The Eurasia Proceedings of Science,
Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2018

Volume 2, Pages 395-402

ICRES 2018: International Conference on Research in Education and Science

Implementation Strategies for the Cuckoo Search and the African Buffalo Optimization for the Benchmark Rosenbrock Function

Julius Beneoluchi ODILI
Anchor University

A. NORAZIAH
Universiti Malaysia Pahang

Radzi AMBAR
Universiti Tun Hussein Onn

Mohd Helmy Abd WAHAB
Universiti Tun Hussein Onn

Abstract: The introduction of five benchmark global optimization test functions by De Jong has remained prominent in Mathematics and Computer Science for over three decades now. This paper examines the effect of the search population and the number of iterations of the Cuckoo Search and the African Buffalo Optimization in providing solutions to one of Dejong function, the Rosenbrock function, sometimes called Dejong2 function which is a unimodal non-separable function. The Rosenbrock function because of its deceptive flat landscape has proven to be a good test case for optimization algorithms since the flat surface provides very misleading information to search agents. After a number of experimental investigations using different iteration numbers and population, this study concludes that the CS provides better solutions but at a cost of more computer resources than the ABO. As a result, this study in harmony with the No Free Lunch Theorem concludes that if speed is the main consideration, the ABO is a better algorithm in solving the Rosenbrock (or a similar function), otherwise, the CS is a better choice.

Keywords: African buffalo optimization, Cuckoo search, Iteration, Rosenbrock, Search population

Introduction

Kenneth Dejong has become a very popular figure in the Mathematics and Computer Science, especially because of his noble contributions to the field of optimization search landscape. In his PhD thesis, Dejong introduced five benchmark optimization functions that are fast becoming effective testbed for several optimization search algorithms. The benchmark functions are the Sphere function, Rosenbrock function, Step function, Quartic function and Shekel Foxhole function. The five functions are presented in Figures 1-5 (De Jong, 1975).

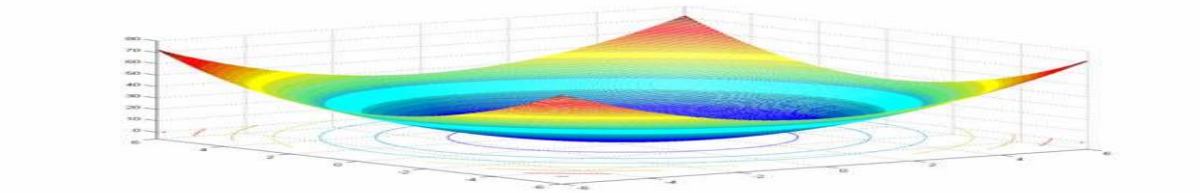

Figure 1. Sphere function (function, Accessed on 2nd May, 2018)

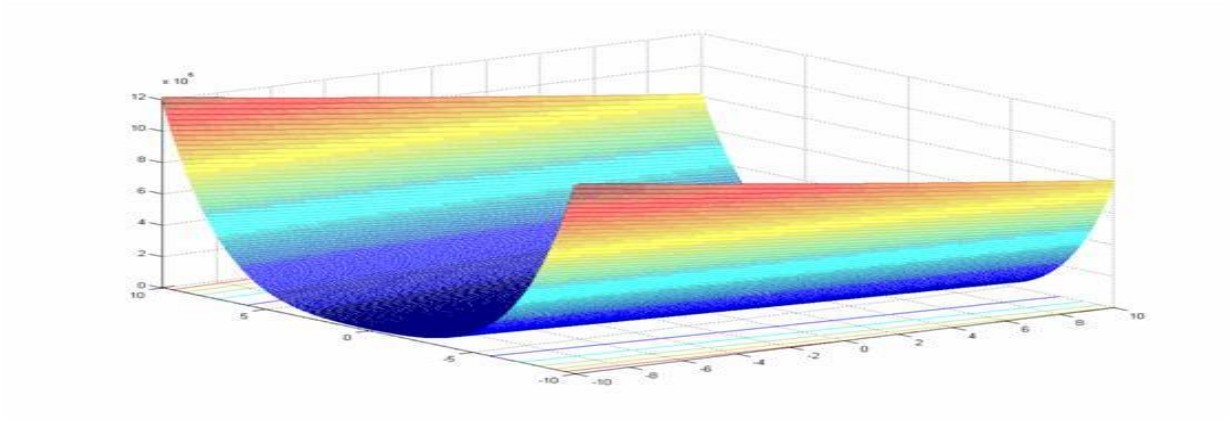

Figure 2.Rosenbrockfunction (Rosenbrock, Accessed on 2nd May, 2018)

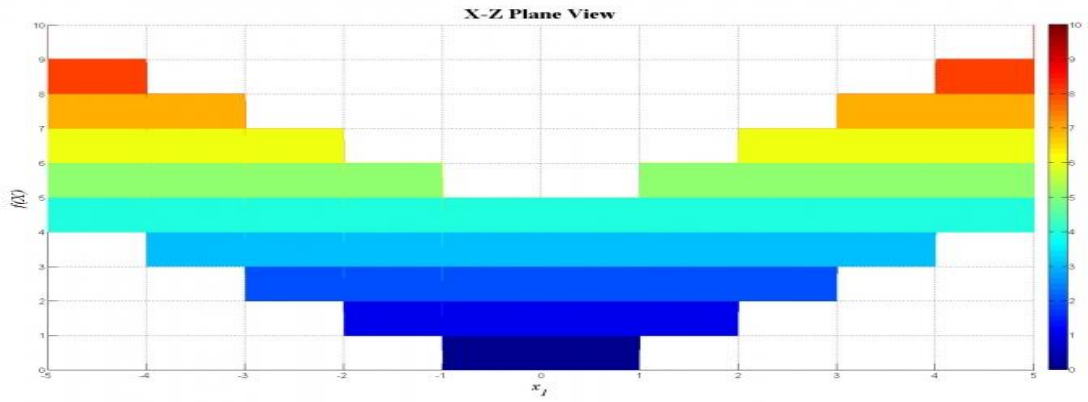

Figure 3. Step function accessed on 2nd May, 2018)

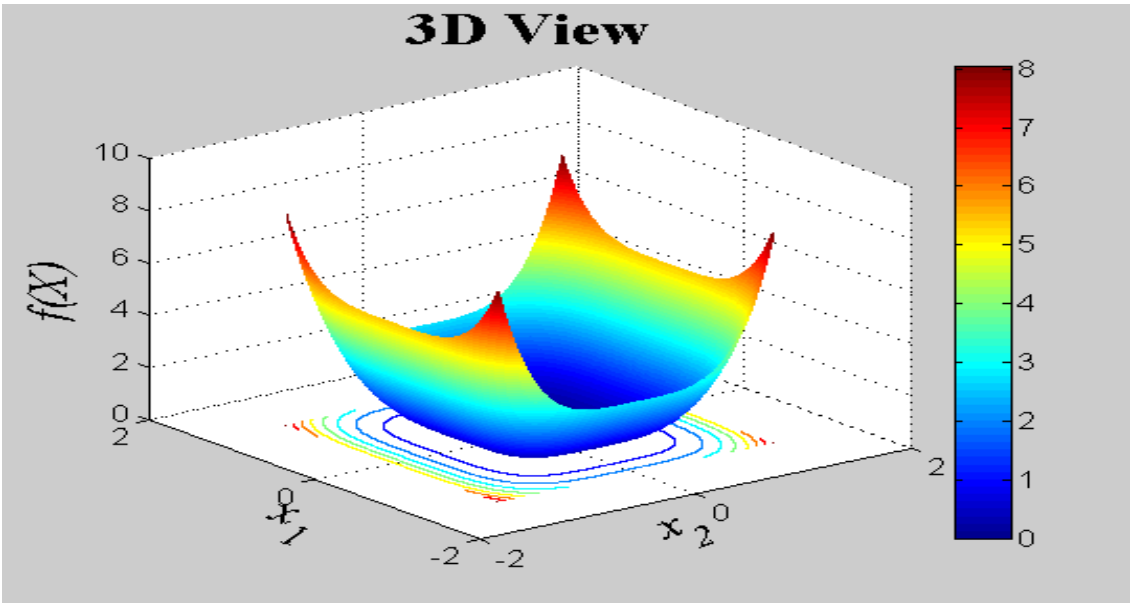

Figure 4. Quartic function

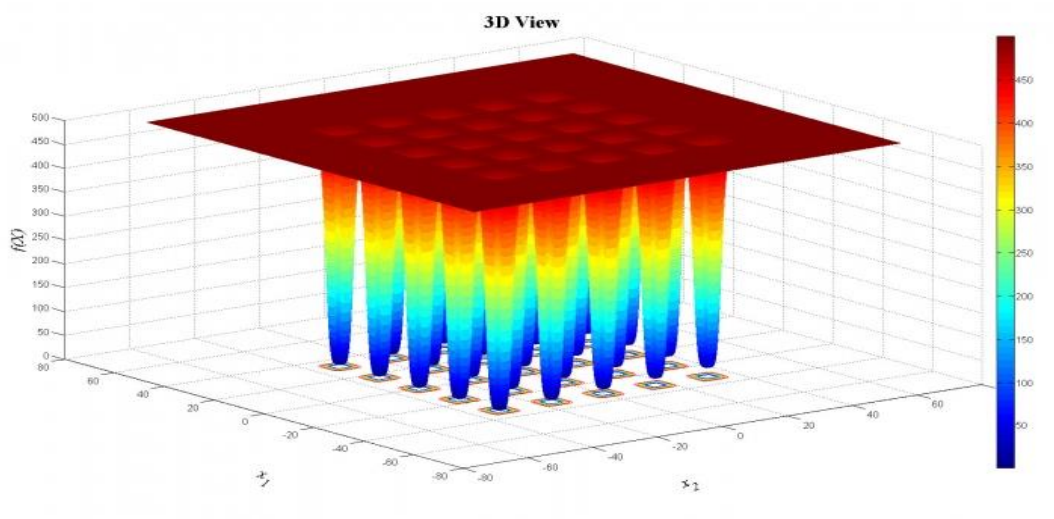

Figure 5. Shekel foxholes function

These functions represent different search landscapes ranging from monomodal Separable (Sphere, Quartic and Step function) Unimodal Non Separable (Rosenbrock) and Multimodal Non Separable (Shekel Foxhole)(Problems, Accessed on 11th February, 2017). A monomodal function has just a single minimum or optimum while a multimodal function has two or more optima or minima (Casini et al., 2012). Please note that by separable function we actually mean separable PDE (Partial Differential Equation) as opposed to ODE (Ordinary Differential Equation) (Odili and Kahar, 2015b). An ODE contains one or more functions of a particular independent variable and its derivatives (Kunna et al., 2015). Similarly, a separable PDE function is such that is divisible into a number of separate variables. That is to say such separable PDE can be re-written as a collection of f functions of just a variable. As such the separability of a function is akin to interrelation or epistasis among variables of a function (Visintin, 2012). Also, please recall that epistasis deals with the measurement of the contributions a gene in relation to other genes to the overall fitness of an individual (Sackton and Hartl, 2016).

In mathematics and computer science, it is believed that Non Separable functions are rather more difficult to optimize hence our interest in comparing the African Buffalo Optimization and the Cuckoo Search, two very effective and efficient algorithms, in solving the benchmark Rosenbrock function which is a non-separable function with particularly focus on the effect of the number of iteration and search populations in obtaining good solutions.

The rest of this paper is organized in the following ways: section two discusses the African Buffalo Optimization algorithm while section three examines the Cuckoo Search; section four is concerned with the experimental setting and discussion of results of the two algorithms in solving the benchmark Rosenbrock function and section five draws conclusions from the study.

Africa Buffalo Optimization

The ABO was designed in 2015 with inspiration from the harmonious herd management of African buffalos in their search for fresh green pastures in different parts of the African landscapes to satisfy their large appetites (Odili and Kahar, 2015a). The African buffalos manage their large herds using two vocalizations: the waaa (explore) and the maaa (exploit) vocalizations. With these two simple vocalizations, the buffalos are able to organize themselves out of harsh fruitless fields to very fertile and fruitful locations. The ABO has been successfully used to solve a number of optimization problems such as the symmetric Travelling Salesman's Problem (Odili and Mohmad Kahar, 2016b), asymmetric Travelling Salesman's Problem (Odili et al., 2015), Collision-avoidance in electric fishes, Strategic Management, PID Controller's parameters tuning of Automatic Voltage Regulators (Odili and Mohmad Kahar, 2016a)etc. The pseudocode of the ABO is presented in Figure 6.

1. Begin
2. Initialize the buffalos randomly within the search space;
3. While (until termination),
4. For j=1: n (n denotes the population),
5. Ascertain the buffalos' exploitation location:
6. $mk' = mk + lp1(bg - wk) + lp2(bpk - wk)$
7. Here wk =exploration move; where mk = exploitation move; bg = best buffalo with the best fitness; lp1 and lp2 represents the learning parameters; bpk , best location of buffalo k
8. Update the exploration fitness of buffalos:
9. $wk' = (wk + mk)\,\lambda$
10. Ascertain whether the bg is updating? Yes, go to 11. If No in 10 iterations, return to 2
11. End for
12.. End while
13. Post best solution.
14. End

Figure 6. Pseudocode of ABO

Cuckoo Search

The Cuckoo Search (CS) which was developed by X. Yang and S. Deb basically simulates the irresponsible attitude of the cuckoo bird to brood over their eggs until hatching (Yang and Deb, 2009). The cuckoo enjoys laying her eggs in the nests of other unwatchful birds (or other cuckoo species) in order to transfer the egg-brooding responsibilities to the host bird. If the host bird discovers the prank of the cuckoo, it either abandons the nest or throws away the cuckoo eggs(exploration). At other times, the host bird goes ahead to broods over and incubates the cuckoo eggs (exploitation). On its part, the cuckoo bird, with a certain probability, perfects her act by imitating the eggs of the host bird in order to perpetuate its fraud. In the CS, the eggs belonging to the host bird in a particular nest represents a solution to an optimization problem, while those of the cuckoo represents newer solutions. The overall aim of these two kinds of egg is to replace the older solutions with the newer ones (cuckoo's).

The CS has been applied successfully to solve a number of optimization problems, such as the wireless sensor networks, travelling salesman's problems, shortest path in distributed system document clustering, speech recognition, flood forecasting, job scheduling, image processing, classification task in health sector etc. with competitive outcomes (Kamat and Karegowda, 2014). The CS pseudocode (Agrawal et al., 2013) is presented in Figure 7

1. Begin
2. Objective function: f(x) x = (x1,x2….xj)
3. Ascertain the initial population of nests and distribute them randomly
4. While (until termination)
5. Generate randomly a cuckoo by Levy flight using
6. $Xij(t + 1) = Xij(t) + \alpha$ Levy (λ)
7. Determine the cuckoo fitness
8. Choose a nest randomly among available host nests
9. If ($f\ i > fk$) then
10. Replace k by the new solution
11. End if
12. Abandon a fraction of the unfruitful nests and replace with newer ones
13. Keep the good solutions
14. Rank the good solutions and obtain the current overall best
15. End while
16. Output the best outcome
17. End

Figure 7. Cuckoo search pseudocode

Implementation Evaluation for Cuckoo Search and African Buffalo Optimization

In this paper the focus is unravelling the impact of the number of iterations as well as the search population needed to obtain the best outcome to the problem under investigation. The experiments were performed using MATLAB on a PC, 4GB RAM, Intel Duo Core i7 370 CPU @ 3.40GHz, 3.40GH running Windows 10. To make sure that the comparisons are fair, all experiments were implemented using MATLAB on the same PC. The buffalo population/nests are 10 and 50. Similarly, number of iterations are 10, 20, 100, 1000, 5000, and 10,000. The ABO parameters used for the experiments are lp1=0.7; lp2=0.5 and those of the CS are pa=0.5; u=rand (size (s)) * sigma; v= rand (size(s)); step = u./abs (v). ^ (1/beta); step size =0.01* step. Each experiment was executed five times. The benchmark Rosenbrock function (Shi and Eberhart, 1999) is

$$f(x) = \sum[(100\ xi - xi2)2 + (xi - 1)2]\ \ d{-}1\ i{=}1 \quad (1)$$

The optimal solution to the Rosenbrock function is:

$$f(x) = 0 \quad (2)$$

Table 1. Comparative search with 10 buffalos/nests

	ABO				**CS**			
Iterations	f_{min}	**Average**	**Time (secs)**	**Average Time (s)**	f_{min}	**Average**	**Time (secs)**	**Average Time (s)**
	0.0359		0.022		2.3080		0.040	
	0.0580		0.021		0.0013		0.031	
10	0.0028	0.0426	0.022	0.021	0.0329	0.5634	0.033	0.031
	0.1025		0.021		0.4738		0.034	
	0.0137		0.019		0.0012		0.018	
	0.0018		0.030		$3.9731e^{-13}$		0.172	
	0.0018		0.059		$9.8137e^{-14}$		0.162	0.163
100	0.0031	0.0527	0.060	0.0542	$4.4142e^{-12}$	$5.0611e^{-13}$	0.154	
	0.0308		0.059		$1.5596e^{-13}$		0.170	
	0.2259		0.063		$5.5449e^{-15}$		0.157	
	0.0011		0.468		$4.6147e^{-85}$		1.565	
	0.0004208		0.455		$1.0024e^{-83}$		1.556	1.5874
1000	0.00065974	0.0077	0.472	0.4650	$4.2801e^{-78}$	$4.4527e^{-78}$	1.563	
	0.0124		0.464		$4.2170e^{-69}$		1.600	
	0.0241		0.466		$8.1493e^{-75}$		1.653	
	$2.503e^{-07}$		2.293		0		8.118	
	$1.0443e^{-04}$		2.291		$3.0304e^{-318}$		8.086	7.9480
5000	$1.4071e^{-06}$	$2.9357e^{-06}$	2.277	2.273	$4.3782e^{-318}$	$2.4697e^{-320}$	7.867	
	$6.6688e^{-06}$		2.247		0		7.848	
	$3.0553e^{-05}$		2.258		$4.9407e^{-324}$		7.821	
	$1.8499e^{-05}$		4.640		0		15.372	
	$3.5446e^{-05}$		4.497		0		15.586	15.761
10000	$2.8062e^{-05}$	$3.7547e^{-05}$	4.421	4.488	0	0	15.683	
	$6.9589e^{-05}$		4.422		0		15.979	
	$3.6141e^{-05}$		4.458		0		16.183	

After a number of experiments, the simulation results of the CS and ABO searching with different populations of 10 and 50 nests/buffalos and different number of iterations ranging from 10 to 10,000 is presented in Table 1.

As can be seen in Table 1, at 10 iterations, the CS has an overall better than the ABO. It was only when the iteration was 10 that the ABO obtained a better result (mean: 0.0426 to CS's 0.5634) In all other instances of the run, the CS average results were better. This could be as a result of the ABO's combination of exploration fitness with exploitation fitness at each iteration resulting in faster convergence at a solution. However, in the other instances (that is, at iteration 100, 1000, 5000 and 10,000), the CS obtained better solutions.

Nonetheless, in terms of time taken to obtain good result, the ABO is the algorithm of choice. In 96% of the runs in Table 1, the ABO spent less time than the CS to obtain results. It was only in one instance that the CS converged earlier than the ABO and that was in the last run of iteration 10 (refer to the portion highlighted in

yellow ink). This fast speed of the ABO is a mark of the algorithm's efficiency because time correlates with use of computer resources (Khompatraporn et al., 2005).

Table 2. Comparative search with 50 buffalos/nests

	ABO				**CS**			
Iterations	f_{min}	**Average**	**Time (sec)**	**Average Time (s)**	f_{min}	**Average**	**Time (secs)**	**Average Time (s)**
10	0.0101	0.0357	0.063	0.051	0.1048	1.0334	0.071	0.0784
	0.0518		0.052		1.1899		0.068	
	0.091		0.051		0.0314		0.068	
	0.013		0.037		2.9252		0.068	
	0.0127		0.052		0.9157		0.117	
100	0.0126	0.0058	0.229	0.228	$4.1464e^{-14}$	$3.8376e^{-15}$	0.172	0.163
	0.0013		0.225		$6.8572e^{-14}$		0.162	
	0.0044		0.229		$2.6223e^{-16}$		0.154	
	0.0036		0.230		$2.7811e^{-15}$		0.170	
	0.0069		0.227		$2.7811e^{-16}$		0.157	
1000	$6.1493e^{-05}$	$4.4423e^{-04}$	1.981	2.032	$1.3143e^{-54}$	$2.2048e^{-55}$	6.231	6.310
	$6.1548e^{-05}$		1.986		$5.4190e^{-56}$		6.257	
	$3.4435e^{-04}$		2.056		$1.6071e^{-56}$		6.364	
	$2.8068e^{-04}$		2.052		$1.0727e^{-55}$		6.384	
	$3.6573e^{-05}$		2.083		$1.6111e^{-54}$		6.313	
5000	$4.1663e^{-06}$	$2.7808e^{-06}$	4.526	8.761	$6.8572e^{-161}$	$4.9646e^{-165}$	26.775	30.042
	$1.9063e^{-07}$		9.830		$2.6223e^{-169}$		25.366	
	$2.7224e^{-05}$		9.737		$2.7811e^{-165}$		31.439	
	$1.6645e^{-05}$		10.012		$7.3016e^{-165}$		32.744	
	$3.4444e^{-06}$		9.702		$5.2609e^{-167}$		33.884	
10000	$7.8032e^{-06}$	$4.1005e^{-06}$	20.117	20.237	$2.0759e^{-269}$	$5.3958e^{-272}$	67.500	68.441
	$5.9999e^{-07}$		20.734		$6.5280e^{-272}$		68.044	
	$2.0300e^{-07}$		19.853		$8.5684e^{-271}$		68.585	
	$1.0420e^{-06}$		20.138		$8.2255e^{-269}$		70.631	
	$3.6275e^{-05}$		20.341		$1.5813e^{-279}$		67.447	

Based on the results presented in Table 1, it may be safe to conclude that while the CS is a more effective algorithm in terms of obtaining optimal or near optimal solution to this particular problem (and by extension, other similar problems to the one) under investigation here, the ABO is a more efficient algorithm since the algorithm converges faster to a solution than the CS.

Furthermore, it is necessary to investigate the performances of both algorithms when a population of 50 search agents (buffalos or nests) are deployed to the search space. In the next set of experiments, 50 buffalo/nests population and different iterations numbers (10, 100, 1000, 5000 and 10,000) are used. The simulation outcome presented in Table 2.

The experimental output in Table 2 follows the trend in Table 1, that is, that the ABO converges faster than the CS. This can be seen in the results of using a population of 50 buffalos or nests when the iteration number is 10. The ABO obtained better result (mean: 0.0357 to CS's 1.0334) due to ABO's capacity for quicker convergence at a solution. As in the first set of experiments, from iteration 100, the tide turned in favor of the more effective CS (refer to Table 2).

Similarly, in terms of time taken to arrive at a solution, the ABO has an edge over the CS just like in the first set of experiments (refer to Table 1). It was only in iteration 100, that the CS had a faster speed than the ABO. In all other instances, the ABO converged faster at a solution, though in most instances, the ABO's output is inferior to those of the CS. In fact, from iteration 1000 to 10,000, the ABO is, at least, three times faster than the CS. The reason for the ABO's speed could be traceable to the Algorithm's use of fewer parameters in its quest for solutions than the CS. The ABO being a parameter-less optimization algorithm uses just two controlling parameters, namely, the $lp1$ and $lp2$. The CS, on the other hand, uses a number of parameters such as pa, step, step size, sigma etc. Deploying several parameters in course of a search has the demerit of slowing down the speed of a search since the algorithm is bogged down by the parameter handling procedure (Sörensen, 2015).

Conclusion

This paper examines the effects of the population of search agents cum iteration number in solving the benchmark Rosenbrock function. The choice of these two comparative algorithms is as a result of their very good results in solving the optimization problems to which they have been deployed. Moreover, aside both algorithms being both recently designed population optimization search algorithms, the CS is a parameterized optimization search algorithm but the ABO is a parameter-less algorithm. A parameter-less algorithm does not require the tuning of individual parameters of a problem to obtain solutions, it simply uses the controlling parameters of the algorithm to solve any kind of problems it is confronted with. In the case of the ABO, the controlling parameters are the lp1 and lp2. The only two other examples of the parameter-less algorithms, in literature, are the Teaching Learning Based Optimization (TLBO) and the Jaya Algorithms. In the light of the differences and similarities of the CS and the ABO, therefore, it is necessary to investigate their capacities to solve different kinds of problems, hence this study.

After a number of experimental evaluations, it was discovered that in solving the benchmark Rosenbrock function (or a similar problem), the CS produced better outcome. The good results of the CS could be the problem-specific tuning of parameters. On the other hand, the ABO converges faster at a solution than the CS. This could be as a result of its use of fewer parameters, regular interactions among the buffalos coupled with straight-forward calculation of exploitation and exploration fitness of the buffalos.

Based on the foregoing, it may be safe to conclude that, in line with the No Free Lunch Theorem which states that there is no algorithm that is best to solve all problems, rather whatever is of interest to the practitioner/researcher may determine the choice of an optimization algorithm to solve a problem. In the problem under investigation here (the benchmark Rosenbrock function or a similar problem to it), if obtaining near-optimal solution at the shortest possible time is the utmost concern of the user, then the ABO is a better algorithm. Also, the main concern is obtaining a solution that is nearest to the optimal, then CS should be choice. Similarly, a nonprofessional user may prefer the ABO, being a parameter-less algorithm, because the user is saved the problem of parameter tuning.

Acknowledgement

The author appreciates the Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, for funding this study under Grant PGRS 1403118. Our appreciation to Anchor University, Ayobo,Lagos, Nigeria and the Universiti Tun Hussein Onn, Malaysia for additional fundings

Conflict of Interest

The author asserts that no conflict of interest exists in the publication of this manuscript

References

Agrawal, S., Panda, R., Bhuyan, S. and Panigrahi, B. K. (2013) 'Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm', *Swarm and Evolutionary Computation*, 11, pp. 16-30.

Casini, F., Vaunat, J., Romero, E. and Desideri, A. (2012) 'Consequences on water retention properties of double-porosity features in a compacted silt', *Acta Geotechnica*, 7(2), pp. 139-150.

De Jong, K. A. (1975) 'Analysis of the behavior of a class of genetic adaptive systems'.

function, S. (Accessed on 30th January, 2017a) 'http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page1113.htm'.

Function, S. (Accessed on 30th January, 2017b) 'http://www.al-roomi.org/benchmarks/unconstrained/n-dimensions/192-step-function-no-1'.

Kamat, S. and Karegowda, A. (2014) 'A brief survey on cuckoo search applications', *Int. J. Innovative Res. Comput. Commun. Eng*, 2(2).

Khompatraporn, C., Pintér, J. D. and Zabinsky, Z. B. (2005) 'Comparative assessment of algorithms and software for global optimization', *Journal of Global Optimization*, 31(4), pp. 613-633.

Kunna, M. A., Kadir, T. A. A., Jaber, A. S. and Odili, J. B. (2015) 'Large-Scale Kinetic Parameter Identification of Metabolic Network Model of E. coli Using PSO', *Advances in Bioscience and Biotechnology*, 6(02), pp. 120.

Odili, J. B. and Kahar, M. N. M. (2015a) 'African Buffalo Optimization (ABO): a New Meta-Heuristic Algorithm', *Journal of Advanced & Applied Sciences*, pp. 101-106.

Odili, J. B. and Kahar, M. N. M. (2015b) 'Numerical Function Optimization Solutions Using the African Buffalo Optimization Algorithm (ABO)', *British Journal of Mathematics & Computer Science*, 10(1), pp. 1-12.

Odili, J. B., Kahar, M. N. M., Anwar, S. and Azrag, M. A. K. 'A comparative study of African Buffalo Optimization and Randomized Insertion Algorithm for asymmetric Travelling Salesman's Problem'. *Software Engineering and Computer Systems (ICSECS), 2015 4th International Conference on*: IEEE, 90-95.

Odili, J. B. and Mohmad Kahar, M. N. (2016a) 'African Buffalo Optimization Approach to the Design of PID Controller in Automatic Voltage Regulator System', *National Conference for Postgraduate Research, Universiti Malaysia Pahang*, September, 2016, pp. 641-648.

Odili, J. B. and Mohmad Kahar, M. N. (2016b) 'Solving the Traveling Salesman's Problem Using the African Buffalo Optimization', *Computational intelligence and neuroscience*, 2016, pp. 1-12.

Problems, B. (Accessed on 11th February, 2017) 'Benchmark Problems', *https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/ortizboyer05a-html/node6.html*.

Rosenbrock (Accessed on 30th Janauary, 2017) 'http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction/rosenbrock.html'.

Sackton, T. B. and Hartl, D. L. (2016) 'Genotypic context and epistasis in individuals and populations', *Cell*, 166(2), pp. 279-287.

Shi, Y. and Eberhart, R. C. 'Empirical study of particle swarm optimization'. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*: IEEE, 1945-1950.

Sörensen, K. (2015) 'Metaheuristics—the metaphor exposed', *International Transactions in Operational Research*, 22(1), pp. 3-18.

Visintin, A. (2012) *Models of phase transitions.* Springer Science & Business Media.

Yang, X.-S. and Deb, S. 'Cuckoo search via Lévy flights'. *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*: IEEE, 210-214.

Author Information

Julius Beneoluchi Odili
Anchor University, Lagos
1-4, Ayobo Road, Ipaja, Lagos
Odili_julest@yahoo.conm

A. Noraziah
Universiti Malaysia Pahang,
Gambang, Kuantan 26300, Malaysia

Radzi Ambar
Universiti Tun Hussein Onn, Malaysia
Batu Pahat, Johor, Malaysia

Mohd Helmy Abd Wahab
Universiti Tun Hussein Onn, Malaysia
Batu Pahat, Johor, Malaysia