

How to cite this paper:

Odili, J. B. (2018). Implementation analysis of cuckoo search for the benchmark rosenbrock and levy test functions. *Journal of Information and Communication Technology (JICT)*, 17 (1), 17-32.

IMPLEMENTATION ANALYSIS OF CUCKOO SEARCH FOR THE BENCHMARK ROSEN BROCK AND LEVY TEST FUNCTIONS

Julius Beneoluchi Odili

Faculty of Computer Systems and Software Engineering
Universiti Malaysia Pahang, Malaysia

odili_julest@yahoo.com

ABSTRACT

This paper presents the implementation analysis of the benchmark Rosenbrock and Levy test functions using the Cuckoo Search with emphasis on the effect of the search population and iterations count in the algorithm's search processes. After many experimental procedures, this study revealed that deploying a population of 10 nests is sufficient to obtain acceptable solutions to the Rosenbrock and Levy test functions (or any similar problem to these test landscapes). In fact, increasing the search population to 25 or more nests was a demerit to the Cuckoo Search as it resulted in increased processing overhead without any improvement in processing outcomes. In terms of the iteration count, it was discovered that the Cuckoo Search could obtain satisfactory results in as little as 100 iterations. The outcome of this study is beneficial to the research community as it helps in facilitating the choice of parameters whenever one is confronted with similar problems.

Keywords: Cuckoo search, iteration, Levy function, population, Rosenbrock function.

IMPLEMENTATION ANALYSIS OF CUCKOO SEARCH FOR THE BENCHMARK ROSEN BROCK AND LEVY TEST FUNCTIONS

Julius Beneoluchi Odili

*Faculty of Computer Systems and Software Engineering
Universiti Malaysia Pahang, Malaysia*

odili_julest@yahoo.com

ABSTRACT

This paper presents the implementation analysis of the benchmark Rosenbrock and Levy test functions using the Cuckoo Search with emphasis on the effect of the search population and iterations count in the algorithm's search processes. After many experimental procedures, this study revealed that deploying a population of 10 nests is sufficient to obtain acceptable solutions to the Rosenbrock and Levy test functions (or any similar problem to these test landscapes). In fact, increasing the search population to 25 or more nests was a demerit to the Cuckoo Search as it resulted in increased processing overhead without any improvement in processing outcomes. In terms of the iteration count, it was discovered that the Cuckoo Search could obtain satisfactory results in as little as 100 iterations. The outcome of this study is beneficial to the research community as it helps in facilitating the choice of parameters whenever one is confronted with similar problems.

Keywords: Cuckoo search, iteration, Levy function, population, Rosenbrock function.

INTRODUCTION

The scientific community has adduced several reasons for the popularity of optimization among researchers since the second half of the 20th century. One of the reasons for this popularity is due to the impact of optimization on some very remarkable scientific and technological breakthroughs the world has

experienced since the advent of the optimization field of knowledge. Some of the areas where the application of optimization principles have been beneficial to mankind includes decision-making (Odili, 2013b), aviation (West et al., 2012), job scheduling (TaHERI, Lee, Zomaya, & Siegel, 2013), vehicle routing (Odili, Kahar, & Anwar, 2015), product assembly plants (D. Yang et al., 2015), parameter-tuning of Proportional Integral and Derivatives Controllers in Automatic Voltage Regulators (Odili & Mohmad Kahar, 2016), etc.

Optimization which is generally a method/technique of getting the maximum outcome from a minimum input could be traceable to the works of early 20th century scientists like John Holland who designed the Genetic Algorithm (Holland, 1992) and Karl Menger who designed the first mathematical formulation of the travelling salesman's problem in the early 1930s (Odili, 2013a). The impact of the works of these early scientists has revolutionized the field of optimization, making it a favored area of scientific investigations.

The development of optimization has led to the development of several optimization techniques that drew their inspiration from various sources ranging from physics, chemistry and biology to other natural phenomena common to man. Some of the most popular optimization techniques are those drawn from the biological processes in plants, man and animals. Some of these popular techniques include the Genetic Algorithm (Holland, 1992), Particle Swarm Optimization (Kennedy, 2011), Ant Colony Optimization (Dorigo & Gambardella, 2016), etc. In the past ten years, some methods have been developed which have proven to be very successful and sometimes more effective than the earlier techniques. Some of these new techniques are the Cuckoo Search (X.-S. Yang, 2012b), Flower Pollination Algorithm (X.-S. Yang, 2012) and African Buffalo Optimization (Odili & Kahar, 2015), etc.

Our interest in this study was born out popularity due to its effectiveness and efficiency in the Cuckoo Search. Though a relatively newly designed technique, the Cuckoo search has enjoyed wide applicability. This study aimed to investigate the effect of the search population as well as the number of iterations needed to obtain very good solutions in the Cuckoo Search. It was our aim that coupled with making the Cuckoo Search more user-friendly, the outcome of this study would benefit the scientific community in terms of parameters-tuning when they are required to solve optimization problems using the Cuckoo Search. Similarly, our choice of the Rosenbrock function as the target of this diagnostic evaluation was due to its popularity among researchers due to its complex nature. The benchmark Rosenbrock function being of the one of the five functions developed by Kenneth DeJong in his PhD thesis in 1975, has become very popular due to its flat surface that tends

to provide insufficient information to many search agents. Similarly, the growing popularity of the benchmark Levy function is a motivation for its choice in this study. As a result of the deceptive landscape of the Levy and Rosenbrock functions, both functions are gradually becoming favorite test cases to many researchers when investigating the search capability of new optimization algorithms (De Jong, 1975).

CUCKOO SEARCH

Cuckoo Search (CS) is an optimization algorithm developed from careful observation, mathematical modelling of the craftiness of the cuckoo bird in the egg incubation process. The cuckoo birds being lazy and irresponsible do not like the laborious egg- incubating process so they rather prefer to lay their eggs among the eggs of other birds or other cuckoo species. The host birds, with a certain probability (randomness), may incubate the cuckoo eggs along with theirs (exploitation), discover the strange eggs and either abandon their nests or throw the strange eggs away (exploration) (X.-S. Yang & Deb, 2009).

In this algorithm (the CS), the eggs of the host bird in any given nest represents an optimization solution, while the strange eggs of the cuckoo birds represent new solutions. Through careful manipulation of the cuckoo eggs and those of the host birds, the CS is able to arrive at good optimization solutions to complex optimization problems (X.-S. Yang & Deb, 2009).

Since its development, the CS has enjoyed wide applications to various optimization problems. Some of the successful application areas of the CS includes the travelling salesman's problems, wireless sensor networks, job scheduling, image processing, flood forecasting, classification task in the health sector, etc. (Anwar et al., 2017; Kamat & Karegowda, 2014). The pseudocode of the CS (Agrawal, Panda, Bhuyan, & Panigrahi, 2013) is presented below:

1. **Begin**
2. Objective function: $f(x)$ $x = (x_1, x_2 \dots x_n)$
3. Randomly initialize the nest in the search space
4. **While** (not termination), do

```

5.           For  $i=1$  to , do
6.               Generate a cuckoo randomly through Levy flight by using
7.                $X_{ij}(t + 1) = X_{ij}(t) + \alpha \text{Levy}(\lambda)$ 
8.               Ascertain the fitness of the generated cuckoo
9.               Randomly select a nest among the host nests available
10.            If ( $f_i > f_k$ ) then
11.                Replace  $k$  with the better solution
12.            End if
13.            Abandon some of the unfruitful nests and generate newer ones
14.            Retain the good solutions found
15.            Rank the newly-found good solutions
16.            Determine the current overall best
17.            End for
18.        End while
19.        Output the best outcome
20.    End

```

The Pseudocode of Cuckoo Search

IMPLEMENTATION EVALUATION OF CUCKOO ROSENBROCK

Since the focus of the first part of this paper was to determine the effect of the search population-cum-number of iterations required to obtain the best output to the Rosenbrock and the second part was to examine the same in Levy test functions (and by implication, other similar problems), it was necessary for the sake of fairness to run the experiments in the same machine. The experiments in this study were performed on a PC, 4GB RAM, Intel Duo Core i7 370 CPU @ 3.40GHz, 3.40GH, Windows 10 OS. The population of nests was 10 and 50. Also, the number of iterations included 10, 20, 100, 1000, 5000, and 10,000. The CS parameters used for the experiments were $u = \text{rand}(\text{size}(s)) * \sigma$; $v = \text{rand}(\text{size}(s))$; $pa = 0.5$; $\text{step} = u / \text{abs}(v) \wedge (1/\beta)$; $\text{step size} = 0.01 * \text{step}$. Each experiment test case was executed five times. The benchmark Rosenbrock function (Shi & Eberhart, 1999) was

$$f(x) = \sum_{i=1}^{d-1} [(100 x_i - x_i^2)^2 + (x_i + 1)^2] \quad (1)$$

It is important to note that the optimum solution to the Rosenbrock test function (see Figure 1) is:

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

The simulation outcomes obtained after a number of experimental evaluations using the CS algorithm with search populations of 10 nests as well as different numbers of iterations ranging from 10 to 10,000 are shown in Table 1.

Table 1

Comparative Search with 10 Population (Nnests)

Iterations	f_{min}	Average	Time (secs)	Average Time (s)
10	2.3080		0.040	
	0.0013		0.031	0.031
	0.0329	0.5634	0.033	
	0.4738		0.034	
	0.0012		0.018	
100	$3.9731e^{-13}$		0.172	
	$9.8137e^{-14}$		0.162	0.163
	$4.4142e^{-12}$	$5.0611e^{-13}$	0.154	
	$1.5596e^{-13}$		0.170	
	$5.5449e^{-15}$		0.157	
1000	$4.6147e^{-85}$		1.565	
	$1.0024e^{-83}$		1.556	1.5874
	$4.2801e^{-78}$	$4.4527e^{-78}$	1.563	
	$4.2170e^{-69}$		1.600	
	$8.1493e^{-75}$		1.653	
5000	0		8.118	
	$3.0304e^{-318}$		8.086	7.9480
	$4.3782e^{-318}$	$2.4697e^{-320}$	7.867	
	0		7.848	
	$4.9407e^{-324}$		7.821	
10000	0		15.372	
	0		15.586	15.761
	0	0	15.683	
	0		15.979	
	0		16.183	

A close look at Table 1 reveals that the CS algorithm obtained the best result in iteration 10,000. It obtained the optimum solution in all runs when searching with a population of 10 nests. A commendable feat, no doubt, since stochastic optimization algorithms, generally, do not guarantee optimal solutions. Though obtaining the optimal result here was commendable another examination reveals that it was obtained at an average of 15.751 seconds. Comparing this result with the ones obtained when the iteration counts were just 10 (mean: $0.5631e^{-13}$) at an average of 0.031 seconds or 100 iterations (mean: 5.0611) at an average of 0.163 seconds, it could be argued that the result obtained at 100 iterations was by far cheaper and, therefore, better. This line of argument is in tandem with the conclusions of an earlier study that a good trade-off in terms of time and output is a mark of a good optimization algorithm (Khompataporn, Pintér, & Zabinsky, 2005).

In the light of the above discussion, this study recommends that in using the CS to solve the Rosenbrock test function (or a similar optimization problem) when the search population is 10, a good enough result is obtainable at iteration 100 in order to save time since the amount of time used to obtain the solution correlates with the use of computer resources. The exception to this recommendation would be in a situation where the main consideration is the ability to obtain the optimum result. If obtaining the optimum solution is the primary concern, then the CS obtains the best result (when solving this particular problem and using the above parameters set) at iteration 10,000 as can be seen in Table 1. It must be emphasized that the results obtained when deploying 1000 and 5000 iterations are also very close to the optimum.

To conclude this part, it is necessary to examine the experimental output when a population of 50 nests are used. The simulation results obtained by using 50 nests and different iteration counts from 10, 100, 1000, 5000 to 10,000 are shown in Table 2.

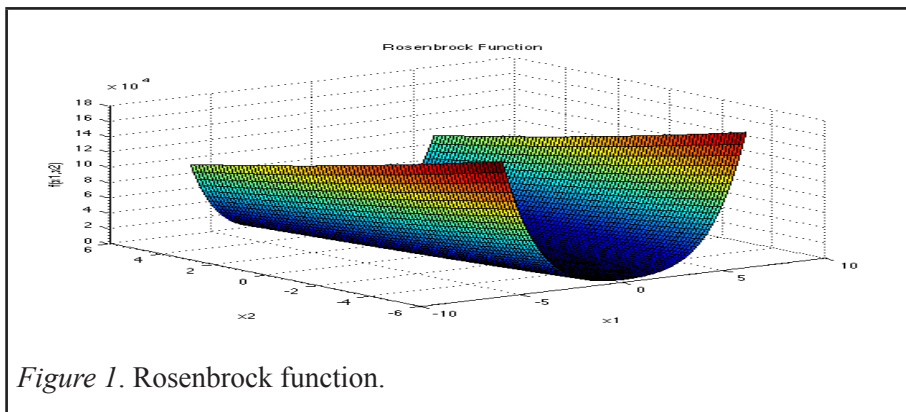


Table 2

Comparative Search with 50 Population (Nests)

Iterations		Average	Time (secs)	Average Time (s)
10	0.1048		0.071	
	1.1899		0.068	0.0784
	0.0314	1.0334	0.068	
	2.9252		0.068	
	0.9157		0.117	
100	$4.1464e^{-14}$		0.172	
	$6.8572e^{-14}$		0.162	0.163
	$2.6223e^{-16}$	$3.8376e^{-15}$	0.154	
	$2.7811e^{-15}$		0.170	
	$2.7811e^{-16}$		0.157	
1000	$1.3143e^{-54}$		6.231	
	$5.4190e^{-56}$		6.257	6.310
	$1.6071e^{-56}$	$2.2048e^{-55}$	6.364	
	$1.0727e^{-55}$		6.384	
	$1.6111e^{-54}$		6.313	
5000	$6.8572e^{-161}$		26.775	
	$2.6223e^{-169}$		25.366	30.042
	$2.7811e^{-165}$	$4.9646e^{-165}$	31.439	
	$7.3016e^{-165}$		32.744	
	$5.2609e^{-167}$		33.884	
10000	$2.0759e^{-269}$		67.500	
	$6.5280e^{-272}$		68.044	68.441
	$8.5684e^{-271}$	$5.3958e^{-272}$	68.585	
	$8.2255e^{-269}$		70.631	
	$1.5813e^{-279}$		67.447	

It is interesting to note that the experimental results of Table 2 when 50 population of nests are used are inferior to those of Table 1 that uses less numbers of search agents. This finding is remarkable because using more search agents leads to more evaluations per iterations and so much time is

taken to obtain results. In spite of so much time being taken, the results are not superior. In fact, at no iteration was the optimum solution obtained. This emphasizes the need of this study to avoid unnecessary waste of computer resources.

IMPLEMENTATION OF THE LEVY FUNCTION

The focus of the second part of this paper was to examine the implementation strategies of the Levy function. The choice of this function was an attempt to popularize this extremely useful benchmark function despite its complexity. This function (see Figure 1) has several peaks and multiple minima and optima solutions. The determination of the global minima is a good test for an optimization algorithm.

In this study, our major considerations were the determination of the effect of the search population-cum-number of iterations required to obtain the best output of this test function and, by implication, other similar problems. Each experiment test case was executed five times.

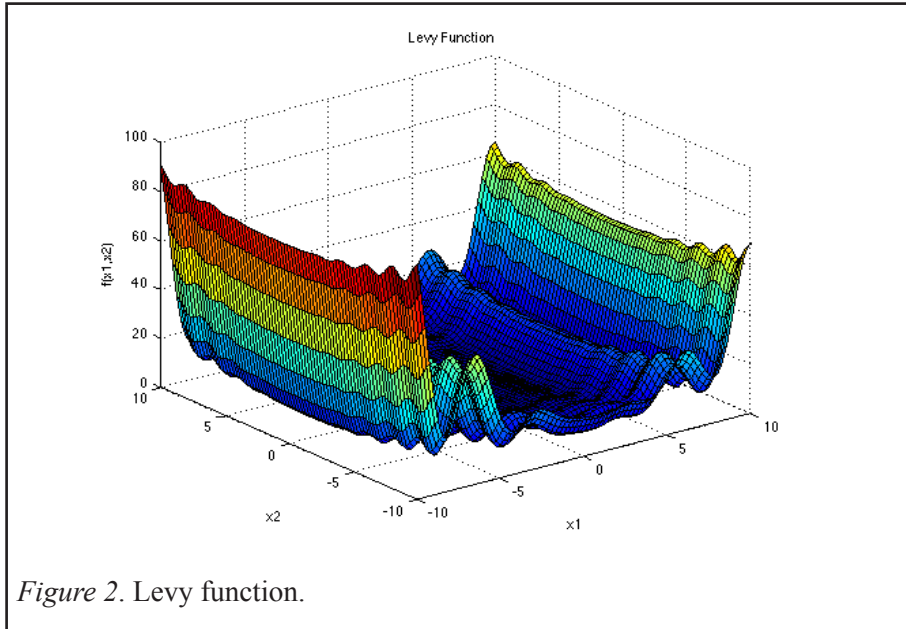


Figure 2. Levy function.

The mathematical description of the Levy function is:

$$f(x) = \sin^2(\pi\omega_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 + [10 + \sin^2(\pi\omega_1 + 1)] + (w_d - 1)^2 [1 + \sin^2(2\pi(w_d))] \quad (3)$$

In Eq. 3, $w_i = 1 + \frac{x_i-1}{d}$ and $i = 1-d$. Moreover, the benchmark Levy function is normally evaluated on a hypercube with $x_i \in [-10, 10]$, for all $i = 1, \dots, d$. The global minimum is:

$$f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (1, 1) \tag{4}$$

The simulation outcome is presented in Table 3.

Table 3

Simulation Outcome of CS on Levy

<i>Population</i>	<i>Iterations</i>	f_{min}	<i>Time (s)</i>
10	100	3.64E-06	0.133
		5.76E-06	0.097
		5.29E-06	0.134
		9.34E-06	0.131
		2.62E-06	0.11
	200	7.84E-06	0.135
		7.57E-06	1.892
		9.39E-06	0.117
		8.07E-06	0.102
		1.06E-06	0.115
	500	5.29E-06	0.109
		2.18E-06	0.133
		9.34E-06	0.133
		2.62E-06	0.104
		7.84E-06	0.119
	2000	9.39E-06	0.143
		8.07E-06	0.1
		1.06E-06	0.101
		5.22E-06	0.111
		6.37E-06	0.126
10,000	7.69E-07	0.126	
	9.29E-06	0.128	
	8.36E-06	0.161	
	7.13E-06	0.116	
	7.66E-06	0.133	

(continued)

<i>Population</i>	<i>Iterations</i>	f_{min}	<i>Time (s)</i>
10	100,00	8.47E-06	0.146
		7.77E-06	0.111
		2.72E-06	0.141
		7.46E-06	0.114
		1.75E-06	0.122

A close examination of Table 4 reveals that the CS was very efficient and effective in solving the benchmark Levy function. It is particularly interesting to note the speed of the algorithm in arriving at a solution. Even when 100,000 iterations were deployed the algorithm was able to converge at a solution in less than a second. This is a mark of the efficiency of CS in the use of computer resources in its search for solutions.

Another remarkable outcome of this investigation, especially when a population of 10 nests were used (see Table 3) was that in spite of the number of iterations deployed, the CS was able to obtain a solution close enough to the global optima of 0 (see Eq. 4). This is an indication of the effectiveness of the CS. Though a recently developed optimization algorithm, the effectiveness of the algorithm as can be seen in this study, may be the primary reason for its popularity among researchers and its consequent wide applicability.

Moreover, Table 3 shows that the least convergence time (0.097 second) was when 100 iterations were used. On the other hand, the highest time taken to converge at a solution was 0.161 second at iteration 10,000. This finding is consistent with existing literature: the use of more iterations in search of solutions leads to longer convergence periods. Based on this, it may be safe to conclude that the CS algorithm follows the same trend. However, it is worthy to note that in 100,000 iterations, the highest convergence time was 0.141 second which was faster than the highest observed in 10,000 iterations (0.161 second). Statistically, it could be argued that the high convergence time of 0.161 second observed in 10,000 iterations could be rogue. Since rogue data are not sufficient ground to invalidate a research finding, this study concludes that if more iterations are deployed in search, more likely, more time is taken to converge at a solution. In the light of this conclusion, it is hereby recommended that when using the CS to solve the benchmark Levy function (or a similar problem to the benchmark Levy function), the

use of a population of 10 nests and 100 iterations may be a desirable choice. In the next set of experiments, 25 search population (nests) were deployed to the search space. Moreover, we added the 1000 and 5000 iterations to the ones used in Table 3. The simulation results of this last set of experiments are presented in Table 4.

Table 4

CS on Levy Function When Using 25 Nests

<i>Population</i>	<i>Iterations</i>	f_{min}	<i>Time (s)</i>
25	100	9.48E-06	0.255
		9.01E-06	0.222
		1.28E-06	0.191
		5.42E-06	0.241
		2.81E-06	0.253
	200	5.45E-06	0.247
		9.06E-06	0.241
		8.88E-06	0.256
		9.08E-06	0.269
		9.01E-06	0.184
	500	6.71E-06	0.268
		6.57E-06	0.254
		9.64E-06	0.237
		9.39E-06	0.214
		5.45E-06	0.203
	1000	6.23E-06	0.316
		6.21E-06	0.237
		7.30E-06	0.269
		6.90E-06	0.274
		8.82E-06	0.231
2000	2.46E-06	0.217	
	2.35E-06	0.221	
	6.92E-06	0.219	
	5.66E-06	0.239	
	7.18E-06	0.107	

(continued)

<i>Population</i>	<i>Iterations</i>	f_{min}	<i>Time (s)</i>
25	5000	1.46E-06	0.28
		3.21E-06	0.257
		9.22E-06	0.242
		6.55E-06	0.176
		7.41E-06	0.223
	10000	4.12E-06	0.279
		8.57E-06	0.274
		6.10E-06	0.208
		3.04E-06	0.198
		9.35E-06	0.205
	100000	6.99E-06	0.189
		7.95E-06	0.231
		8.91E-06	0.251
		9.56E-06	0.317
		6.48E-06	0.227

The experimental output recorded in Table 4 follows the same trend of Table 3. The increase in the number of iterations did not translate to improved results. Again, it is obvious that the use of a higher number of nests (25) leads to an increase in convergence time.

In the light of the above, since the use of more search agents did not improve the simulation outcomes but rather increased the processing overhead, the use of 10 search agents is sufficient to obtain optimal or near-optimal solutions. Again, it can be observed in Table 4 that at no point did the CS obtain the optimal solution. Consequently, the use of 100 iterations is recommended.

CONCLUSIONS

This paper presented the diagnostic evaluation of the effects of search population and the number of iterations in solving the benchmark Rosenbrock and Levy functions using the CS. After a number of experimental procedures, it was discovered that in solving the benchmark function as well as the benchmark Levy function, the CS obtained better results when a population of 10 nests were deployed to the search space. By using 10 nests, near-optimal solutions were obtainable from as low as 100 to 10,000 iterations. In fact, at 10,000 iterations, the CS obtained the optimum solution to Rosenbrock function in every run. Surprisingly, however, this study discovered that increasing the

search population to 25 and 50 nests did not improve the solution. It rather worsened the result and increased the processing time.

Similarly, for the Levy function, the near-optimal solution was obtainable at 100 iterations and with a search population of 10 nests. The use of more iterations-cum-nests was discovered to only increase the processing overhead without the expected improvement in results

In the light of the findings of this paper, it is recommended that when solving the Rosenbrock and Levy functions or any similar problems, using a population of 10 nests will give an optimum or near-optimum solution. In terms of the required number of iterations, this study recommends from 100 to 10,000 depending on the primary considerations. If the main consideration is the optimum solution, then using 10,000 iterations will almost certainly guarantee the optimum result to the Rosenbrock function. Results obtainable from 100 to 5000 iterations are near the optimum too. Conversely, for the Levy function, the use of 100 iterations is sufficient to give near-optimal solutions.

Before this study, it was believed by some researchers that the more populations deployed to the search space, the more likely it was to obtain optimal results. This study proved otherwise. Based on this finding, it is recommended that further implementation evaluations be carried out on the Rosenbrock and Levy functions using different platforms and parameter-set to validate the findings of this study.

REFERENCES

- Agrawal, S., Panda, R., Bhuyan, S., & Panigrahi, B. K. (2013). Tsallis entropy-based optimal multilevel thresholding using cuckoo search algorithm. *Swarm and Evolutionary Computation, 11*, 16-30.
- Anwar, S., Inayat, Z., Zolkipli, M. F., Zain, J. M., Gani, A., Anuar, N. B., Chang, V. (2017). Cross-VM cache-based side channel attacks and proposed prevention mechanisms: A survey. *Journal of Network and Computer Applications*.
- De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. (Doctoral dissertation, University of Michigan). *Dissertation Abstracts International, 36*(10), 5140B. (University Microfilms No. 76-9381)

- Dorigo, M., & Gambardella, L. (2016). *Ant-Q: A reinforcement learning approach to the traveling salesman problem*. Paper presented at the Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning.
- Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267(1), 66-72.
- Kamat, S., & Karegowda, A. (2014). A brief survey on cuckoo search applications. *Int. J. Innovative Res. Comput. Commun. Eng*, 2(2).
- Kennedy, J. (2011). Particle swarm optimization. *Encyclopedia of Machine Learning* (pp. 760-766): Springer.
- Khompatraporn, C., Pintér, J. D., & Zabinsky, Z. B. (2005). Comparative assessment of algorithms and software for global optimization. *Journal of Global Optimization*, 31(4), 613-633.
- Odili, J. B. (2013a). Application of ant colony optimization to solving the traveling salesman's problem. *Science Journal of Electrical & Electronic Engineering*, 2013.
- Odili, J. B. (2013b). Staff development programs and job performance: Implications for productivity in Lagos state ministry of education. *Journal of Business Administration and Management Sciences Research*, 2(12), 330-334.
- Odili, J. B., & Kahar, M. N. M. (2015). African buffalo optimization (ABO): A new meta-heuristic algorithm. *Journal of Advanced & Applied Sciences*, 3, 101-106.
- Odili, J. B., Kahar, M. N. M., & Anwar, S. (2015). African buffalo optimization: A swarm-intelligence technique. *Procedia Computer Science*, 76, 443-448.
- Odili, J. B., & Mohmad Kahar, M. N. (2016). 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*, 641-648.
- Shi, Y., & Eberhart, R. C. (1999). *Empirical study of particle swarm optimization*. Paper presented at the Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on.

- Taheri, J., Lee, Y. C., Zomaya, A. Y., & Siegel, H. J. (2013). A bee colony-based optimization approach for simultaneous job scheduling and data replication in grid environments. *Computers & Operations Research*, 40(6), 1564-1578.
- West, P., Sculli, G., Fore, A., Okam, N., Dunlap, C., Neily, J., & Mills, P. (2012). Improving patient safety and optimizing nursing teamwork using crew resource management techniques. *Journal of Nursing Administration*, 42(1), 15-20.
- Yang, D., Jiao, J. R., Ji, Y., Du, G., Helo, P., & Valente, A. (2015). Joint optimization for coordinated configuration of product families and supply chains by a leader-follower Stackelberg game. *European Journal of Operational Research*, 246(1), 263-280.
- Yang, X.-S. (2012a). *Flower pollination algorithm for global optimization*. Paper presented at the International Conference on Unconventional Computing and Natural Computation.
- Yang, X.-S. (2012b). Nature-inspired metaheuristic algorithms: Success and new challenges. *arXiv preprint arXiv:1211.6658*.
- Yang, X.-S., & Deb, S. (2009). *Cuckoo search via Lévy flights*. Paper presented at the Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on Nature & Biologically Inspired Computing.

