National Conference for Postgraduate Research
Universiti Malaysia PAHANG, Kuantan

Jan/2015

AFRICAN BUFFALO OPTIMIZATION (ABO): A NEW META-HEURISTIC ALGORITHM

Julius Beneoluchi Odili*, Mohd Nizam Mohmad Kahar

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

*Corresponding author. Tel: +60102384948
E-mail address: odili_julest@yahoo.com

Abstract

This paper proposes a new meta-heuristic approach to solving numerical and graph-based problems. The African buffalo algorithm evolved from an understanding of the animal's survival instincts and the search techniques they utilize in the African forests and savannahs; the search for the optimal path to pasture is aligned to their cooperative, intelligent, and social nature. The African Buffalo Optimization (A.B.O) algorithm simulates the African buffalos' behaviour by encapsulation in a mathematical model; which solves a number of discrete optimization problems using graph-based route planning, job scheduling and it extends Swarm Intelligence paradigms. When compared to the Ant Colony Optimization algorithm, Simulated Annealing and Genetic Algorithm, the results obtained from African Buffalo Optimization show that the algorithm works well and can be extended to solving problems like: path planning, scheduling, vehicle routing in addition to other constraint-driven problems.

Keywords:

Optimization
Soft computing
Travelling salesman problem
African Buffalo
Meta-heuristic

Accepted: 26 Dec2014

1. Introduction

The demands for an intelligent system in today's world have attracted many researchers to explore the area of soft computing. Some of the very popular studies include the Ant Colony Optimization [6], Artificial Bee Colony [3], Genetic Algorithm [2], Particle Swarm Optimization [1], and Differential Evolution (DE) [5] amongst many others. These techniques have been successfully applied to many combinatorial problems such as the travelling salesman's problem [3], job scheduling [2] and vehicle routing [2].

A critical evaluation of these algorithms revealed several flaws ranging from premature convergence [5] to complicated fitness function [3], inefficiency in the exploration of the search space [3], the use of several parameters [2], weakness in refining the search space at a later stage [1] and complex implementation strategies [4]. Some of these weaknesses informed the development of the African Buffalo

Optimization algorithm. This algorithm is derived from the observation of the African buffalos in documentaries on the National Geographic Television network. This animal has some unique intelligence, cooperative and navigational characteristics that enable it to sustain itself in the African forests and savannahs in spite of several competitors and a couple of predators [10]. These works models this beast's navigational ingenuity and apply the same to solve optimization problems ranging from Travelling Salesman's Problem, PID controller parameter tuning to Knapsack problem.

African Buffalo Optimization attempts to develop a totally-new algorithm that will demonstrate exceptional capacity in the exploitation and exploration of the search space. ABO solves the problem of pre-mature convergence or stagnation by making sure the position of the best buffalo is updated regularly and in the case where the best buffalo location is not improved in a number of iterations, the entire herd is re-initialised. Tracking the maaa (stay on to exploit) signals ensures adequate exploitation of the search space. In the same vein, tracking the waaa (move ahead) signal as well as tapping into the experience of other buffalos as well as that of the best buffalo enables the ABO to achieve adequate exploration. Similarly ABO ensures fast convergence with its use of very few parameters.

The first part of this paper highlights the motivation for this research, the second introduces the African Buffalo Optimization (ABO) algorithm. This is followed by an explanation of the basic flow of the algorithm indicating the general working and movements of the buffalos in search for a solution. The third section of the paper presents the experimental results and a detailed analysis of the results obtained. This is followed by the conclusion, recommendation for future research endeavours with the new algorithm and references.

2. Introducing the African Buffalo Optimization (ABO)

African Buffalo Optimization (ABO) is an attempt to simulate the alert 'maaa' sounds urging the buffalos to stay on to exploit and alarm 'waaaa' sounds urging the buffalos to move on to explore behaviour of African buffalos in its foraging assignments. With these sounds, the buffalos are able to optimize their search for food source. The ABO is a population-based algorithm in which individual buffalos work together to solve a given problem. Each buffalo within the ABO algorithm represents a solution within the search space.

2.1 The Basic Flow of the ABO

The algorithm begins by initializing the population of buffalos. It does this by allocating a random location to each buffalo within the solution space. Next, it updates each buffalo's fitness within the search space. If the fitness is

better than the individual buffalo's maximum fitness (bp_{max}), it saves the location vector for the particular buffalo. If the fitness is better than the herd's overall maximum, it saves it as the herd's maximum (bg_{max}). Finally it checks if the best buffalo is updating. If it is updating, it moves on to validate the stopping criteria. At this point, if our global best fitness meets our exit criteria, it ends the run and provides the location vector as the solution to the given problem. The ABO algorithm is shown is Figure 1.

Step1. Objective function $f(x)$ $x=(x1, x2, \ldots\ldots xn)^T$

Step2. Initialization: randomly place buffalos to nodes at the solution space;

Step3. Update the buffalos fitness values using Equation (1)

Step4.

$$w_{.k}+1 = w_{.k} + lpr_1(bg_{maxt.k} - m_{.k}) + lpr_2(bp_{max.k} - m_{.k}) \quad (1)$$

where $w_{.k}$ and $m_{.k}$ represents the exploration and exploitation moves respectively of the k^{th} buffalo (k=1,2,.....N) ; lp_1 and lp_2 are learning factors; r_1 and r_2 are random numbers between [0,1];
bg_{max} is the herd's best fitness and bp_{max} , the individual buffalo's best

Step5. Update the location of buffalo k in relation to $bp_{max.k}$ and $bg_{max.k}$ using (2)

$$m_{.k}+1 = \lambda (w_{.k} + m_{.k}). \quad (2)$$

λ is a unit of time

Step6. Is bg_{max} updating. Yes, go to 6. No, go to 2

Step7. If the stopping criteria is not met, go back to algorithm step 3

Step8. Output best solution.

Figure 1. ABO algorithm

Referring to Figure 1, it should be observed that the algorithm's Democratic equation has three parts: the first $w_{.k}$ represents the memory of the buffalos past location. The memory of each buffalo is a list of solutions that can be used as an alternative for the current local maximum location. There is a probability of choosing one of the target list of solutions of the buffalo's memory instead of the present herd's maximum point. The second $lpr_1(bg_{maxt.k} - m_{.k})$ is concerned with the cooperative part of the buffalos and is a pointer to the buffalo's social and information-sharing experience and then the third part $lpr_2(bp_{max.k} - m_{.k})$ indicates the intelligence part of the buffalos. So the ABO exploits the memory, intelligent and caring capabilities of the buffalos in arriving at solutions.

2.2 Buffalo Movement

Two main equations control the movement of buffalos within the solution space. These are Equations (1) and (2) (refer figure 1). The movement equation (1) provides for the actual movement of the herd using their specific speed. The maaa update (stay on to exploit) equation (2) provides movement adjustment given the two competing forces (bp_{max} *and* bg_{max}). The λ parameter which defines the discrete time interval over which the buffalo must move is usually set to 1.0. The result is a new location for the animal.

The first equation, otherwise called the Democratic equation, has three major parts, namely, w_k+1 (the animals' memory) indicating that animals are able to tell that they are in a new location distinct from location w_k; the global maximum and the personal maximum positions: each defining the representative influence over the animal's choices. The algorithm subtracts the maaa element (asking the animal to move ahead and explore) $m._k$ from the maximum vector and then multiplies this by a random number (r_1, r_2) usually between 0.0 to 0.6 and a learning factor (lp). (Using the random numbers between 0.0 to 0.6 has so far proved effective in obtaining fast convergence. Further investigation is ongoing to get figures that may yield better results). The sum of these products is then added to the waaa element (asking the animals to move on; to explore) for the given dimension of the sector. It should be emphasized that the random numbers give an amount of randomness in the path to help the animals move throughout the solution space. It does this by giving more or less emphasis to the global

Table 1 Comparative experimental result

Algorithms	Performance	Oliver30	Att48	Eil51	KroA200
		TSPLIB Value	TSPLIB Value	TSPLIB Value	TSPLIB Value
		423.74	**33522**	**426**	**29368**
ABO	Best	425.44	33524	426	29370
	Average	427.20	33579	427	30152
	Rel. Error	0.4%	0.2%	0.2%	2.66%
GA	Best	423.74	33818	445.80	30168
	Average	423.74	36649	458	-
	Rel. Error	0%	8.3%	2.7%	-
ACO	Best	452.96	39931	447.59	30082
	Average	479.57	41374.88	471.27	31057.02
	Rel. Error	5.8%	3.6%	5.3%	3.24%
SA	Best	425	34980	1275	-
	Average	438	35746	1392	-
	Rel. Error	3.0%	2.2%	9.1%	-

Rel. Error = Relative Error; Best = Best result obtained; Average = Average result after a number of runs

or personal maximum solutions depending on the need for more exploration or exploitation

respectively as the algorithm progresses. The primary difference between ABO and some related algorithm like the Artificial Bee Colony (ABC), Ant Colony Optimization (ACO) and the Particle Swarm Optimization (PSO) is that while the ABC exploits the sense of sight (watching closely the dance steps of the employed and scout bees); the ACO tracks the sense of smell (of pheromones); the PSO tracks the velocity and position of other the particles; the ABO tracks the sense of sound of other buffalos (waaa and maaa signals).

3. Experiments and Discussion of the Results

The ABO algorithm was used to solve the Travelling Salesman's Problem in order to validate its performance and the results obtained were compared with that of other meta-heuristic algorithms like the Ant Colony Optimization (ACO) [6], Genetic Algorithm (GA) [2] and the Simulated Annealing (SA) [9]. The dataset for the TSP test cases are available in TSPLIB95 [6] for Oliver30, Att48, Eil51and KroA200 [6] and the experiments were performed using Intel Duo Core ™ 2.00Ghz, 2.00Ghz, 1GB RAM on a Window7. To obtain the results in Table1 above, we used a population size of 40 and 50 iterations. The Relative Error ('Rel. Error') was obtained by calculating (Average ('Avg')-Best)/Best x 100/1

As can be seen in Table 1, the GA outperformed the ABO in realizing the optimal solution in Oliver30. Another great performer in Oliver30 is the Simulated Annealing that has just 3.0% relative error and then the African Buffalo Optimization. The worst performer here is the Ant Colony Optimization (ACO). However, the ABO outperformed the GA, ACO and the SA in other test cases, namely the Att48, Eil51 and the KroA2OO. It is important to note that the ABO was able to get the optimal result in Eil51. The GA and the ACO had fairly good performance in Eil51 in spite of their poor comparative performance in Att48. It was difficult obtaining experimental results about the Average results involving Genetic Algorithms for 200 cities, same with Simulated Annealing for 200 cities. In any case, we included the available results involving GA in order to prove the extremely good performance of the ABO above the GA in the available instances.

Generally, the ABO results are closer to discovering the optimal solution to the test cases than the other algorithms. This is due to the fact that the ABO uses fewer parameters in achieving its solutions. Aside the random numbers whose values range from 0-1, the main parameters are the lp1 and lp2. This is unlike the Genetic Algorithm that makes use of Evolution, Selection, Crossover and Mutations and the ACO that needs proper choice of its several parameters such as the pheromone update, pheromone evaporation rate, ant construction capabilities etc. Unlike GA whose performance is questionable in dynamic sets and is poor in constrained-based

optimization problems [8] or the ACO that has issues with convergence [9] and which has the disadvantage of falling into local minima due to its pheromone updates [9], the ABO has stable convergence, capacity to search large spaces, uses just two main parameters and is very simple to implement. The ABO performed well when compared to the SA, GA and ACO in terms of rate of convergence, the ABO was able to obtain optimal results within 40 iterations except in KroA200 that obtained the optimal result in the 90th iteration.

4. Conclusion

In general, this paper introduces the novel algorithm, the African Buffalo Optimization and shows its capacity to solve the Travelling Salesman's Problem. The performance obtained from using the ABO is weighed against the performance of the Genetic Algorithm (GA), Ant Colony Optimization (ACO) [6] and the Simulated Annealing [9]. The results showed amazing performance of the ABO and is a testimony that the ABO has immense potentials in solving optimization problems

5. Future work

To further validate this novel algorithm, further applications of the African Buffalo Optimization (ABO) to solve Job scheduling problems, knapsack problem and tuning PID Controller parameters are recommended

Acknowledgements

The authors are grateful to the Faculty of Computer Systems and Software Engineering for providing the facilities and the conducive environment to carry out this research.

References

[1] Angeline P. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences. *Evolutionary programming* VII. 1998; 601–610.

[2] Bin Y, Zhong.Zhen Y, Baozhen Y. An improved ant colony optimization for vehicle routing. *European Journal of Operational Research* 2009; *196, 171–176*

[3] M. Dorigo M, L. Gambardella, L. Ant colonies for the Traveling salesman problem. Biosystems. 1997.43, 73-81.

[4] Karaboga D., Akay B. A Modified Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems. *Apple Soft Computing* 2011, 11(3), 3021–3031

[5] Mezura-Montes E., Velázquez-Reyes J, Coello C.A. A comparative study of differential evolution variants for global optimization. *Proceedings of the 8th annual conference on Genetic and evolutionary computation.* 2006; 485–492

[6] Reinelt G. TSPLIB - A Travelling Salesman Problem Library, *ORSA Journal on Computing,* 1991, 3(4), Fall, 376-384

[7] Shi Y., Eberhart R. Parameter Selection in Particle Swarm Optimization. *Evolutionary programming* VII. 1998; 591–600

[8] Khaze S.R., Maleki I, Hojjatkhah S, A Bagherinia A. Evaluation of the Efficiency of Artificial Bee Colony and the Firefly Algorithm in Solving the Continuous Optimization. *International Journal on Computational Science & Applications* 2013; 3, 23-35

[9] Yan, X (et al). Solve Travelling Salesman's Problem using Particle Swarm Optimization. *International Journal of Computer Science Issues.* 2012. 9 6(2) 264-271

[10] Wilson D.S. Altruism and Organism: Disentangling the themes of multi-level selection theory. *The American Naturalist.* 1997. 150 (S1) pp122-134

[11] Stochastic Global Optimization Algorithms Workshop, University of Canterbury, Christchurch, New Zealand, June, 2005

[12]. Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. On Evolutionary Computation,* 1(1), pp.53-66, 1997.

[13] Karaboga, D., Akay B. A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation* 214 (2009) 108–132