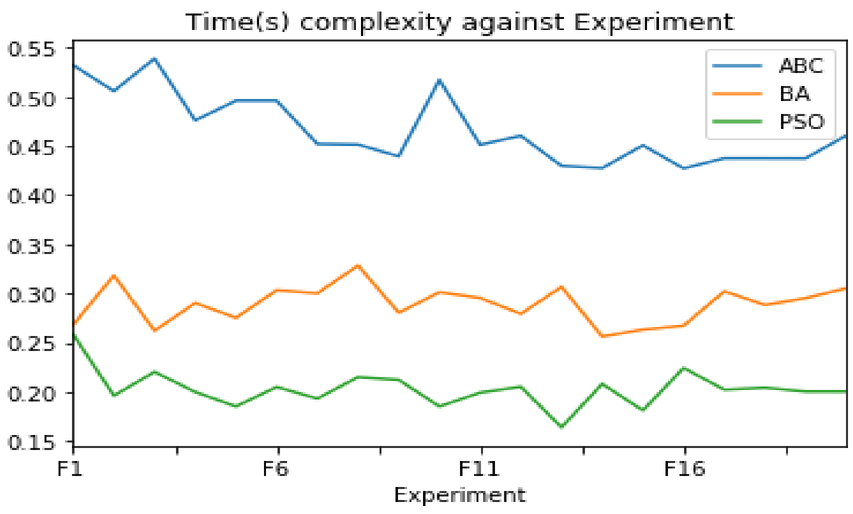

Figure 3. Running time of each experiments

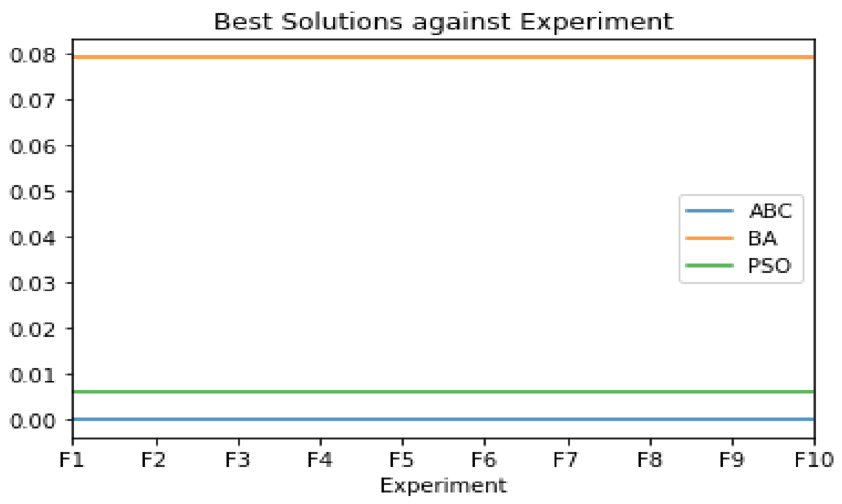

Figure 4. Comparison of best case of each experiment

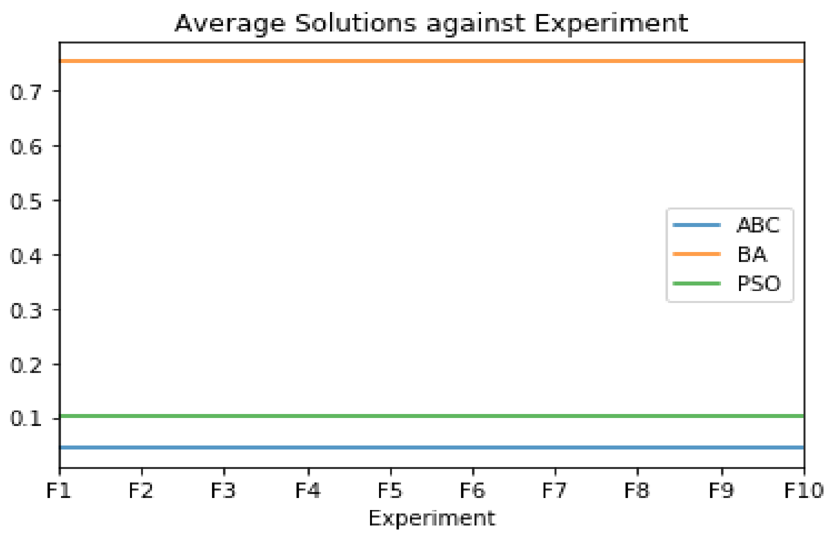

Figure 5. Comparison of average case of each experiment

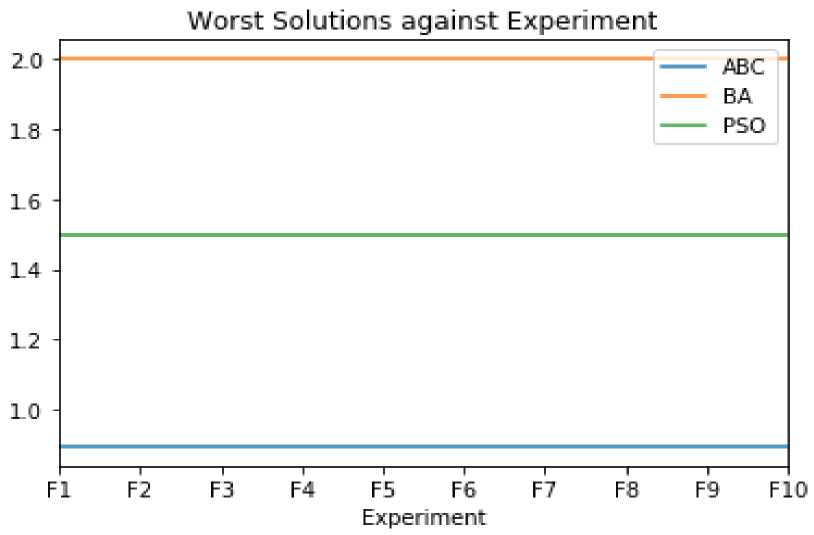

Figure 6. Comparison of worst case of each experiment

Tables 1-4 above show the results of the experiments for each of the three algorithms. The mean, best, and worst solutions are recorded and presented for ease of reference and comparison. Charts for each table are designed and displayed in figure 3-6 to make interpretation simple. Table ii present the best solutions for each of the algorithms. Chats are displayed in figures for each of the tables. Figure 2 showed the time complexity of the algorithms, as shown PSO has small running time than both BA and ABC.

The results revealed that the Particle swarm optimization converge to global optimum faster than the other two (BA and ABC), while in terms of quality of solution ABC outperformed the rest of the algorithms.

The results appeared like this because the algorithms are different and have different approach even though they are designed to achieve the same task. This study could be used by researcher to help choose a better algorithm to solve a problem or answer a question.

4. Conclusions

In conclusion, the objective of the research was to conduct an experiment using Opytimizer to determine/measure the performance (solution quality and time complexity) of three nature-inspired algorithms; particle swarm optimization, bat algorithm and artificial bee colony to determine which of the algorithms converge faster. Opytimizer python micro-framework was used on several benchmark functions. The experiment was run several times and the mean, best and worst cases were recorded. It revealed that PSO converge to global optimum faster than both the other two; BA and ABC. In terms of quality of solution ABC outperformed the rest of the algorithms. We have used basic versions of these algorithms without finely tuning the parameters to compare the results.

Author Contributions: Conceptualization, M.A. and M.A.; methodology, M.A.; software, M.A.; validation, M.A., A.A. and D.U.E.; formal analysis, M.A.; investigation, M.A. and A.A.; resources, M.A., A.A. and D.U.E.; data curation, M.A.; writing—original draft preparation, M.A.; writing—review and editing, M.A. and D.U.E; visualization, A.A. and D.U.E.; supervision, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflict of interest.

References

1. Chiroma, H., Gital, A. Y., Rana, N., Abdulhamid, S. M., Muhammad, A. N., Umar, A. Y., & Abubakar, A. I. (2019). *Nature Inspired Meta-heuristic Algorithms for Deep Learning: Recent Progress and Novel Perspective. Polish River Basins and Lakes – Part II, 59–70.* doi:10.1007/978-3-030-17795-9_5
2. Tajmiruzzaman, Md & Asadujjaman, Md. (2014). Artificial Bee Colony, Firefly and Bat Algorithm in Unconstrained Optimization.
3. X.-S. Yang, A New Metaheuristic Bat-Inspired Algorithm, in: *Nature Inspired Coop-erative Strategies for Optimization (NISCO 2010)* (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74 (2010).
4. Prabha, S., & Yadav, R. (2019). *Differential evolution with biological-based mutation operator. Engineering Science and Technology, an International Journal.* doi:10.1016/j.jestch.2019.05.012
5. Wang L., Liu X., Sun M., Qu J., and Wei Y. (2018) *A New Chaotic Starling Particle Swarm Optimization Algorithm for Clustering Problems*. Mathematical Problems in Engineering, Volume 2018, Article ID 8250480, 14 pages, https://doi.org/10.1155/2018/8250480
6. Chiroma, H., Herawan, T., Fister, I., Fister, I., Abdulkareem, S., Shuib, L., … Abubakar, A. (2017). *Bio-inspired computation: Recent development on the modifications of the cuckoo search algorithm. Applied Soft Computing, 61, 149–173.* doi:10.1016/j.asoc.2017.07.053
7. NiaPy: Python microframework for building nature-inspired algorithms Grega Vrbančič, Lucija Brezočnik, Uroš Mlakar, Dušan Fister, and Iztok Fister Jr.
8. Greco, R., Vanzi, I., New few parameters differential evolution algorithm with application to structural identification, Journal of Traffic and Transportation Engineering (English Edition) (2018), https://doi.org/10.1016/j.jtte.2018.09.002
9. Yang, X.-S., & Karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Computation. Swarm Intelligence and Bio-Inspired Computation, 3–23.* doi:10.1016/b978-0-12-405163-8.00001-6

10. Akay, B., & Karaboga, D. (2012). *A modified Artificial Bee Colony algorithm for real-parameter optimization. Information Sciences, 192, 120–142.* doi:10.1016/j.ins.2010.07.015
11. Jin, D., & Lin, S. (Eds.). (2011). *Advances in Computer Science, Intelligent System and Environment. Advances in Intelligent and Soft Computing.* doi:10.1007/978-3-642-23777-5
12. Meng, X., & Pian, Z. (2016). *Theoretical Basis for Intelligent Coordinated Control. Intelligent Coordinated Control of Complex Uncertain Systems for Power Distribution Network Reliability, 15–50.* doi:10.1016/b978-0-12-849896-5.00002-7
13. Martínez, C. M., & Cao, D. (2019). *Integrated energy management for electrified vehicles. Ihorizon-Enabled Energy Management for Electrified Vehicles, 15–75.* doi:10.1016/b978-0-12-815010-8.00002-8
14. 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. doi: 10.1109/EPEPS.2014.7103606
15. Chiroma, H., Khan, A., Abubakar, A. I., Saadi, Y., Hamza, M. F., Shuib, L., … Herawan, T. (2016). A new approach for forecasting OPEC petroleum consumption based on neural network train by using flower pollination algorithm. Applied Soft Computing, 48, 50–58. doi:10.1016/j.asoc.2016.06.038
16. H. A. Choudhury, N. Sinha and M. Saikia, Nature inspired algorithms (NIA) for efficient video compression – A brief study, Engineering, Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2019.10.001
17. Panda, M., & Das, B. (2019). *Grey Wolf Optimizer and Its Applications: A Survey. Proceedings of the Third International Conference on Microelectronics, Computing and Communication Systems, 179–194.* doi:10.1007/978-981-13-7091-5_17
18. Varadarajan, M., & Swarup, K. S. (2008). *Differential evolutionary algorithm for optimal reactive power dispatch. International Journal of Electrical Power & Energy Systems, 30(8), 435–441.* doi:10.1016/j.ijepes.2008.03.003
19. https://grega.xyz/2018/05/how-to-utilize-niapy-for-solving-knn-parameter-optimization-problem/ retrieved 9th January, 2020
20. Bujok, P., Tvrdík, J., & Poláková, R. (2019). *Comparison of nature-inspired population-based algorithms on continuous optimisation problems. Swarm and Evolutionary Computation.* doi:10.1016/j.swevo.2019.01.006
21. Karaboga, D., & Akay, B. (2011). *A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. Applied Soft Computing, 11(3), 3021–3031.* doi:10.1016/j.asoc.2010.12.001