35. Walkley, A. and Black, I. A., An examination of the Degtrajeff method for determining soil organic matter and a proposed modification of the chromic acid titration method. *Soil Sci.*, 1934, **37**, 29–38.
36. Blake, G. R. and Hartge, K. H., Bulk density. In *Methods of Soil Analysis Part 1 – Physical and Mineralogical Methods* (ed. Klute, A.), Agronomy Monograph 9, American Society of Agronomy – Soil Science Society of America, Madison, 1986, 2nd edn, pp. 363–382.
37. Campbell, B. L., Loughran, R. J. and Elliott, G. L., A method for determining sediment budgets using cesium-137, Sediment Budgets, Porto Alegre Symposium (December 1988), International Association of Hydrological Sciences (IAHS), 1988, **174**, 171–179.
38. Ritchie, J. C. and McHenry, J. R., Application of radioactive fallout Cesium-137 for measuring soil erosion and sediment accumulation rates and patterns: a review. *J. Environ. Qual.*, 1990, **19**, 215–233.
39. Walling, D. E. and He, Q., Improved models for estimating soil erosion rates from Cesium-137 measurements. *J. Environ. Qual.*, 1999, **28**(2), 611–622.
40. Walling, D. E. and He, Q., Models for converting ^{137}Cs measurements to estimates of soil redistribution rates on cultivated and undisturbed soils (including software for model implementation), Report to IAEA, University of Exeter, Exeter, UK, 2001, p. 32.
41. Walling, D. E., *Using Environmental Radionuclides as Tracers in Sediment Budget Investigations*, IAHS Publication, Crediton in Devon, UK, 2003, vol. 283, pp. 57–78.
42. Owens, P. N. and Walling, D. E., The use of a numerical mass balance model to estimate rates of soil redistribution on uncultivated land from ^{137}Cs measurements. *J. Environ. Radioact.*, 1998, **40**, 185–203.
43. Chappell, N. P., Webb, R. A., Viscarra, R. and Bui, E., Australian net (1950s–1990) soil organic carbon erosion: implications for CO_2 emission and land-atmosphere modelling. *Biogeosciences*, 2014, **11**, 5235–5244.
44. Cremers, A. *et al.*, Quantitative analysis of radiocaesium retention in soils. *Nature*, 1998, **335**, 247–249.
45. Khodadadi, M., Mabit, L., Zaman, M., Porto, P. and Gorgi, M., Using ^{137}Cs and ^{210}Pb measurements to explore the effectiveness of soil conservation measures in semi arid land: a case study in the Konhin region of Iran. *J. Soils Sediments*, 2019, **19**(4), 2103–2113.
46. Zhang, J., Yang, M., Sun, X. and Zhang, F., Estimation of wind and water erosion based on slope aspects in the crisscross region of the Chinese Loess plateau. *J. Soils Sediments*, 2018, **18**, 1620–1631.
47. Sharda, V. N. and Mandal, D., Prioritization and field validation of erosion risk areas for combating land degradation in north western Himalayas. *Catena*, 2018, **164**, 71–78.
48. Bajracharya, R. M., Lal, R. and Kimble, J. M., Erosion effects on CO_2 concentration and C-flux from an Ohio Alfisol. *Soil Sci. Soc. Am. J.*, 2000, **64**, 694–700.
49. Zhang, X. B., Qi, Y. Q., Walling, D. E., He, X. B., Wen, A. B. and Fu, J. X., A preliminary assessment of the potential for using '''Pbex measurement to estimate soil redistribution rates on cultivated slopes in the Sichuan Hilly Basin of China. *Catena*, 2006, **68**, 1–9.
50. Owens, L. B., Malone, R. W., Hothem, D. L., Starr, G. C. and Lal, R., Sediment carbon concentration and transtport from small watersheds under various conservation tillage practices. *Soil Till. Res.*, 2002, **67**, 65–73.
51. Roose, E. J., Lal, R., Feller, C., Barthes, B. and Stewart, B. A., *Soil Erosion and Carbon Dynamics*, CRC Press, Boca Raton, FL, USA, 2006; https://doi.org/10.1201/9780203491935
52. Mandal, D. and Dadhwal, K.S., Land evaluation and soil assessment for conservation planning and enhanced productivity. CSWCRTI Annual Report, 2012, p. 90.
53. Lal, R., Kimble, J. M., Follett, R. F. and Stewart, V. A., *Assessment Method for Soil Carbon*, CRC Publication, Boca Raton, Washington DC, USA, 2001.

ACKNOWLEDGEMENTS. We thank the Indian Council of Agricultural Research (ICAR), New Delhi for financial support (ICAR-National Fellow project) and the Director, Indian Institute of Soil and Water Conservation for providing the necessary facilities and support to carry out the field experiment and other research work. Laboratory support extended by the Centre for Advanced Research in Environmental Radioactivity (CARER), Mangalore University, for carrying out ^{137}Cs concentration measurements in the soil is acknowledged.

Received 21 February 2019; revised accepted 3 July 2019

doi: 10.18520/cs/v117/i5/865-871

Comparative implementation of the benchmark Dejong 5 function using flower pollination algorithm and the African buffalo optimization

Julius Beneoluchi Odili[1,*] **and A. Noraziah**[2,3]

[1]Department of Mathematical Sciences, Anchor University Lagos, Ipaja, Lagos, Nigeria
[2]Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Kuantan 26300, Malaysia
[3]IBM Centre of Excellence, Universiti Malaysia Pahang, Kuantan 26300, Malaysia

This communication presents experimental research findings on the application of the flower pollination algorithm (FPA) and the African buffalo optimization (ABO) to implement the complex and fairly popular benchmark Dejong 5 function. The study aims to unravel the untapped potential of FPA and the ABO in providing good solutions to optimization problems. In addition, it explores the Dejong 5 function with the hope of attracting the attention of the research community to evaluate the capacity of the two comparative algorithms as well as the Dejong 5 function. We conclude from this study that in implementing FPA and ABO for solving the benchmark Dejong 5 problem, a population of 10 search agents and using 1000 iterations can produce effective and efficient outcomes.

Keywords: Benchmark, comparative implementation, iteration, optimization algorithms, search agents, test functions.

K. A. DEJONG has made significant contributions to computer science. One of his remarkable contributions is the

*For correspondence. (e-mail: odili_julest@yahoo.com)

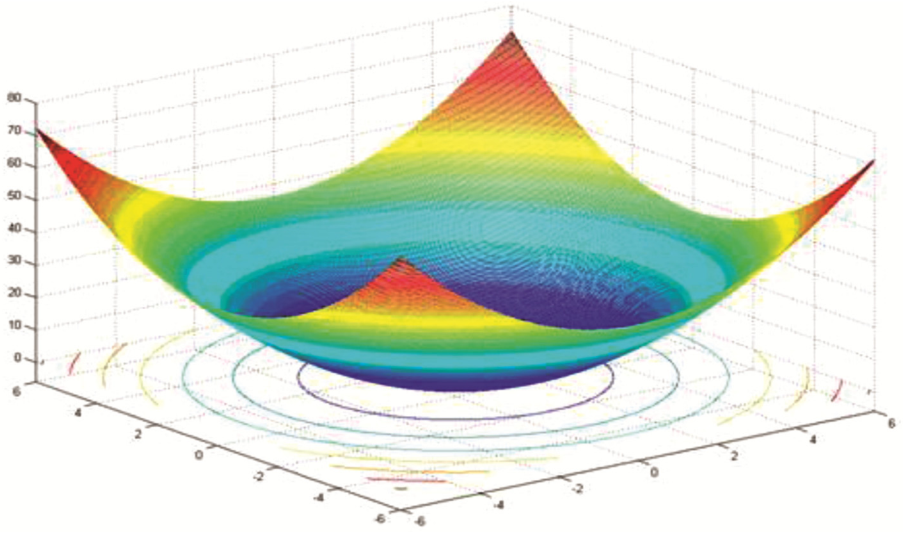

Figure 1. Dejong 1 (sphere) function[3].

introduction of a set of functions, sometimes called by his name. In his Ph D thesis, Dejong introduced five functions[1,2]. These are sometimes called Dejong 1–5, or they are named separately. For instance, Dejong 1 is popularly called sphere function; Dejong 2 is Rosenbrock; Dejong 3 is step function; Dejong 4 is quartic function, and the complex and unpopular Dejong 5 is also called Shekel's foxholes function[3].

In computer science and applied mathematics, in general, test functions, otherwise referred to as artificial landscapes, are used in evaluating the efficiency and effectiveness of optimization algorithms in terms of their robustness, convergence rate, versatility, precision and general performance. In recognition of the need for these artificial landscapes, De Jong suggested the above-listed five functions because of his belief that they represent the most commonly observed difficulties encountered in optimization problems. A brief description of these five functions is presented below:

- The sphere is the most popular of all of Dejong functions. It is a unimodal, symmetric and smooth function. The capacity of an algorithm to identify its global optimum is, doubtless, a good measure of its effectiveness.
- Unlike the 'popular and easy' sphere function, the Rosenbrock function is sometimes considered the nightmare of optimization algorithms. This function has a very narrow ridge whose tip is rather sharp, and the Rosenbrock tends to run around a parabola. Many optimization algorithms find it difficult to discover the global optimum of the Rosenbrock function.
- The step function is a representation of problem landscapes with deceptive flat surfaces. Usually, flat surfaces pose problems to optimization algorithms because they provide insufficient or no information to these algorithms with regard to which direction is favourable. Unless the optimization algorithm has in-built effective use of randomness, it may likely get stuck in one of the flat surfaces.
- The quartic function, on the other hand, is a unimodal function that is padded with Gaussian noise. The addition of Gaussian noise makes it extremely difficult for the algorithm to obtain an appropriate value on any given point during the search process. This function provides a good test for any optimization algorithm searching a noisy landscape.
- Finally, the shekel's foxholes function provides a good example of multimodal search landscape. This function has 25 local optima and 1 global optimum. The capacity of an optimization algorithm to discover each of these 25 local optima as well as locate the global optimum is a mark of a good optimization search algorithm.

In this study, we focus on Dejong 5 function because of its complexity. Due to its nature, we prefer to refer to this function as the multi-legged table function[4]. It has 25 local optima and one global minimum. Also, because of its deceptive nature, it poses a great problem to optimization algorithms, that researchers tend to ignore it. This is our motivation as it provides a good test for optimization search algorithms. Figures 1–5 present the five benchmark Dejong functions[5,6].

In this study, we evaluate the capacity of the flower pollination algorithm (FPA), one of the less popular, yet powerful optimization algorithms developed by Yang[7] and the African buffalo optimization (ABO), a recently designed optimization algorithm inspired by the migrant lifestyle of African buffalos[8,9], to solve the Dejong 5. The choice of these two algorithms for a comparative study of

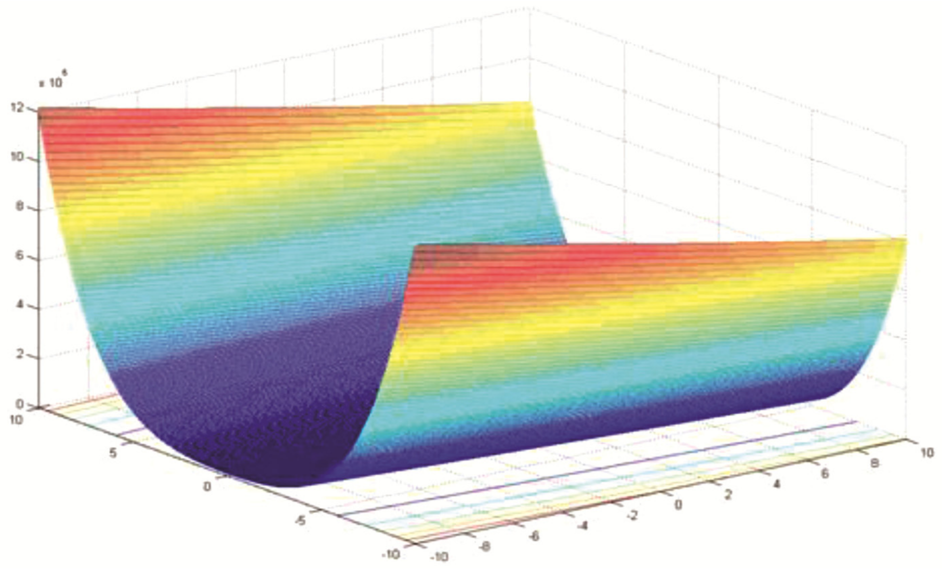

Figure 2. Dejong 2 (Rosenbrock) function[5].

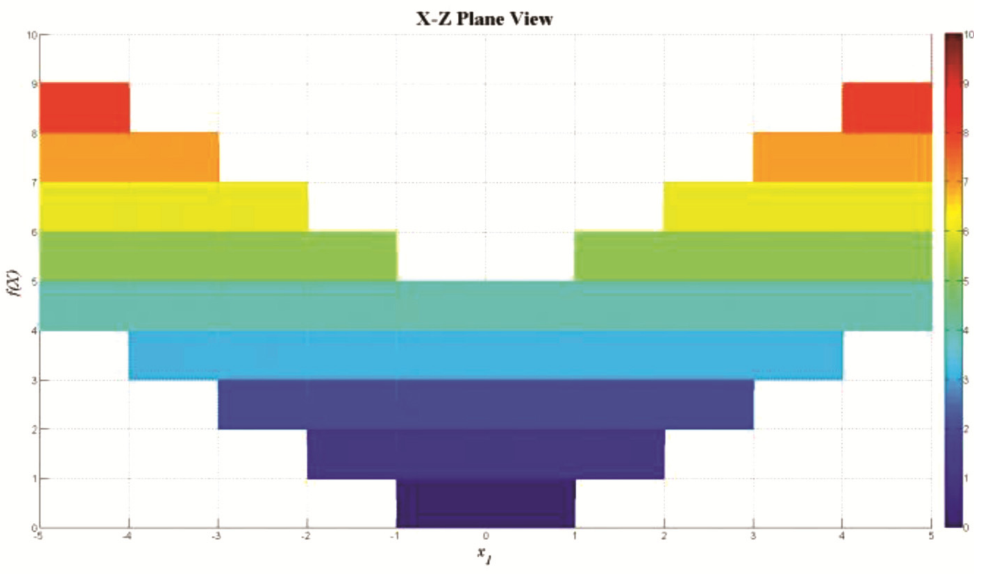

Figure 3. Dejong 3 (step) function[6].

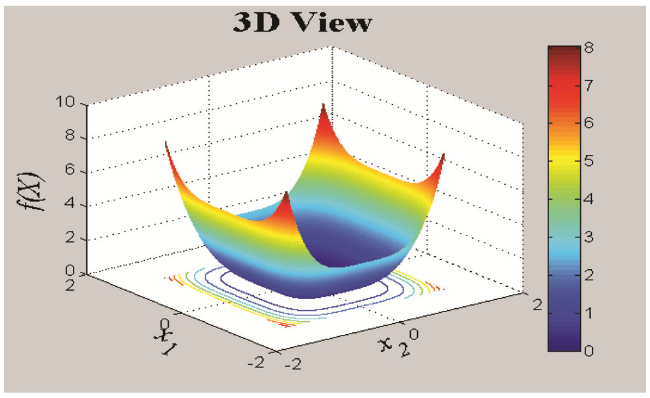

Figure 4. Dejong four (quartic) function.

this benchmark function is in recognition of their effectiveness[10–12] and the need to popularize them among researchers.

The FPA was developed by Yang[7] with inspiration from the pollination characteristics of flowers. Usually flowers are self- or cross-pollinated. Self-pollinated flowers use wind or rain to help them transport pollen grains for pollination. On the other hand, cross-pollinators use animals, insects or other agents for pollination. In the FPA, self-pollinators are deemed to be local search (exploitation), while the cross-pollinators are regarded as global search (exploration). An interplay of the exploitation and exploration stages enables the algorithm to arrive at good solutions[12].

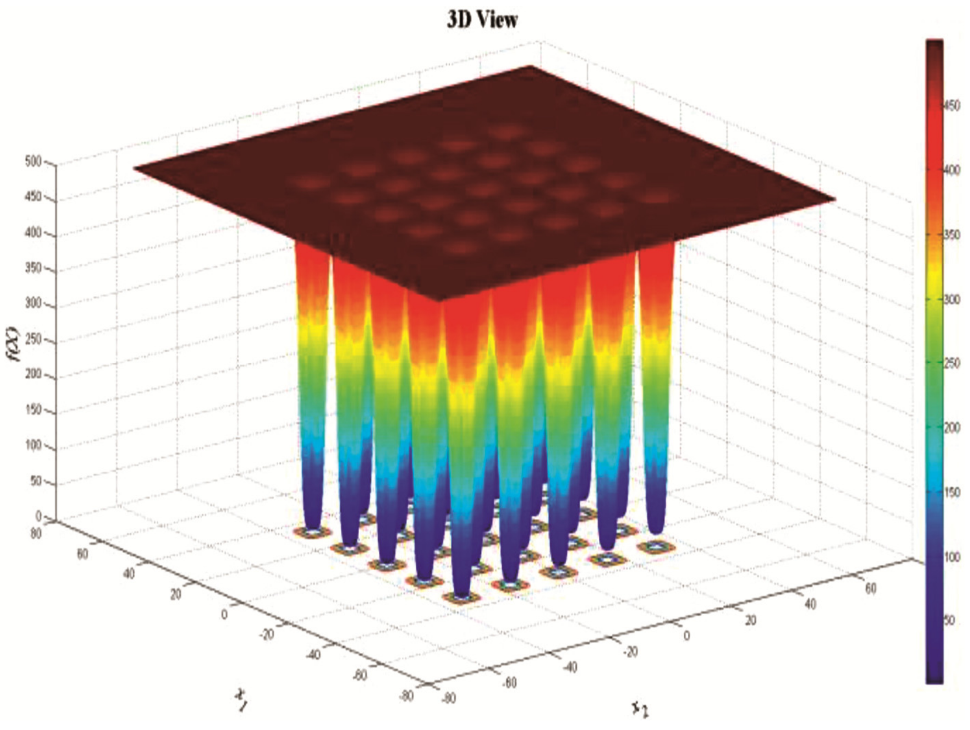

Figure 5. Dejong 5 (shekel's foxholes) function.

```
1.  Begin
2.  Randomly initialize the flowers as well as the pollinators
3.  Evaluate the flowers and pollinators
4.  Ascertain the present best flower
5.  Determine the search constancy probability $p \in [0, 1]$.
6.    While (until termination),
7.          For $j = 1$ to $n$ ($n$ = number of flowers in the population)
8.          If rand < $p$, (rand=random number) then
9.          Use a Levy flight to ascertain the global pollinator
10.         Choose $j$ and $k$ among the obtained solutions and execute
              local pollination (search) with eq. (1)
11.         End if
12.         Ascertain new solutions and evaluate the present flower
              fitness
13.         If new solutions are better than the previous, then
14.         Replace the previous with the new solution
15.         Else retain the previous solution
16.         End if
17.         End for
18.   End while
19. Output best outcome
20. End
```

Figure 6. Pseudocode of the flower pollination algorithm.

In the FPA, global and local pollination (search) processes are controlled by a switch p that has a probability[7]

$$p \in [0, 1]. \tag{1}$$

During the global search, flower constancy is represented by

$$x_i(t+1) = x_i(t) + L(x_i(t) - g^*), \tag{2}$$

where g^* represents the present best solution, $x_i(t)$ denotes pollen i at iteration t along a given vector x_i and L represents Levy flight.

Since its development, the FPA has been successfully applied to solve global optimization problems[7], load frequency control for a hydro-thermal deregulated power systems[13], fractal image compression, Sudoku puzzle, disc brake design problems, retinal vessel segmentation, multi-objective problems, disc brake design problem[14], etc. with good results. Figure 6 presents the pseudo code of the FPA[7].

The ABO was developed by Odili and Kahar in 2015 with inspiration from the movement of African buffalos in their continental landscapes in search of pastures. The African buffalos manage their large herds sometimes numbering more than 1000 animals using two major sounds:/maaa/calling the animals together for a diligent search of the landscape since it holds promise of good pastures, and the /waaa/ calls that caution the animals to get out of a particular location to explore other gazing locations. With judicious use of the /maaa/ (exploit) and /waaa/ (explore) calls, ABO is able to arrive at a good and optimized search results[15]. Figure 7 presents the pseudocode of the ABO.

So far, the ABO has been successfully applied to solve energy and delay routing in mobile ad-hoc networks[16], symmetric TSP[17], asymmetric TSP[18], benchmark global optimization functions[19], numerical function optimization problems, strategic management, collision avoidance in electric fish, tuning of PID parameters of automatic voltage regulators[9], etc. with good results.

In this comparative evaluation of FPA and ABO, the emphasis is on the effect of iteration numbers and search population in obtaining good results. The experiments were implemented using MATLAB on a desktop PC (Intel Duo Core i7 370 CPU @ 3.40 GHz, 3.40 GHz, 4GB RAM, running Windows 10). To ensure fair comparison, both algorithms were run in the same platform and using the same language (MATLAB), same population (10 and 50) and same number of iterations (10, 100, 1000, 5000, 10,000). Also, both algorithms were used to execute five times each the benchmark Dejong 5 function

$$f(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}, \qquad (3)$$

where

$$a_{ij}=\begin{bmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & -32\\ 32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32\end{bmatrix},$$

$$-65{,}536 \le x_i = 65.536,\ i = 1, 2,$$

$$x_i^* \approx -31.97833.$$

Other parameters used for the FPA are $n = 1$, $p = 1$, upper bound (UB) = 2, lower bound (LB) = –2 and $d = 3$. For the ABO, lp1 = 0.7, lp2 = 0.5. The optimal solution of the benchmark Dejong 5 function is

$$f(x) = 0.9980.$$

Table 1 provides details of the experimental outcomes of both algorithms when the population of buffalos or flowers is 10, and the number of iterations ranges from low (10,100) to medium (1000) and high (5000, 10,000). Table 2 presents the experimental outcome of deploying a population of 50 buffalos or flowers, and varying the number of iterations from 10 to 10,000.

A close look at Table 1 shows that both algorithms performed well while searching the solution space with a population of 10 flowers/buffalos. Nonetheless, when the iteration number was 10, the ABO showed better results with an average of 12.827. In fact, one run produced an outcome of 3.3325. This is significant because the optimal solution is 0.9980. The result of the FPA was not as good. It posted an average of 29.980 way off the mark. However, in terms of execution time, the FPA spent an average of 0.026 sec to 0.112 sec in case of the ABO. This shows that the FPA executed faster.

Similarly, when the iteration number was increased to 100, output of the FPA improved significantly to an average of 1.991 compared to 6.814 of the ABO. The execution time of the FPA averaged at 0.137 compared to 0.573 for the ABO: another proof of execution efficiency of the FPA. Similar trend was observed when the iteration count was adjusted to 1000 with the FPA posting an average result of 1.957 compared to 5.481 for the ABO. Nevertheless, effectiveness of the ABO was visible in the algorithm obtaining the optimum solution here (light grey shaded portions in Tables 1 and 2). In terms of execution time, the FPA was still faster with an average of 1.137 sec compared to 5.520 sec for the ABO. With 5000 iterations, the trend was similar as in 1000 iterations. However, remarkable changes were observed when the iteration was adjusted to 10,000. The performance of the ABO improved significantly with three optimal solutions and an average of 0.9981. The FPA however, reversed its excellent run of good results with two optimal solutions and an average of 0.9984.

In view of the above, we can conclude that the FPA has faster execution time than the ABO when using a population of 10 flowers/buffalos. In any case, the ability of the ABO to obtain optimal or near-optimal solution even with as little as 1000 iterations is not in doubt. These findings are in agreement with the 'no free lunch (NFL) Theorem'[20] which states that a parameter of interest to a researcher determines his choice of an optimization algorithm. For this test function, if speed is a more crucial requirement, then the FPA is recommended; but if obtaining the optimal solution as fast as possible is the primary concern, the ABO has an edge.

Next we examine the performance of the algorithms when we deploy 50 search agents (buffalos or flowers). Table 2 presents the experimental outcome.

The behaviour of the algorithms in Table 2 is not much different from the trend displayed when we deployed 10

```
1.  Begin
2.  Initialize the buffalos randomly to different locations within the
      solution space;
3.    While (until termination)
4.        For $j$ = 1: $n$ ($n$ = the population of buffalos)
5.        Ascertain the buffalos exploitation fitness with:
6.        $m'_k = m_k + \mathrm{lp1}(\mathrm{bg} - w_k) + \mathrm{lp2}(\mathrm{bp}_k - w_k)$
7.        where $w_k$ = exploration move; $m_k$ = exploitation move;
            bg = position of the best buffalo; bg is the herd's best
            fitness; lp1 and lp2 denotes learning factors, $\mathrm{bp}_k$ = best
            location of the $k$th buffalo
8.        Update the exploration fitness of buffalos with:
9.        $w'_k = (w_k + m_k)/\lambda$.
10.       Check if $\mathrm{bg}_{\max}$ is updating? Yes, go to 11. If No in 10
            iterations, return to 2
11.   End for
12. End while
13. Output best outcome.
14. End
```

Figure 7. Pseudocode of the African buffalo optimization.

search agents (Table 1). The ABO obtained better outcome (average 5.8331, 2.149, 1.248 respectively) while using 10, 100 and 1000 iterations at an average of 0.29, 2.646 and 26.379 sec respectively. The results (average: 29.980, 3.981, 1.591 respectively) of the FPA were not as good as those of the ABO. However, at 5000 iterations, the FPA had a better average result of 1.001 compared to 1.1305 for the ABO. Just like in the earlier experiment (Table 1), the ABO was able to obtain one optimum solution at 1000 and 5000 iterations, and three optimal solutions at 10,000 iterations. In fact, the ABO had an average of 0.9980, which is the optimum solution at 10,000 iterations. The FPA had only two optimal solutions at 10,000 iterations, with an average of 0.9984.

From the analysis, it is obvious that both algorithms show consistent behaviour when using either 10 or 50 search agents. While the FPA proved to be faster than the ABO in solving the benchmark Dejong 5 function, the ABO had a better run in terms of obtaining the optimal or near-optimal solutions.

In this study we analysed the complex and unpopular benchmark Dejong 5 function using the ABO and the FPA with special focus on the number of populations and iterations required to obtain optimal or near-optimal solutions. The Dejong 5 function has 25 local optimal and 1 global optimum solution. The complexity of this benchmark test function, should be considered as a merit, since it makes the function a good testbed for optimization search algorithms.

After a number of experimental evaluations, this study recommends the usage of 10 search agents (buffalos or flowers) with this test function or any other problems similar to it. This conclusion arises from the need to minimize the use of computer resources and save time while not compromising on search effectiveness. Since the use of computer resources correlates with time spent on solving a problem[21] coupled with the findings of this study that the use of more search agents does not lead to a significant improvement in search outcomes, using a larger population is not recommended.

Both algorithms performed better at 10,000 iterations with the ABO obtaining three optimal solutions when

Table 1. Performance of African buffalo optimization (ABO) and flower pollination algorithm (FPA) when using a population of 10 buffalos/flowers

		ABO		FPA	
Iterations	Population	$f_{\min}$	Time (sec)	$f_{\min}$	Time (sec)
10	10	13.9122	0.243	40.3190	0.025
		22.3328	0.092	60.1138	0.025
		3.3325	0.075	19.2325	0.026
		11.8802	0.075	11.2907	0.027
		12.6747	0.074	18.9458	0.029
Average		**12.827**	**0.112**	**29.980**	**0.026**
100		12.6705	0.574	2.5526	0.173
		7.8740	0.568	1.0754	0.124
		5.7373	0.572	4.3196	0.136
		6.6766	0.581	1.0014	0.125
		1.1096	0.569	1.0034	0.125
Average		**6.814**	**0.573**	**1.991**	**0.137**
1000		2.0259	5.528	3.4697	1.146
		3.8699	5.539	0.9961	1.126
		0.9980	5.521	0.9982	1.143
		10.8268	5.488	3.2507	1.134
		9.6829	5.524	1.0724	1.134
Average		**5.481**	**5.520**	**1.957**	**1.137**
5000		4.9505	27.466	0.9986	5.550
		2.0291	27.639	0.9983	5.565
		0.9980	27.347	0.9981	5.538
		2.9821	27.395	1.0036	5.575
		0.9992	27.395	1.0042	5.517
Average		**2.392**	**27.448**	**1.001**	**5.549**
10000		0.9980	54.655	0.9992	11.097
		0.9980	54.861	0.9983	11.035
		0.9983	55.299	0.9980	11.316
		0.9980	55.129	0.9980	11.335
		0.9981	54.914	0.9984	11.096
Average		**0.9981**	**54.972**	**0.9984**	**11.176**

Table 2. Simulation results using 50 search agents

		ABO		FPA	
Iterations	Population	$f_{\min}$	Time (sec)	$f_{\min}$	Time (sec)
10	50	7.8742	0.295	40.3190	0.025
		3.3238	0.302	60.1138	0.025
		1.2178	0.291	19.2325	0.026
		4.9463	0.292	11.2907	0.027
		11.8032	0.294	18.9458	0.029
Average		**5.8331**	**0.295**	**29.980**	**0.026**
100		3.4537	2.649	2.5526	0.173
		2.0037	2.664	1.0754	0.124
		3.2031	2.634	4.3196	0.136
		1.0819	2.648	1.0014	0.125
		1.0013	2.636	1.0034	0.125
Average		**2.149**	**2.646**	**3.981**	**1.366**
1000		1.2515	26.554	3.4697	1.146
		0.9980	26.309	0.9961	1.126
		1.9921	26.248	0.9982	1.143
		0.9981	26.469	3.2507	1.134
		0.9980	26.315	1.0724	1.134
Average		**1.248**	**26.379**	**1.597**	**1.137**
5000		1.0882	131.066	0.9986	5.550
		0.9980	131.170	0.9983	5.565
		1.0106	131.159	0.9981	5.538
		2.4279	131.407	1.0036	5.575
		0.9987	130.650	1.0042	5.517
Average		**1.1305**	**131.090**	**1.0001**	**5.549**
10000		0.9981	261.595	0.9992	11.097
		0.9980	262.437	0.9983	11.035
		0.9981	262.438	0.9980	11.316
		0.9980	262.674	0.9980	11.335
		0.9980	254.971	0.9984	11.096
Average		**0.9980**	**260.823**	**0.9984**	**11.758**

either using 10 or 50 search agents, while the FPA obtained two optimal solutions each on both counts. However, this study recommends the use of 1000 iterations in order to save CPU processing time. The results obtained when using 1000 iterations are close enough to the optimum to merit this recommendation. Indeed, the ABO was able to obtain the optimum solution while using 10 or 50 buffalos in the search space at 1000 iterations.

Finally, in agreement with the NFL theorem, this study concludes that if speed is the main consideration, then the FPA is recommended. However, if the primary consideration for solving this problem is the need to obtain the optimum solution, then the ABO is recommended.

Conflict of interest: The authors declare that there is no conflict of interest.

1. http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction/quartic.html (accessed on 30 January 2017).
2. De Jong, K. A., Analysis of the behavior of a class of genetic adaptive systems, 1975; https://deepblue.lib.umich.edu/handle/2027.42/4507 (accessed on 20 August 2019).
3. http://www-optima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page1113.htm (accessed on 30 January 2017).
4. Foxholes, S., Electric power systems analysis and nature-inspired optimization algorithms, http://www.al-roomi.org/benchmarks/unconstrained/2-dimensions/7-shekel-s-foxholes-function (accessed on 2 February 2017).
5. http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction/rosenbrock.html (accessed on 30 January 2017).
6. http://www.al-roomi.org/benchmarks/unconstrained/n-dimensions/192-step-function-no-1 (accessed on 30 January 2017).
7. Yang, X.-S., Flower pollination algorithm for global optimization. In International Conference on Unconventional Computing and Natural Computation, Springer, pp. 240–249.
8. Odili, J. B., Kahar, M. N. M. and Anwar, S., African buffalo optimization: a swarm-intelligence technique. *Proc. Comput. Sci.*, 2015, **76**, 443–448.
9. Odili, J. B. and Mohmad Kahar, M. N., African buffalo optimization approach to the design of PID controller in automatic voltage regulator system. In National Conference for Postgraduate Research, Universiti Malaysia Pahang, Malyasia, 2016, pp. 641–648.
10. Odili, J. B., Kahar, M. N. M., Anwar, S. and Azrag, M. A. K., In IEEE 4th International Conference on Software Engineering and Computer Systems (ICSECS), 2015, pp. 90–95.
11. Odili, J. B. and Kahar, M. N. M., Numerical function optimization solutions using the African buffalo optimization algorithm (ABO). *Br. J. Math. Comput. Sci.*, 2015, **10**, 1–12.
12. Yang, X.-S., Karamanoglu, M. and He, X., Flower pollination algorithm: a novel approach for multiobjective optimization. *Eng. Optim.*, 2014, **46**, 1222–1237.
13. Lakshmi, D., Fathima, A. P. and Muthu, R., A novel flower pollination algorithm to solve load frequency control for a hydro-thermal deregulated power system. *Circuits Syst.*, 2016, **7**, 166.
14. Balasubramani, K. and Marcus, K., A study on flower pollination algorithm and its applications. *Int. J. Appl. Innov. Eng. Manage.*, 2014, **3**, 230–235.
15. Odili, J. B. and Kahar, M. N. M., African buffalo optimization (ABO): a new meta-heuristic algorithm. *J. Adv. Appl. Sci.*, 2015, **3**, 101–106.
16. Hassan, M. H. and Muniyandi, R. C., An improved hybrid technique for energy and delay routing in mobile ad hoc networks. *Int. J. Appl. Eng. Res.*, 2017, **12**, 134–139.
17. Odili, J. B., Kahar, M. N. and Noraziah, A., Solving traveling salesman's problem using African buffalo optimization, honey bee mating optimization and Lin-Kerninghan algorithms. *World Appl. Sci. J.*, 2016, **34**, 911–916.
18. Odili, J. B. and Mohmad Kahar, M. N., Solving the traveling salesman's problem using the African buffalo optimization. *Comput. Intell. Neurosci.*, 2016, 1–12.
19. Odili, J. B. and Noraziah, A., African buffalo optimization for global optimization. *Curr. Sci.*, 2018, **114**, 627–636.
20. Wolpert, D. H. and Macready, W. G., No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1997, **1**, 67–82.
21. Khompatraporn, C., Pintér, J. D. and Zabinsky, Z. B., Comparative assessment of algorithms and software for global optimization. *J. Global Optim.*, 2005, **31**, 613–633.

ACKNOWLEDGEMENTS. We thank the Department of Mathematical Sciences, Anchor University Lagos, Nigeria, and the Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Kuantan, Malaysia for providing funds (Grant GRS 1403118).

Received 11 May 2017; revised accepted 20 June 2019

doi: 10.18520/cs/v117/i5/871-877