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

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

Volume 2, Pages 376-394

ICRES 2018: International Conference on Research in Education and Science

A Critical Review of Major Nature-Inspired Optimization Algorithms

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: Nature-inspired Algorithms are getting more and more popular in the past few decades owing to their amazing successes in solving a number of real-world optimization problems in different spheres of human endeavour ranging from the financial, medical and industrial to educational applications etc. Nature-inspired Algorithms (NAs) simulate the harmonious cooperation and competition in nature resulting in amazing solutions to seemingly impossible human problems. This paper examines ten nature-inspired techniques and their applications to different fields of human endeavour and concludes that the Nature-inspired Algorithms has enormous promise in the quest for greater human development through the harnessing of NAs' potentials for speeding-up of industrial processes, minimization of time, financial and computer resources required to obtain solutions to complex optimization problems etc. However, the study points out certain areas of concern in the development of the NAs, namely, the apparent lack of clear mathematical cum theoretical proofs of convergence of these algorithms, manual tuning of parameters and the recurring issue of experimenting with small-scale problems vi-a-vis the large and complex real-life problems.

Keywords: Nature-inspired algorithms, Metaheuristics, Trajectory-based, Population-based, Deterministic algorithms

Introduction

There has been an upsurge in the number of research studies in Artificial Intelligence and specifically in Nature-inspired Algorithms (NAs) in the past few decades. Such extensive studies have led to the design and development of several optimization algorithms such as the Great Deluge (Dueck, 1993), Hill Climbing (Selman & Gomes, 2006), Simulated Annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), Particle Swarm Optimization (Kunna, Kadir, Jaber, & Odili, 2015), Artificial Bee Colony (Karaboga & Aslan, 2015), Bee Colony Optimization (Teodorović & Dell'Orco, 2005), Firefly Optimization (X.-S. Yang, 2009a). Before the birth of NAs, the deterministic algorithms such as the finite difference (Forsythe & Wasow, 1960), Raphson-Newton (Pletcher, Minkowycz, Sparrow, & Schneider, 1988), Nelder–Mead Simplex (Lagarias, Reeds, Wright, & Wright, 1998), Hook-Jeeves optimization methods (Aghamohammadi & Pourgholi, 2008) etc. were the dominant optimization algorithms for scientific investigations. The application areas of the NAs range from logistics (Vassiliadis & Dounias, 2009), image processing (Cuevas & Sossa, 2013), signature verification (Selman & Gomes, 2006), engineering designs (X.-S. Yang, 2005), financial applications (Brabazon, 2008), protein folding problems (Burke, Bykov, & Hirst, 2007) to business optimization (X.-S. Yang, Deb, & Fong, 2011) etc. The successful applications of NAs to these non-lineal and complex optimization problems far above the performance of the deterministic models in terms of computational time used to achieve results and their capacity to obtain optimal or near-optimal solutions to complex optimization problems is a motivation for further research and this study in particular.

The success of NAs over the deterministic algorithms is as a result of the methodology applied by NAs in search for solutions to complex multimodal problems (X.-S. Yang, 2009b). Generally, NAs apply randomization mechanism which enable them construct solutions iteratively over a number of construction steps. Moreover, they are relatively simple algorithms in terms of their fitness updates and implementation strategies. For these reasons, the NAs are user-friendly. This recommends them to wide applicability by professionals in diverse research and application fields. Furthermore, NAs are general-purpose algorithms that can be applied to solve different kinds of problems. To achieve this, the primary requirement is the proper tuning of parameters to suit a particular problem at hand. This cannot be said of deterministic algorithms that are rather specialized, though capable of obtaining the optimal solutions.

The rest of this paper is organised thus: section two attempts a classification of NAs, highlighting the working of some of these algorithms and their areas of successful applications, strengths and weaknesses; section three discusses the findings of this review; section four draws conclusion on the study and section five acknowledges support for the study.

Classification of Nature-Inspired Metaheurıstic Algorithms

A broad way to classify optimization algorithms is to identify them as either Stochastic or Deterministic, in terms of their solution dynamics (Venter, 2010). The deterministic algorithms were designed in such a way that they are able to obtain the optimal solution but their stochastic counterparts cannot guarantee the optimal results but could solve larger and more complex problems within a relatively short time. A popular general-purpose deterministic algorithm, for instance, is the DIRECT algorithm which makes use of Lipschitzian technique to identify promising sub-regions in the search space. (Olafsson, 2006). For the purpose of this paper, however, our interest is the stochastic optimization techniques which are usually nature-inspired. Nature inspired Algorithms (NAs) simulate nature's way of solving its own problems and this has opened a new vista in computer science.

Furthermore, NAs can be classified as Bio-inspired, Environment-based or Physics/Chemistry-based (Binitha & Sathya, 2012). As their name implies, Bio-inspired algorithms (BAs) are inspired by the biological processes of nature such as the behavior of plants and animals, evolutionary trends in nature and ecology. The Evolutionary algorithms arm of BAs, on the one hand, is broadly inspired by the Darwin theory of Evolution (Crawford & Krebs, 2013). Examples of the algorithms in this category are the Genetic Algorithm (Whitley, 1994), Evolutionary Strategies (Tyrrell, Hollingworth, & Smith, 2001), Genetic Programming (Banzhaf, Nordin, Keller, & Francone, 1998) etc. Another sub class of BAs is the Swarm-based algorithms which are inspired by the cooperative or competitive behavior of animals in our environment. Examples of the Swarm-based algorithms are the Particle Swarm Optimization (Kennedy, 2010), Ant Colony Optimization (Dorigo, Birattari, & Stützle, 2006) etc. The last subclass of BAs is the Ecology-based algorithms which are inspired by the natural ecosystem. Examples are the Biogeography-based Optimization (Simon, 2008), Invasive Weed Optimization (Mehrabian & Lucas, 2006), Symbiosis algorithms (Binitha & Sathya, 2012) etc. At this juncture, let us attempt taxonomy of the Nature-inspired algorithms. This effort is by no means exhaustive but will give a clearer picture of these optimization techniques (Please see Figure 1).

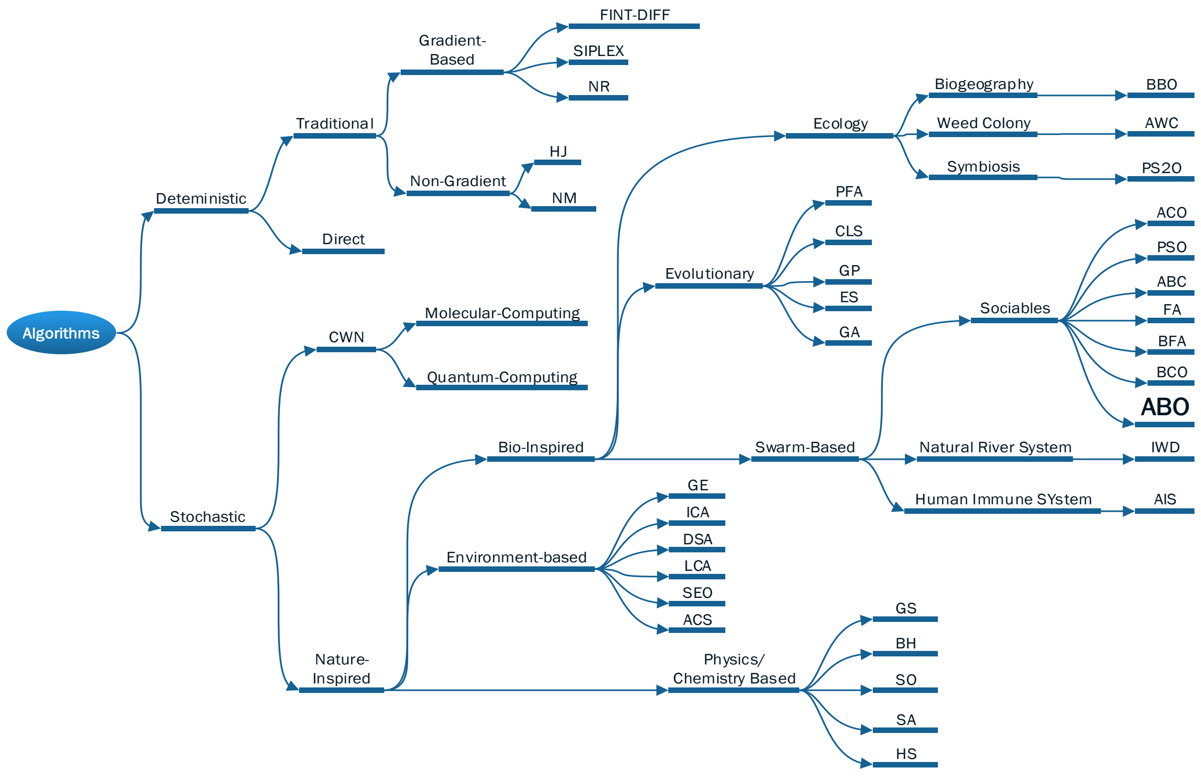

Figure 1. Taxonomy of algorithms

CWN=Computing with Nature
FINT-DIFF =Finite Difference
GA= Genetic Algorithm
N.R = Newton-Raphson method
CLS= Classifier Systems
GP= Genetic Programming
LCA= League Championship Algorithm
DSA= Differential Search Algorithm
ICA= Imperialist Competitive Algorithm
GE= Grammatical Evolution
GS= Gravitational Search
BH= Black Hole
SO= Spiral Optimization
SA= Simulated Annealing
HS= Harmony Search
ABO= African Buffalo Optimization
SEO= Search Engine Optimization

ACS= Ant Colony System
HJ= Hooke-Jeeves pattern
NM= Nelder-Mead simplex
ES= Evolution Strategies
BBO= Biogeography-Based Optimization
FA= Firefly Algorithm
PFA= Paddy Field Algorithm
BCO= Bee Colony Optimization
AWC= Artificial Weed Colony
BFA=Bacteria Foraging Algorithm
IWD= Intelligent Water Drop
AIS= Artificial Immune System
ACO= Ant Colony Optimization
PSO= Particle Swarm Optimization
ABC= Artificial Bee Colony
PS20= 2-Level PSO

Another way to classify Nature-inspired stochastic algorithms could be to categorize them as either population-based or trajectory-based (X.-S. Yang, 2009b). Population-based optimization techniques solve problems using a number of search agents which also result in obtaining a number of solutions in a particular iteration. Such population-based search algorithms are usually designed in such a way as to permit elitism which enables the algorithms to select the best solution amongst the several generated solutions. This way, these algorithms build solutions incrementally (Dorigo, Caro, & Gambardella, 1999). Population-based algorithms are sometimes, called exploration-based methods since they are exceptionally good in the diversification of the search space. An example of this is the GA which uses a set of strings (Whitley, 1994). So also is Particle Swarm Optimization that uses a number of agents or particles (Kennedy, 2010). On the other hand, Simulated Annealing, Great Deluge and Hill Climbing, Tabu Search etc. are trajectory-based and use a single agent that moves through the search space in a zigzag fashion as the iterations continue (Kennedy, 2010; Mirjalili, Mirjalili, & Lewis, 2014). In trajectory-based methods, the initial solution is, usually, obtained randomly and then improved upon iteratively. The difference between the Population-based and the Trajectory-based metaheuristics is primarily in the number of temporal solutions applied in an iteration of the search. While the population-based methods use a number of agents and therefore, solutions, the Trajectory-based methods use a single search agent and, as such, generate a single solution per iteration.

Trajectory-based Algorithms

Trajectory-based algorithms start with a single solution and as the particular algorithm progresses; such a single solution is improved upon and replaced by the current solution. Some experts call them exploitation-based algorithms since they usually emphasize intensification of the search space (Manjarres et al., 2013). The search agent simply moves through the search space tracing the path to the optimal solution. This random movement through the solution space within the neighborhood of the current solution to another continues until the stopping criterion is reached which could be a maximum number of iteration or the satisfaction of some predefined stoppage criteria. The basic trajectory pseudocode has this format as in Fig.2:

```
1. Generate initial solution (s(0))
2. t := 0
3. While not Termination  (s(t)), do
4. Explore neighborhood s'(t) := SelectMove(s(t));
5. If  Move(s'(t)) accepted,  then
6. s(t) := ApplyMove(s'(t))
7. End if
8. t := t+1
9. End While
```

Figure 2. Trajectory-based algorithms pseudocode

To help improve the efficiency in terms of time and quality of solutions of the Trajectory-based methods, experts introduced parallelism. The three popular parallel models in literature are the parallel multi-start, the parallel evaluation and parallel moves models (Alba, Talbi, Luque, & Melab, 2005)

• Parallel multi-start model: The parallel multi-start models, simply involve launching a number of trajectory-based searches with the aim of achieving robust solutions. These multi-start searches could work cooperatively or independently. Also, they could start from a particular point or from different points. They could use a similar or totally different parameters for their search (Mezmaz, Melab, & Talbi, 2006). A good example is the Kriging optimization method (Peri & Tinti, 2012)

• Parallel moves model: This uses a master-slave architecture that complies with the particular heuristic technique conducting the search. At the initial stage of the search, the master distributes the nodes among the slaves and at the expiration of the search; the slaves return their results to the master. This model basically speeds up the search as it is an imitation of population-based search, but this time under the harmonization of a master (Halambi et al., 2008). An example is the Panmictic Simulated Annealing (PSA) (Domínguez & Alba, 2011).

• Parallel Move acceleration model: This model evaluates each parallel move in a centralized place before subsequent moves are allowed. Some versions of this model permit such evaluation to be parallelized. Generally, parallelism has the disadvantage of low speed owing to much use of computer resources (Alba et al., 2005). Ethane Algorithm employs this technique. A major disadvantage of employing parallelism is the issue of huge computational cost that arises with such usage. Moreover, the application of parallelism is not a guarantee that the optimal solution will be achieved. (Domínguez & Alba, 2011)
It is pertinent to note that a number of trajectory-based search algorithms are available in literature. Some of the popular ones are the Simulated Annealing, Hill Climbing, Great Deluge algorithms.

Hill Climbing Algorithm

Hill Climbing (HC) searches for solution to problems in an iterative manner by first selecting an arbitrary solution and then tries to obtain better results/solutions by altering a single solution element (Hoffmann, 2010). If by this alteration, the new solution is better than then the previous, another alteration is made to another element of the newly-found solution until no improvement is possible on the solution. If, however, any alteration results in a poorer solution, then another alteration is made in the HC's efforts to obtain optimal solution. The HC is different from similar algorithms like the Gradient Descent in that the Gradient Descent adjusts all the values of the present solution but the HC adjusts just one value/element of the present solution in the next iteration (Perkins, Lacker, & Theiler, 2003). The Hill Climbing pseudocode is presented in Figure 3:

```
1.  PresentNode = startNode;
2.  While (not termination) do
3.          For j = Neighbors of PresentNode)
4.         NextEval = -INF
5.         NextNode = NULL
6.              For all k in j
7.              If (EVAL(x) > nextEval)
8.             NextNode = k
9.             End if
10.           End for
11.      End for
12.     NextEval = EVAL (k)
13.     If nextEval <= EVAL (PresentNode)
14.    End if
15. End While
16. //Return present node if no better neighbors exist
17. Return PresentNode
18. PresentNode = nextNode;
```

Figure 3. Hill climbing pseudocode

In practical terms, HC is a kind of Depth-First search. However, unlike the Depth-First search method which outrightly rejects or accepts a solution, the HC uses a feedback mechanism which approximates the closeness or otherwise of the newly-found solution to determine the next direction of the search. Hill Climbing has been successfully applied to configure application servers (Xi, Liu, Raghavachari, Xia, & Zhang, 2004), Travelling Salesman's Problem (Selman & Gomes, 2006), signature verification (Galbally, Fierrez, & Ortega-Garcia, 2007) etc.

The HC has advantage over some other algorithms in that it uses very little computer resources in its search since it stores only the present solution. Moreover, HC is capable of returning fairly better results than other algorithms in a situation of unexpected interruption to the algorithms execution. Nevertheless, in addition to the problem of low speed in instances involving ridges, alleys or plateau, HC is prone to falling into local minima in non-convex functions.

Simulated Annealing

Simulated Annealing (SA) which models the heating and cooling processes of materials in metallurgical engineering was developed by Kirkpatrick, Gelatt and Vecchi (Kirkpatrick et al., 1983). The temperature is lowered gradually to ensure that the cooling process minimizes the system energy thereby making the metals very strong. In the cooling process, the algorithm starts with a random search at high temperature which becomes greedy descent until the temperature reaches zero. Randomness attribute helps Simulated Annealing escape local optima since greedy descent is prone to falling into local minima. Better results are usually obtained in lower temperatures than in higher ones.

In its search for solutions, SA assigns random values to random variables, at each move, then, it evaluates the acceptance probability to ascertain whether it is an improvement on the objective function (and that it does not increase the conflict); which is for a minimization problem, a lower objective value. However, sometimes the SA, with certain probability, accepts points that raise the objective function and in this way avoids being trapped in local minima, thus, ensuring global exploration. The acceptance probability *ap* is calculated using:

$$ap = e^{\left[-\frac{\Delta E}{kBT}\right]} \tag{1}$$

Here, ΔE represents the change energy, kB is the Boltzmann's constant, and T is the Temperature. Similarly, the change in the objective function $\Delta \mathrm{f}$ is directly related to the change in the Energy $\Delta \mathrm{E}$:

$$\Delta \mathrm{E} = \gamma \Delta \mathrm{f} \tag{2}$$

Here γ is a positive real constant. For effectiveness, an Annealing schedule which is sometimes linear and at other times geometric is chosen to systematically reduce the temperature as the algorithm progresses. The decrease of the temperature enables the SA to minimize the search scope and ensure early convergence.
SA has been applied to several problems, such as the Quadratic Assignment Problem, Job Shop Scheduling, Travelling Salesman's Problems, N-Queens problem, Artificial Neural Networks Training etc. (Ledesma, Aviña, & Sanchez, 2008). The pseudocode pf the SA is presented in Figure 4.

```
1. Let $x = x_0$
2. For $y = 0$ to $y_{max}$ :
3.     $T \leftarrow temperature(ky/ y_{max})$
4.     Select a neighbor randomly, $x_{new} \leftarrow neighbor(x)$
5.     If $P(E(s), E(x_{new}), T) \geq random(0, 1)$, move to the new state
6.    End if
7. End for
8. $x \leftarrow x_{new}$
9. Output: the final state x
```

Figure 4. SA pseudocode

One of the biggest strengths of SA is her ability to escape being trapped in local minima with her tacit manipulation of the cooling temperature. However, SA is not very effective in a smooth energy landscape or in instances with few local minima. Moreover, SA has the problem of delay in arriving at quality solutions (Kumbharana & Pandey, 2013). This may be due to engaging in several evaluations of the cost function in each iteration.

The Great Deluge

The Great Deluge (GD) which was developed by G. Dueck is inspired by the activities of a person moving in different directions upwards a hill in times of a deluge in an attempt to avoid his/her feet being wet as the water level rises (Dueck, 1993). GD begins by assigning an initial value which is the same as the initial objective function to the parameter 'Level' and as the search progresses, this value is iteratively reduced. The algorithm accepts solutions that has lower (or at least, equivalent) values of the objective function than the present 'Level'(Mcmullan, 2007). A later version of GD accepts all downhill moves, thus, hybridizing GD with Hill Climbing to ensure greater efficiency. In implementing the GD, usually, an approximate *solution J* of the optimum solution is selected. Next a random value of *badness K* is chosen and used to evaluate the desirability or otherwise of the approximate solution.

The GD pseudocode is presented in Figure 5.

```
1.  SolutGD ←Sol
2.  SolutOptimalGD ←Sol
3.  f (SolutGD) ←f(Sol)
4.  f (SolutOptimalGD)←f(Sol)
5.  Determine best rate of final solution, BestRate
6.  Determine number of iterations, NumOfIterGD
7.  Determine initial level: level ←f (SolutGD)
8.  Determine decreasing rate ΔB = ((f (SolutGD)–BestRate)/ (NumOfIterGD)
9.  Determine iteration ←0
10. Determine not_improving_counter ←0, not_improving_ length_GDA
11.       Do while (iteration < NumOfIterGD)
12.      Apply neighborhood structure Ni where i ∈{1... K} on SolutGD, TempSolutGDi
13.      Evaluate cost function f (TempSolGDi)
14.      Seek the best solution among TempSolutGDi where i ∈{1... K} call new solution SolutGD*
15.      If (f (SolutGD*) < f (SolOptimalGD)) SolutGD ←SolutGD*
16.          SolutOptimalGD ←SolutGD*
17.      End if
18.      If not_improving_counter ←0
19.          Level = level - B
20.          Else if (f (SolutGD*) ≤Level) SolutGD ←SolutGD*
21.      End if
22.      If not_improving_counter ←0
23.      Else not_improving_counter++
24.     End if
25.     If (not_improving_counter == not_improving_length_GDA)
26.     Level= level + random (0, 3)
27.    Increase iteration by 1
28. End if
29. End do
30. Return SolutOptimalGD
```

Figure 5. Great deluge pseudocode

Please note that the higher the value of *badness* evaluation, the more undesirable solution *J* is. Another parameter *tolerance* is used to evaluate a number of factors leading to the choice of a new approximate solution *J'* that is a neighbor of *J*. The *badness* of *J'* is then calculated and the result is compared with parameter *tolerance*. If the output is better than *tolerance*, the GD is restarted recursively:

$$J := J' \tag{3}$$

However, if the output of *J'* is worse than *tolerance*, another neighbor *J''* is selected for *J* and the process is then repeated until all the neighbors of *J* produce better results than *tolerance*, then the GD is terminated and *J* is then output as the solution.

So far, the application areas of the GD include examination, sports and course timetabling, patient admission problems (Kifah & Abdullah, 2015), protein structure prediction (Burke et al., 2007), facility layout problems (Nahas, Kadi, & El Fath, 2010) etc.

A close look at GD reveals that it is different from Hill-Climbing and Simulated Annealing because it accepts a candidate solution from the neighborhood. The GD permits the definition of the characteristics, for instance, processing time and processing region of the expected solution, of a search process in advance. These help to make GD much more efficient than some other algorithms such as Hill Climbing and Simulated Annealing (Burke, Bykov, Newall, & Petrovic, 2003) . Nonetheless, GD has the disadvantage of falling into a local minima and this has led to the development of some variants of the algorithm (Mcmullan, 2007).

Population-based Metaheuristics

Generally, population-based methods utilize a set of decision vectors which is usually denoted by:

$$X = \{x_{1,}\ x_{2,}\ x_{3,}\ \dots x_N\ \} \tag{4}$$

where

$$Xi = (x_i1, x_i2, \ldots, x_in) \tag{5}$$

N represents the size of the population and n, the number of design variables in each decision vector (i.e. individuals in a population). Basically, population-based metaheuristics share a common structure made up of four main parts, namely, the main algorithm; an extension to deal with the constrained optimization problems; another extension to retain promising solutions and a part to halt the algorithm (stopping criteria). In most cases, the main algorithm has component tracks information among the population (crossover), another component that pushes the population forward for further exploration (sometimes called mutation in some methods) and the part that computes the best solution in each iteration (selection) (Chau, Kwong, Diu, & Fahrner, 1997). Based on the foregoing discussion, most population-based metaheuristics have a format similar to Figure 6:

```
1.
2. Initialize population
3. Evaluate the objective function
4. Repeat
5. Evaluate the population quality
6. Apply the variation operator
7. Evaluate the objective function
8. Until termination condition
```

Figure 6. Population-based algorithms pseudocode

Most population-based techniques exploit some previous knowledge of the solution space and use this in the initialization phase to move the search agents towards the feasible region. In situations where this information is absent, the decision vectors are distributed uniformly within the search space. Examples of the population-based algorithms are the Genetic Algorithm, Firefly Algorithm, the African Buffalo Optimization algorithm etc.

Particle Swarm Optimization

Particle Swarm Optimization (PSO), which was developed by Kennedy and Eberhart is a simulation of the flocking of birds and schooling of fishes in search of food. PSO is a population-based, stochastic, global optimization technique specifically designed to be an easy-to-implement yet effective search optimization method (Kefi, Rokbani, Krömer, & Alimi, 2015). Since its development in 1995, PSO has been a very successful algorithm which has been applied to a wide range of application areas. The PSO models the velocity and positions of particles in their search for food. The position of each particles in PSO represents a solution to the problem. The algorithm makes use of five major parameters: the particles' velocity, their present position, the global best particles' position, the individual particle's knowledge of its previous best position achieved and the best position found by its neighbor. As the algorithm progresses, the particles update their position and velocity with each iteration until it reaches termination condition. Similarly, the PSO maintains an information repository that keeps track of the best achieved objective function values for each particle involved in the search process (Kefi et al., 2015)

PSO models the behavior of, for example, a swarm of birds searching for a food source. The entire particles converge on the best solution through the use of the information gathered by each particle, the neighboring particle as well as that obtained from the entire flock. The algorithm starts by initializing the particles in the search space, followed by updating the position and velocity of each particle in each iteration. Basically, the action steps in PSO algorithm are as follows:

```
1.  Initialize particle
2.  Do  (Until termination)
3.      For individual particle
4.      Calculate fitness value
5.      If  fitness value is better than its overall best fitness value (pBest) since inception
6.          Record the current value as its new pBest
7.      End if
8.      Select the particle that has the best fitness value in the swarm as the gBest
9.      End for
10.     For individual particle
11.         Determine the individual particle's velocity using velocity equation
12.         Update the individual particle's position using position equation until termination condition
13.     End for
14. End do
15. Output the best result
```

Figure 7. PSO pseudocode

So far, PSO has been used successfully to address several problems ranging from complex nonlinear function optimization, communications and control applications, task assignment (Shi & Eberhart, 1999), antenna design, combinatorial optimization problems, biomedical and pharmaceutical applications, fault diagnosis etc. (Poli, 2007).

A critical assessment of the PSO algorithm, however, reveals that there are about 30 different variants of PSO presently (Langeveld & Engelbrecht, 2011). These great number of variants were developed in response to the observed weaknesses of the algorithm. Of this number, the classical PSO algorithm is relatively simple to implement and uses relatively fewer parameters than in some algorithms like the ACO (Pereira, 2011).

Furthermore, PSO is rather more efficient in its use of memory than the Evolutionary algorithms like the GA. In addition, PSO has the capacity for broad diversity in its search space since all the particles use the information obtained by the best particles either in the neighborhood or the overall best in each iteration to improve their positions and speed. This search mechanism of the PSO is different from that of GA, for example, because in the GA, worse solutions are discarded and the population only concentrates on the fittest individuals. It should be emphasized, nonetheless, that the PSO shares some similarities with Evolutionary algorithms like the Genetic Programming (GP), Evolutionary Strategies (ES) Evolutionary Programming (EP) and Genetic Algorithm (GA) through the initialization of solutions and the update of generations. Even though the PSO does not use evolution operators like crossover, mutation, recombination, inversion and selection, the particles in PSO are simply attracted by the best in the neighborhood or the global best as the case may be.

In all, PSO can be said to be an efficient and effective algorithm in searching continuous functions and in multimodal search environments. It has been successfully applied to solve the Travelling Salesman's Problems, PID tuning of Automatic Voltage Regulators, global optimization problems, computer networking problems, robotic applications, scheduling, signal processing (Odili & Kahar, 2016) etc. However, PSO uses several parameters such as the constriction factor, inertia weight, random numbers, social factors, neighborhood best, personal best, global best etc. and this affects the algorithm's speed and management of computer resources (Tanweer, Suresh, & Sundararajan, 2015).

Ant Colony Optimization

The Ant Colony Optimization (ACO) is one of the most popular metaheuristic algorithms in literature. The ACO algorithm designed by Marco Dorigo and Di Caro in 1999 as a modification of the Ant System by Marco Dorigo and Ant Colony System by Dorigo and Gambardella (Anwar, Salama, & Abdelbar, 2015). The ACO is a simulation of the random movements of ants in search of food. Once the ants discover a food source is, they pick particles of the food and on their way back to their nest, usually following a shorter route, deposits some amount of pheromones as a way of informing other ants of their discovery. As the neighboring ants receive the information by perceiving the scent of the pheromone, they will likely join the successful ant in the harvest of the food source. As these ants locate the food source, they, in turn, carry some fragments of the food and on their way back to the nest, drop pheromones as they further optimize the route of the initial ant. This process increases the pheromone concentration on the favorite 'shortest' route(s) and in that process attract other ants.

This harvest cycle continues and within a short while, the entire colony of ants are on the optimized route harvesting the food source to the safety of their nests. With time, changes are that the food source is either partially or totally exhausted, the pheromone concentration diminishes as the discouraged ants will no more drop pheromones. This serves as negative reinforcement as the route loses its attraction due to pheromone evaporation. This situation leads the ants to explore other areas (Wu, Xin, & Zhang, 2015).
The ACO basically is the modelling and simulation of ants' foraging behavior, brood sorting, nest building and self-assembling. The Algorithm has three main subroutines: ConstructAntSolutions, Pheromone Update and DeamonActions.

(i) ***ConstructAntSolutions:*** This subroutine constructs the solution by stimulating the movement of artificial ants through adjacent states of a problem according to a transition rule and, thus, builds a solution iteratively.
(ii) ***Pheromone Update:*** This subroutine performs pheromone update trails either through pheromone reinforcement or evaporation. The is achieved in two ways depending on the variant of the ACO one is working with: the pheromone update subroutine could be done at the end of each iteration or when the ants individually completes a solution.
(iii) ***DeamonActions:*** This subroutine is optional depending on the problem being solved. It involves increasing the pheromone levels to select promising edges.(Kumbharana & Pandey, 2013)

In general, the ACO algorithm uses the following steps (Liu, Zhu, Ma, Zhang, & Xu, 2015):

```
1. Initialize pheromone values τ for all the edges in the graph.
2. For a start, all the edges should have equal amount of pheromone unless there exist some
   heuristic information favoring some edges that may lead to speedier convergence.
3.     Construct a solution for each ant, x= (1,2,...,N)
4.     Update the pheromone values for each edge depending on the quality of solution.
5.  End for
6. Go to Step 2 until stopping criterion is reached.
7. Output the best result
```

Figure 8. ACO pseudocode

So far, the ACO has proven to be effective in solving routing problems in computer and telecommunication networks, Sequential Ordering Problems, Travelling Salesman's Problems (Odili, 2013), Resource Constraint Project Scheduling, Subset problems, Machine Learning Problems, Vehicle routing, Proportional, Integral and Derivative parameters-tuning of Automatic Voltage Regulators, Stochastic optimization problems (Stützle, López-Ibáñez, & Dorigo, 2011) etc.

One of the strengths of the ACO is its flexibility in being hybridized with other heuristics or metaheuristics in a quest for greater efficiency. Moreover, it is robust in search situations where the graph is prone to dynamic changes. Also ACO performs well in distributed computing environments. However, it has the weakness of easily falling into premature convergence because its pheromone update is according to the present best path. Again, it uses several parameters that require proper tuning. Such parameters as pheromone quantity, pheromone update rule, evaporation rate, pheromone reinforcement rate etc and this affects the speed of the algorithm. (Gutjahr, 2003; Kumbharana & Pandey, 2013) .

Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) which was inspired by the behavior of natural honey bee swarm was developed by Karaboga and Basturk in 2009 (Karaboga & Akay, 2009). The ABC categorizes bees into three, namely, scout bees, onlooker bees and employed bees. Scout bees are those that fly around the search place seeking solutions (food source). Similarly, onlooker bees wait in the nest expecting the report of the scout bees. On the other hand, employed bees refer to those bees that joins in the food source exploitation after watching the waggle dance of the scout bees. This classification is dynamic in that a scout bee could transform into an employed bee once it (the same scout bee) is involved in harvesting the food source and an onlooker bee at another stage. In this algorithm, the food source represents a solution to the optimization problem. The volume of nectar in a food source represents the quality (fitness) of the solution (Nozohour-leilabady & Fazelabdolabadi, 2015).Whenever the employed bees bring some nectar to the hive, they have three alternatives: return to get more nectar, accompany other dancing bee to a new site or simply stay back in the hive. The bees' decision is informed by a number of factors ranging from the quality and quantity of the available nectar at the food source, the distance of the food source to the hive and the number of employed bees harvesting the nectar at that food source. The global search mechanism of the ABC depends of the random search capacity of the scout bees while the exploitation capacity of the algorithm is based on the activities of the employed and onlooker bees. This bee behavior can be replicated in some real life problems, such as in transportation and business applications (Pham et al., 2011).

In general, the pseudo code of the classical ABC (Nozohour-leilabady & Fazelabdolabadi, 2015) is:

```
1. (Initialize population with random solutions.
2. Evaluate fitness of the individual bees in the population.
3. While (stopping criterion is not reached), form new population.
4.     Choose site(s) for the next neighborhood search.
5.     Select bees for chosen locations to be searched ensuring that more bees are allocated to the best
       locations & evaluate fitness.
6.     Choose the fittest bee from each patch for the next best harvesting site(s).
7.     Assign remaining bees to search randomly and evaluate their fitness.
8. End While.
```

Figure 9. ABC pseudo-code

The ABC has been successfully applied to solving a number of problems such as Travelling Salesman's Problems, accident diagnosis, reactive power optimization problems, radial distribution (Akay & Karaboga, 2015), training neural networks, wireless sensor networks, signal, image and video processing (Karaboga, Gorkemli, Ozturk, & Karaboga, 2014) etc.

Previous studies on the ABC has shown that it is very effective in feed-forward artificial neural networks training, in addition to the algorithm being very efficient in multidimensional search environments due to its capacity to get out of a local minimum with ease. However, studies on the Artificial Bee Colony are still not widespread (Karaboga, Akay, & Ozturk, 2007; Karaboga & Basturk, 2007; Karaboga & Ozturk, 2009) and there are several parameters to appropriately tune in order to get good results. Generally, the ABC is a slow algorithm.

African Buffalo Optimization

The African Buffalo Optimization algorithm was designed by Odili and Kahar in 2015 (Odili & Kahar, 2015a). It drew its inspiration from the random movement of the African buffalos deploying principally two distinct sounds (/waaa/ and /maaa/) to move around the diverse African landscape in search of grazing pastures (Odili & Kahar, 2015b). This algorithm harnesses the three main characteristics of the African buffalos that are responsible for their successful migration. These are their extensive memory represented by m_k' and w_k' in the algorithm; extensive communication represented by $\mathrm{bg} - \mathrm{w_k}$) and their personal intelligence, represented by $\mathrm{bp_k} - \mathrm{w_k}$. The ABO algorithm is presented in Figure 10

```
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) λ
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 10. ABO algorithm

The algorithm has not enjoyed wide application since it was first published. Some of the application areas include Mobile Ad-hoc Network (Hassan & Muniyandi, 2017), asymmetric travelling salesman's problem (Odili & Mohmad Kahar, 2016b), symmetric travelling salesman's problem (Odili, Mohmad Kahar, & Noraziah, 2016), strategic management, collision avoidance in electric fish, tuning of PID parameters of Automatic Voltage Regulators (Odili & Mohmad Kahar, 2016a) etc. Because of the limited applications so far, it is not very easy to evaluate the algorithm. In any case, the ABO has performed well in the areas of its application.

Firefly Algorithm

This population-based algorithm which was inspired by the flashing attitude of fireflies was developed by Xin-She Yang (Yeomans & Yang, 2014). In this algorithm, a number of fireflies work together to solve a problem through bioluminescent glowing that enables them to efficiently solve problems. The solutions to problems are modelled as a firefly whose flashes are proportional to the quality of solutions they represent. As a result, a brighter firefly attracts others colleagues and this aids further exploration of the search space. The four main characteristics of the algorithm include:

(a) All fireflies are unisex, so they are attracted by a brighter colleague notwithstanding the sex
(b) The brightness of a firefly is a function of the distance between the fireflies: the nearer they are to one another, the more the effect of the brightness of the brighter firefly and vice-versa.
(c) If there is no brighter firefly than a particular firefly, the firefly without a brighter colleague, moves randomly
(d) The landscape of the objective function affects the brightness of the firefly. The brightness of the firefly is proportional to the objective function in a maximization problem. The pseudo-code of the Firefly Algorithm is presented in Fig. 11:

```
1.  Randomly place the  bee to an empty solution
2.  For each bee, execute the forward pass phase
3.      Set k=1; k is a counter for constructive moves in the forward pass;
4.      Determine all possible constructive moves;
5.      End for
6.  Select the next move using the roulette wheel;
7.      k= k + 1; If $k \leq NC$ (NC= maximum number of counters),
8.      Let the bees start the backward pass phase;
9.      Sort the bees;
10.     Let each bee decide, randomly, whether to continue its own exploration and
        become a recruiter, or to become a follower of bees with higher objective function value;
11. End roulette wheel
12. For every follower bee, choose (until termination)a new solution from recruiter bees using the
    roulette wheel
13. Repeat steps 7-11
14. Output the best result.
```

Figure 10.FFA pseudo-code

FFA has been successfully applied to Industrial Optimization, Image processing , Antenna design ,Business optimization, Civil engineering, Robotics, Semantic web, Chemistry, Meteorology, Wireless sensor networks (Fister, Yang, & Brest, 2013) etc.

The FFA is believed to have least error percentage compared to many other metaheuristic algorithms such as the GA and the PSO. Moreover, it is relatively simple to implement and has proven to perform well in multi-modal search environments. It is quite close to the PSO except that it does not employ search velocities in its quest for solutions (X.-S. Yang, 2012). However, it has complicated fitness function and hugely depends on correct parameter setting to arrive at good solutions. Even though it obtains good results, the FFA is a slow algorithm. This may be due to the number of parameters that the algorithm uses in search of solutions.

Biogeography- Based Optimization

The BBO which was designed by D. Simon was inspired by the immigration and emigration between habitats by different species in nature (Simon, 2008). Using the values of the Habitat Suitability Index as the objective function, the BBO emphasizes communication between candidate solutions (habitats): the Low Habitat Suitability Index (LHSI) and the High Habitat Suitability Index (HHSI) which are features of each solution. Basically Habitat Suitability Index (HSI) of a solution is a measure of its fitness and this is dependent on the characteristic environmental features in that habitat. This algorithm obtains good solutions through regular interaction/communication and feature-sharing between the LHSI and the HHSI. Another very important parameter to the working of the BBO is the *mutation* parameter which, like the Genetic Algorithm, enables the algorithm to generate diversity and thus greater exploration (Simon, 2008). The BBO algorithm is presented in Figure 12.

```
1.  Initialization: Initialize the population candidate solutions
2.  While (not termination), do
3.        For each candidate solution, create emigration probability fitness of each
4.        candidate solution, with probability of (0, 1)
5.              For each candidate solution, create immigration probability
6.                  For each individual specie, set independent variable index
7.                  Using the set probabilistically criterion, determine whether to immigrate;
8.                  If yes, then
9.                       Using the created probabilistically choose the emigrating individual
10.                 End if;
11.                End for
12.           End for
13.      End for
13.    End while.
14.  Consider the next independent variable index;
15.  Probabilistically mutate to create another individual;
16.  Consider the next individual;
17.  Next generation
18.  Repeat steps 2 and 3 until stopping criteria
19. Output best solution
```

Figure 12. BBO pseudocode

The BBO has been successfully applied to solve numerical problems, population distribution problems, Combined Heat and Power Economic Dispatch Problem etc. (Qu & Mo, 2011; Simon, Ergezer, & Du, 2009)
The BBO is an exploration-based algorithm and even though it adopts the mutation parameter of GA, it does not discard the previous population after a new Generation; rather, the BBO modifies the original population through migration. Similarly, the BBO depends on the evaluation of newly generated solution fitness to determine whether the species' rate of emigration or migration. However, BBO has been discovered to be rather weak in the migration and mutation stages, as such its overall performance is suspect (Alroomi, Albasri, & Talaq, 2013). BBO's ineffectiveness may possibly due to the use of several parameters that require proper tuning in order to get good results. Finally, the speed of the algorithm is suspect as a result of its use of several parameters.

Bee Colony Optimization

In this algorithm, a population of artificial bees conducts a search for the optimal solution(s). Each bee generates one solution to the problem (Karaboga & Basturk, 2007). The Bee Colony Optimization algorithm makes use of two alternating phases, namely, forward-pass phase and backward-pass phase. In the forward-pass phase, the bees explore the search space through a number of moves that construct or improve a solution and thereby creating a new solution. After arriving at this partial solution, the bees return to the nest to initiate the backward-pass phase where they communicate information about their newly-found solutions. They communicate the distance as well as the quality of the food source (solution) to the nest through a waggle dance. Based on the information collected at this phase, the bees (including the dancer) decide, with a certain probability, to abandon the newly-found solution and remain in the nest or to follow the dancing bee to the food source. This leads to the second forward-pass where the bees improve on the previously found solutions and after which they return to another backward-pass phase. These phases are performed repeatedly until a stopping criterion is reached.

The stopping criterion could be the maximum number of forward-/backward-pass phases, the maximum number of forward-/backward-phases without tangible improvement of the objective function or the arrival at the optimal solution. Basically, the pseudo-code of the Bee Colony Optimization algorithm is as in Fig. 13.

```
1.  Randomly initialize the bees within the search space
2.  For each bee, perform the forward-pass:
3.      Set k=1; k is a counter for constructive moves in the forward pass;
4.      Determine all possible constructive moves;
5.      Select one move using the roulette wheel;
6.      k= k + 1; If k≤NC (NC= maximum number of counters),
7.      Return to step 2
8.  End for
9.  Let the bees start the backward-pass phase;
10. Based on the objective function value of each bee, sort the bees;
11. Let each bee decide, randomly, whether to continue its own exploration and become a recruiter,
    or to become a follower of bees with higher objective function value;
12. For every follower bee,
13.      Choose a new solution from recruiter bees using a roulette wheel;
14.       If the stopping criterion is not reached,
15.            Return to step 2
16.       End if
17. End for
18. Output the best result.
```

Figure 13. BCO pseudo-code

Bee colony Optimization has been successfully applied to solving the Job Shop Problem, MANET- Routing Protocol, Generalized Assignment Problem, Engineering Optimization, Travelling Salesman's Problem, Numerical Assignment Problems etc. (Nagpure & Raja)

The mechanism that informs the decision of a bee to follow a particular dancer is not well established but it is rather vaguely considered that the decision to follow a particular bee is a function of the food source (Camazine & Sneyd, 1991). However, the abandonment phenomenon is of great benefit to the algorithm. That is to say that when the employed bees could not find the optimized solutions after some repetitions, they transform to the scout bees again and move in random paths to start searching for optimized solutions. This way, the solutions which are not optimized are abandoned and further searches are made for the global optimized points. This procedure helps the algorithm not to fall into a local optima or minima in a multi-dimensional search environment. As a result, it could be safe to say that the artificial bees use a combination of local and global searching methods to arrive at solutions.

It should, however, be observed that that the BCO has complicated fitness function. As such, obtaining good results is a function of proper setting of parameters such as minimum overshoot, rise time, steady state error and settling time in the state response. Moreover, it has been observed that Bee Colony algorithm is not as adaptive as ACO. Finally, BCO algorithm shows poor performance and remains inefficient in exploring search space because its search equation is significantly influenced by a random quantity which helps in exploration at the cost of exploitation of the search space (Babaeizadeh & Ahmad, 2014)

Findings

The development of several optimization algorithms, some of which have been described in this study, no doubt, has been of immense benefits to the scientific community in general and computer scientists, in particular (Di Caro, Ducatelle, Gambardella, & Dorigo, 2005; Giannakouris, Vassiliadis, & Dounias, 2010; Ridge a't, Kudcnko, & Kazakov'i, 2005; Wedde & Farooq, 2005; X.-S. Yang, 2009a). These algorithms have sped up industrial processes, minimized waste of time, money and computer resources. However, a critical look at these NAs indicates that there are no clear theoretical proofs of their workings. It is rather worrisome that in cases where there are results of convergence analysis as in the GA (Rudolph, 1994), these results are rather limited. Moreover, there are no clear mathematical proofs in most of these algorithms (Farmer, Guttman, & Thayer, 1993). As a result, there is need for deliberate research on the mathematical, theoretical and convergence evaluation of NAs. The research community will benefit maximally from NAs when the convergence conditions of NAs are clearer to the extent that the non- experts can easily understand.

Moreover, this study observes that that most application of NAs were concerned with relatively simple to medium-scale experimental cases involving scores of variables. It is not well established if these NAs can do as well in practical real-life situations that may involve hundreds or even thousands of variables. This should also be an area of further research investigation.

Furthermore, the application areas of the NAs are, to say the least, largely theoretical. Except in few situations like the practical Proportional, Integral and Derivative parameters-tuning of AVR and DC motors (Al Gizi, Mustafa, Al Zaidi, & Al-Zaidi, 2015; Meshram & Kanojiya, 2012) etc, most of the studies available in literature are more or less theoretical. The danger with such simulated studies is that it may not work appropriately in a real-life situation.

Finally, it is observed that most of the NAs still use manual setting of parameters. It will be better if the NAs employ dynamic setting of parameters in course of the algorithm execution. There have been a few studies employing dynamic parameter setting (Lobo, Lima, & Michalewicz, 2007; Schraudolph & Belew, 1992; B. YANG & ZHANG, 2004; Zielinski & Laur, 2007) but, we dare say, these studies are quite few and are basically at their infancy. The scientific community stands to benefit a lot more from NAs when the NAs employ dynamic parameter tuning.

Conclusion

This paper did a critical review of ten Nature-inspired optimization (NAs) techniques, picking three trajectory-based algorithms and seven population-based. The trajectory-based algorithms examined are the Simulated Annealing, the Great Deluge and the Hill Climbing. Similarly, the six population-based algorithms investigated in this study are the PSO, ACO, ABC, BCO, BBO ABO and the FFA. At the end of the critical analysis of these ten algorithms, it was observed that the NAs have made immense contributions to different aspects of human development: scientific, engineering, medical, business etc. Nevertheless, the study revealed that there is a general lack of clear mathematical and theoretical proof of convergence in most NAs. Moreover, the algorithms pay so much attention to problems with relatively less number of variables than that are common in practical day-to-day real-life industrial problems that has several hundreds, thousands and even millions of variables. There is, therefore, little or no guarantee that the same methodology used to obtain results in the small problems will produce same results in larger real-life environments. Furthermore, the studies are largely simulated with the possibility of not being as effective in real-life situations. It is, therefore, recommended that future research studies in NAs should place greater emphasis on experimentation with real-life situations, unravelling the mathematical and theoretical convergence analysis of the NAs, in addition to, focusing on problems with very large number of variables.

Finally, in terms of the algorithm design, it was observed that many of the NAs make use of several parameters leads to extensive use of computer resources and slow speed. The faster algorithms, according to this review, are usually those that deploy fewer parameters. This is in consonance with the recent drive for lean metaheuristic design deliberately avoiding the Frankenstein phenomena. Frankenstein phenomena refers to a situation where an optimization algorithm deploys several parameters in its search for solution to the extent that the individual contribution of each parameter is difficult to pinpoint (Sörensen, 2015; Sörensen & Glover, 2013). It is, therefore, recommended that algorithm designers should deploy only the number of parameters needed to obtain good solutions. This will enhance algorithm efficiency without compromising effectiveness.

Acknowledgment

The author appreciates the support of the Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Kuantan 26300, Malaysia, for providing environment conducive for this research as well as the funding for this research under Grant 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 there exists no conflict of interest in the publication of this article.

References

Aghamohammadi, M., & Pourgholi, M. (2008). Experience with SSFR test for synchronous generator model identification using Hook-Jeeves optimization method. *International Journal of System Applications, Engineering and Development, 2*(3), 122-127.

Akay, B., & Karaboga, D. (2015). A survey on the applications of artificial bee colony in signal, image, and video processing. *Signal, Image and Video Processing, 9*(4), 967-990.

Al Gizi, A. J., Mustafa, M., Al Zaidi, K. M., & Al-Zaidi, M. K. (2015). Integrated PLC-fuzzy PID Simulink implemented AVR system. *International Journal of Electrical Power & Energy Systems, 69*, 313-326.

Alba, E., Talbi, E., Luque, G., & Melab, N. (2005). 4. Metaheuristics and Parallelism. *Parallel Metaheuristics: A New Class of Algorithms. Wiley*, 79-104.

Alroomi, A. R., Albasri, F. A., & Talaq, J. H. (2013). Solving the Associated Weakness of Biogeography-Based Optimization Algorithm. *International Journal on Soft Computing, 4*(4), 1-20.

Anwar, I. M., Salama, K. M., & Abdelbar, A. M. (2015). Instance Selection with Ant Colony Optimization. *Procedia Computer Science, 53*, 248-256.

Babaeizadeh, S., & Ahmad, R. (2014). A Modified Artificial Bee Colony Algorithm for Constrained Optimization Problems. *Journal of Convergence Information Technology, 9*(6), 151.

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming: an introduction* (Vol. 1): Morgan Kaufmann San Francisco.

Binitha, S., & Sathya, S. S. (2012). A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering, 2*(2), 137-151.

Brabazon, A. (2008). *Natural computing in computational finance* (Vol. 1): Springer Science & Business Media.

Burke, E., Bykov, Y., & Hirst, J. (2007). Great Deluge Algorithm for Protein Structure Prediction.

Burke, E., Bykov, Y., Newall, J., & Petrovic, S. (2003). A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research ISSN: 0354-0243 EISSN: 2334-6043, 13*(2).

Camazine, S., & Sneyd, J. (1991). A model of collective nectar source selection by honey bees: self-organization through simple rules. *Journal of theoretical Biology, 149*(4), 547-571.

Chau, C., Kwong, S., Diu, C., & Fahrner, W. (1997). *Optimization of HMM by a genetic algorithm.* Paper presented at the Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on.

Crawford, C., & Krebs, D. L. (2013). *Handbook of evolutionary psychology: Ideas, issues, and applications*: Psychology Press.

Cuevas, E., & Sossa, H. (2013). A comparison of nature inspired algorithms for multi-threshold image segmentation. *Expert systems with applications, 40*(4), 1213-1219.

Di Caro, G., Ducatelle, F., Gambardella, L. M., & Dorigo, M. (2005). AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications, 16*(5), 443-455.

Domínguez, J., & Alba, E. (2011). Ethane: a heterogeneous parallel search algorithm for heterogeneous platforms. *arXiv preprint arXiv:1105.5900*.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE, 1*(4), 28-39.

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial life, 5*(2), 137-172.

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational physics, 104*(1), 86-92.

Farmer, W. M., Guttman, J. D., & Thayer, F. J. (1993). IMPS: An interactive mathematical proof system. *Journal of Automated Reasoning, 11*(2), 213-248.

Fister, I., Yang, X.-S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation, 13*, 34-46.

Forsythe, G. E., & Wasow, W. R. (1960). Finite-difference methods for partial differential equations.

Galbally, J., Fierrez, J., & Ortega-Garcia, J. (2007). Bayesian hill-climbing attack and its application to signature verification *Advances in Biometrics* (pp. 386-395): Springer.

Giannakouris, G., Vassiliadis, V., & Dounias, G. (2010). Experimental study on a hybrid nature-inspired algorithm for financial portfolio optimization *Artificial Intelligence: Theories, Models and Applications* (pp. 101-111): Springer.

Gutjahr, W. J. (2003). A converging ACO algorithm for stochastic combinatorial optimization *Stochastic algorithms: Foundations and applications* (pp. 10-25): Springer.

Halambi, A., Grun, P., Ganesh, V., Khare, A., Dutt, N., & Nicolau, A. (2008). *EXPRESSION: A language for architecture exploration through compiler/simulator retargetability.* Paper presented at the Design, Automation, and Test in Europe.

Hassan, M. H., & Muniyandi, R. C. (2017). An Improved Hybrid Technique for Energy and Delay Routing in Mobile Ad-Hoc Networks. *International Journal of Applied Engineering Research, 12*(1), 134-139.

Hoffmann, J. (2010). A heuristic for domain independent planning and its use in an enforced hill-climbing algorithm *Foundations of Intelligent Systems* (pp. 216-227): Springer.

Karaboga, D., & Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review, 31*(1-4), 61-85.

Karaboga, D., Akay, B., & Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks *Modeling decisions for artificial intelligence* (pp. 318-329): Springer.

Karaboga, D., & Aslan, S. (2015). *A new emigrant creation strategy for parallel Artificial Bee Colony algorithm.* Paper presented at the 2015 9th International Conference on Electrical and Electronics Engineering (ELECO).

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization, 39*(3), 459-471.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review, 42*(1), 21-57.

Karaboga, D., & Ozturk, C. (2009). Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Network World, 19*(3), 279.

Kefi, S., Rokbani, N., Krömer, P., & Alimi, A. M. (2015). *A New Ant Supervised-PSO Variant Applied to Traveling Salesman Problem.* Paper presented at the Hybrid Intelligent Systems: 15th International Conference HIS 2015 on Hybrid Intelligent Systems, Seoul, South Korea, November 16-18, 2015.

Kennedy, J. (2010). Particle swarm optimization *Encyclopedia of Machine Learning* (pp. 760-766): Springer.

Kifah, S., & Abdullah, S. (2015). An adaptive non-linear great deluge algorithm for the patient-admission problem. *Information Sciences, 295*, 573-585.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science, 220*(4598), 671-680.

Kumbharana, N., & Pandey, G. M. (2013). A Comparative Study of ACO, GA and SA for Solving Travelling Salesman Problem. *International Journal of Societal Applications of Computer Science, 2*(2), 224-228.

Kunna, M. A., Kadir, T. A. A., Jaber, A. S., & 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), 120.

Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder--Mead simplex method in low dimensions. *SIAM Journal on optimization, 9*(1), 112-147.

Langeveld, J., & Engelbrecht, A. P. (2011). *A generic set-based particle swarm optimization algorithm.* Paper presented at the International conference on swarm intelligence, ICSI.

Ledesma, S., Aviña, G., & Sanchez, R. (2008). Practical considerations for simulated annealing implementation. *Simulated Annealing, 20*, 401-420.

Liu, J., Zhu, H., Ma, Q., Zhang, L., & Xu, H. (2015). An Artificial Bee Colony algorithm with guide of global & local optima and asynchronous scaling factors for numerical optimization. *Applied Soft Computing, 37*, 608-618.

Lobo, F. G., Lima, C. F., & Michalewicz, Z. (2007). *Parameter setting in evolutionary algorithms* (Vol. 54): Springer Science & Business Media.

Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M. N., Salcedo-Sanz, S., & Geem, Z. W. (2013). A survey on applications of the harmony search algorithm. *Engineering Applications of Artificial Intelligence, 26*(8), 1818-1831.

Mcmullan, P. (2007). An extended implementation of the great deluge algorithm for course timetabling *Computational Science–ICCS 2007* (pp. 538-545): Springer.

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological informatics, 1*(4), 355-366.

Meshram, P., & Kanojiya, R. G. (2012). *Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor.* Paper presented at the Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on.

Mezmaz, M., Melab, N., & Talbi, E.-G. (2006). *Using the multi-start and island models for parallel multi-objective optimization on the computational grid.* Paper presented at the e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Nagpure, H., & Raja, R. The Applications Survey on Bee Colony Optimization.

Nahas, N., Kadi, D. A., & El Fath, M. N. (2010). *Iterated great deluge for the dynamic facility layout problem*: CIRRELT.

Nozohour-leilabady, B., & Fazelabdolabadi, B. (2015). On the application of Artificial Bee Colony (ABC) algorithm for optimization of well placements in fractured reservoirs; efficiency comparison with the Particle Swarm Optimization (PSO) methodology. *Petroleum*.

Odili, J. B. (2013). Application of Ant Colony Optimization to Solving the Traveling Salesman's Problem. *Science Journal of Electrical & Electronic Engineering, 2013*.

Odili, J. B., & Kahar, M. N. M. (2015a). African Buffalo Optimization (ABO): a New Meta-Heuristic Algorithm. *Journal of Advanced & Applied Sciences, 03*(03), 101-106.

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

Odili, J. B., & Kahar, M. N. M. (2016). African Buffalo Optimization. *International Journal of Software Engineering & Computer Systems, 2,* 28-50. doi:http://dx.doi.org/10.15282/ijsecs.2.2016.1.0014

Odili, J. B., & 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*, 641-648.

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

Odili, J. B., Mohmad Kahar, M. N., & Noraziah, A., Odili Esther Abiodun (2016). African Buffalo Optimization and the Randomized Insertion Algorithm for the Asymmetric Travelling Salesman's Problems *Journal of Theoretical and Applied Information Technology, 87*(3), 356-364.

Olafsson, S. (2006). Metaheuristics. *Handbooks in operations research and management science, 13,* 633-654.

Pereira, G. (2011). Particle Swarm Optimization. *INESCID and Instituto Superior Tecnico, Porto Salvo, Portugal, gpereira@ gaips. inesc-id. pt, Verified email at gaips. inesc-id. pt, April, 15*.

Peri, D., & Tinti, F. (2012). A multistart gradient-based algorithm with surrogate model for global optimization. *Communications in Applied and Industrial Mathematics, 3*(1).

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research, 3,* 1333-1356.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2011). *The Bees Algorithm–A Novel Tool for Complex Optimisation.* Paper presented at the Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference 3-14 July 2006.

Pletcher, R., Minkowycz, W., Sparrow, E., & Schneider, G. (1988). Overview of basic numerical methods. *Handbook of Numerical Heat Transfer*, 1-88.

Poli, R. (2007). An analysis of publications on particle swarm optimization applications. *Essex, UK: Department of Computer Science, University of Essex*.

Qu, Z., & Mo, H. (2011). Research of hybrid biogeography based optimization and clonal selection algorithm for numerical optimization *Advances in Swarm Intelligence* (pp. 390-399): Springer.

Ridge a't, F., Kudcnko, D., & Kazakov'i, D. (2005). Moving Nature-Inspired Algorithms to Parallel, Asynchronous and Decentralised Environments. *Self-Organization and Autonomic Informatics (I), 1,* 35.

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *Neural Networks, IEEE Transactions on, 5*(1), 96-101.

Schraudolph, N. N., & Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine learning, 9*(1), 9-21.

Selman, B., & Gomes, C. P. (2006). Hill-climbing Search. *Encyclopedia of Cognitive Science*.

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.

Simon, D. (2008). Biogeography-based optimization. *Evolutionary Computation, IEEE Transactions on, 12*(6), 702-713.

Simon, D., Ergezer, M., & Du, D. (2009). *Population distributions in biogeography-based optimization algorithms with elitism.* Paper presented at the Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on.

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

Sörensen, K., & Glover, F. W. (2013). Metaheuristics *Encyclopedia of operations research and management science* (pp. 960-970): Springer.

Stützle, T., López-Ibáñez, M., & Dorigo, M. (2011). A concise overview of applications of ant colony optimization. *Wiley Encyclopedia of Operations Research and Management Science*.

Tanweer, M., Suresh, S., & Sundararajan, N. (2015). *Improved SRPSO algorithm for solving CEC 2015 computationally expensive numerical optimization problems.* Paper presented at the Evolutionary Computation (CEC), 2015 IEEE Congress on.

Teodorović, D., & Dell'Orco, M. (2005). *Bee colony optimization–a cooperative learning approach to complex transportation problems.* Paper presented at the Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research.

Tyrrell, A. M., Hollingworth, G., & Smith, S. L. (2001). *Evolutionary strategies and intrinsic fault tolerance.* Paper presented at the Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on.

Vassiliadis, V., & Dounias, G. (2009). NATURE–INSPIRED INTELLIGENCE: A REVIEW OF SELECTED METHODS AND APPLICATIONS. *International Journal on Artificial Intelligence Tools, 18*(04), 487-516.

Venter, G. (2010). Review of optimization techniques. *Encyclopedia of aerospace engineering*.

Wedde, H. F., & Farooq, M. (2005). A performance evaluation framework for nature inspired routing algorithms *Applications of Evolutionary Computing* (pp. 136-146): Springer.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing, 4*(2), 65-85.

Wu, Y., Xin, Y., & Zhang, Y. (2015). Application of ACO to Vehicle Routing Problems Using Three Strategies.

Xi, B., Liu, Z., Raghavachari, M., Xia, C. H., & Zhang, L. (2004). *A smart hill-climbing algorithm for application server configuration.* Paper presented at the Proceedings of the 13th international conference on World Wide Web.

YANG, B., & ZHANG, Z.-k. (2004). Dynamic Characteristic Parameter Setting Method for Human-simulated Intelligent Controller. *Information and Control, 33*(6), 670-673.

Yang, X.-S. (2005). Engineering optimizations via nature-inspired virtual bee algorithms *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach* (pp. 317-323): Springer.

Yang, X.-S. (2009a). Firefly algorithms for multimodal optimization *Stochastic algorithms: foundations and applications* (pp. 169-178): Springer.

Yang, X.-S. (2009b). Harmony search as a metaheuristic algorithm *Music-inspired harmony search algorithm* (pp. 1-14): Springer.

Yang, X.-S. (2012). Nature-inspired mateheuristic algorithms: success and new challenges. *arXiv preprint arXiv:1211.6658*.

Yang, X.-S., Deb, S., & Fong, S. (2011). Accelerated particle swarm optimization and support vector machine for business optimization and applications *Networked digital technologies* (pp. 53-66): Springer.

Yeomans, J. S., & Yang, X.-S. (2014). Municipal waste management optimisation using a firefly algorithm-driven simulation-optimisation approach. *International Journal of Process Management and Benchmarking, 4*(4), 363-375.

Zielinski, K., & Laur, R. (2007). *Adaptive parameter setting for a multi-objective particle swarm optimization algorithm.* Paper presented at the IEEE Congress on.Evolutionary Computation, 2007. CEC 2007.

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