African buffalo optimization for global optimization

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

[1]Department of Mathematical Sciences, Faculty of Natural and Applied Sciences, Anchor University, Lagos
[2]Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Kuantan 26300, Malaysia

In this study we apply the African buffalo optimization (ABO) to solve benchmark global optimization problems. Such problems which are artificial representation of different search landscapes ranging from unimodal to multimodal, separable to non-separable, constrained to unconstrained search landscapes have become a veritable instrument to test the search capacities of optimization algorithms. After a number of experimental procedures involving 28 benchmark problems, results from ABO prove to be rather competitive leading to the conclusion that it is a worthy addition to the body of swarm intelligence techniques.

Keywords: African buffalo optimization, global optimization, search landscapes, swarm intelligence techniques.

THE need for getting things done in a more effective and efficient manner makes optimization a favoured area of research. This trend has grown considerably in the last six decades leading to the development of several optimization search techniques whose contributions to scientific, engineering and industrial development have been invaluable. To date, some of the popular methods have been the genetic algorithm (GA)[1], particle swarm optimization (PSO)[2], artificial bee colony optimization (ABC)[3], firefly algorithm (FA)[4], bat algorithm (BA)[5], cuckoo search (CS)[6], teaching–learning based optimization (TLBO)[7] and Jaya algorithm (JA)[8]. The newly designed African buffalo optimization (ABO)[9] has proven to be quite successful in the areas it has so far been applied[10]. Hence this study proposes to investigate the capacity of ABO in global optimization test functions.

Benchmark global test functions have now become popular for investigating the capabilities of newly designed optimization techniques. The popularity of benchmark test functions is due to the fact that they represent diverse search landscapes imaginable from multimodal to unimodal, separable to non-separable and constrained to unconstrained. Investigating the capacity of ABO in these diverse landscapes and comparing the outcomes with those obtained from other optimization algorithms is therefore worthwhile.

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

Comparative algorithms

In this study a number of newly designed optimization algorithms, also called 21st century algorithms, are examined with special regard to their search capacities of benchmark global optimization test functions. The results obtained from these algorithms were compared with those from ABO. It is necessary to examine each of the comparative algorithms.

Firefly algorithm

This population-based algorithm which was inspired by the flashing behaviour of fireflies was developed by Yeomans and Yang[11]. In this algorithm, a number of fireflies work together 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 and this aids further exploration of the search space. The four main characteristics of the algorithm include[12]:

(a) All fireflies are unisexual; so they are attracted by brighter individuals, irrespective of the sex.

(b) The brightness of a firefly is a function of the distance between two fireflies: the nearer they are to one another, the more the effect of the brighter firefly and vice versa.

(c) If there are no fireflies brighter than a particular firefly then it will move 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. Figure 1 presents the pseudocode of FA[12]. FA has been successfully applied to industrial optimization, image processing, travelling salesman problem, antenna design, business optimization, civil engineering, robotics, semantic web, chemistry, meteorology, wireless sensor networks, etc.[13]. The algorithm is considered to have least error percentage compared to many other metaheuristic algorithms such as GA and PSO. Moreover, it is relatively simple to implement and has proven to perform well in multi-modal search environments. FA is similar to PSO, except that it does not employ search velocities in its quest for solutions[14]. However, it has complicated fitness

```
Initialize algorithm parameters: MaxGen: the maximal number of generations:
γ: the light absorption coefficient; r: the particular distance from the light source; d: the domain space

Define the objective function of f(x), where x = (x_1, …, x_d)^T
    Generate the initial population of fireflies or x_i (i = 1, 2, …, n)
    Determine the light intensity of I_i at x_i via f(x_i)

End
While (t < MaxGen)
    For i = 1 to n (all n fireflies);
        For j = 1 to n (n fireflies)
            If (i_j > L_i), move firefly i towards j using the firefly equation;
            End if
        Attractiveness varies with distance r via Exp[γr;
        Evaluate new solutions and update light intensity;
        End for j;
    End for i;
    Rank the fireflies and find the current best;
End while;
Post
Process results and visualization
End procedure
```

Figure 1. Firefly algorithm pseudocode.

```
Objective function: f(x) x = (x_1, x_2 …. x_j)
Determine initial population of nests
While termination not reached:
    Randomly generate a cuckoo by Levy flight
    Evaluate the cuckoo fitness
    Randomly select a nest among available host nests
        If (f_i > f_k); replace k by the new solution
        End if;
    Abandon a fraction of worse nets and replace with new
      ones
    Keep the best solutions;
    Rank the best solutions and obtain the current best;
End while;
Output the best result
```

Figure 2. Cuckoo search pseudocode.

function and highly depends on correct parameter setting to achieve good results. These drawbacks make FA less user-friendly, and the use of several parameters places a huge demand on computer resources, thus making it a relatively slow algorithm compared to PSO, for instance. In fact, as if to emphasize the complexity of FA, PSO is referred as a simplified version of FA[15]. So there is need for a more user-friendly and less complex metaheuristic algorithm.

Cuckoo search

CS which is a simulation of the crafty attitude of the cuckoos was designed by Yang and Deb[16]. These birds lay their eggs in the nests of other birds, sometimes of other species too. The host bird either incubates the cuckoo's eggs, throws it away or simply relocates its own eggs to another nest. In response to this, the cuckoo bird by making its own egg look as much as possible as the host's. In CS, the host eggs in a nest represent an optimization solution, while the cuckoo egg is a representation of a newer solution with the objective of using it to replace the existing one.

In CS, it is assumed that a cuckoo lays an egg at a time in any nest chosen randomly. Next, the nest with the best quality/number of eggs carries on to the next generation. Moreover, there are a fixed number of nests and the cuckoo egg is discovered by the host bird with a probability usually between 0 and 1. Figure 2 presents the CS pseudocode.

The algorithm, though relatively young, has enjoyed wide applications in different areas such as travelling salesman problems, document clustering, wireless sensor networks, speaker recognition, flood forecasting, shortest path in distributed system, image processing, classification task in health sector, job scheduling, etc.[17]. CS is fairly effective, obtaining results where other algorithms fail[17,18]. In spite of its wide applications, CS has the problem of speed due to the use of several parameters and sometimes falling into local optima[18]. Hence, the emergence of several types of the algorithms such as discrete CS, improved CS, etc.[19].

Bat algorithm

BA which is inspired by the echolocation of micro-bats that employ different pulse rates of loudness and

```
Objective function f(x), x = (x_1 ... x_k)^T
Initialize the population of bats, velocity V_i and position X_i
Determine the pulse frequency f_i at position x_i
Define the pulse rates r_1 and the loudness A_i
While (not termination) do
    Generate new solutions by adjusting frequency, updating velocities and positions using the
        frequency, velocity, position equations
        If (rand >r_l)
        Select a solution among the best solutions
        Select a local solution (position) around the selected best position (solution)
        End if
    Select another solution by getting the micro-bats to fly randomly
        If (rand < A_i and f(X_i) < f(x*)), do
        Accept the new solutions
        Increase the value of r_l and reduce A_i
        End if
End while
Rank the bats and determine the current best solution x*
Output the position of the best micro-bat
```

Figure 3. Pseudocode of Bat Algorithm.

```
Initialize
    PS ← Population size
    NDV ← Number _of design variables
    TER _ COD ← Termination _ condition
End
Until the termination condition, do repeat the next steps
Evaluate the best and worst solution
    Set best ← Best solution population
    Set worst ← Worst solution population
    Modify the solution
Update the previous solution
    Until No update in the previous solution
Output the best result
```

Figure 4. Jaya algorithm pseudocode.

emission was designed by Yang in 2010. In this algorithm, individual artificial bats fly randomly as they employ a velocity v_i at and at position (solution) x_i at a given dynamic frequency (wavelength) and loudness A_i. As the search process progresses, the micro-bats find a prey and change their frequency, pulse emission r and loudness. The local search component in this algorithm is done via a random walk. At each iteration, the algorithm selects the position of the best-performing bat as its solution until the stopping condition is reached.

So far, BA has been successful since its design and has enjoyed considerable applications such as in combinatorial optimization problems, parameter estimation and classification, image processing, data mining, etc. Its effectiveness is traceable to its simplicity, flexibility and straightforward implementation strategy. Moreover the algorithm provides quick convergence at the initial stages through switching from one exploration to another[20]. The pulse emission and loudness components of the algorithm help it to control and zoom into new regions (exploration). However, there is inherent danger in allowing BA to switch to exploitation stage too early by varying *rl* and *Ai* too fast; then, the algorithm settles into premature convergence. Moreover, the speed of BA is affected by several parameters since it appears to be a simplification of PSO. Invariably, it inherits the basic parameters of PSO, in addition to some of its own algorithm-specific parameters[21]. Figure 3 presents the pseudocode of BA[5].

Jaya algorithm

JA which was developed by Rao[8] is in furtherance of the new paradigm in algorithm development, namely parameter-less algorithms following the successful design of TLBO, a novel approach to optimization. JA operates on the premise that solutions are obtainable for any problem moving towards the best results and deliberately avoiding the worst solutions. Like other optimization algorithms, JA requires few control parameters such as population size, maximum number of generations and number of design variables. It does not require any problem-specific parameters that need to be tuned to get appropriate results. Figure 4 presents the pseudocode JA.

JA is a simple-to-implement algorithm and has been successfully applied to solve benchmark global optimization functions, travelling salesman problem as well as power flow problems[22]. Although JA is parameter-less, controlling the algorithm-specific parameters is not as easy task (see for example, Pandey[23]).

Set $k = 1$;
Objective function $f(x)$, $x = (x_i \ldots x_k)T$
Generate initial students of the classroom randomly x^j $i = 1 - n$
Calculate the objective function $f(x)$ for the whole of the students in the classroom
While (not termination) do:
    Calculate the average fitness value of each design variable x_{mean}
    Select the best solution (teacher)

    Teacher phase
    Calculate the mean of each design variable x_{mean}
    Identify the best solution (teacher)
        For $i = 1 \ldots n$
        Calculate teaching factor T_F^i round [1 + rand (0, 1) 2 – 1]
        Update solution based on the best solution (teacher)
        $x_{\text{new}}^i + x^i + \text{rand}(0,1)[x_{\text{teacher}} - (T_F^i - x_{\text{mean}})$
        Calculate objective function for new mapped student $f(x_{\text{new}}^i)$
            If x_{new}^i is better than x^i,
            Then $x^i = x_{\text{new}}^i$
            End if, that is, end of the teacher phase

    Student phase
    Choose another student (learner) x^j randomly
        If x^i has better fitness than x^j, Then $x_{\text{new}} = x^i + \text{rand}(0,1)...(x^j - x^i)$,
        Else $x_{\text{new}} = x^j + \text{rand}(0,1)\ (x^i - x^j)$
        End if.
    If x_{new}^i is better than x^i, Then $x^i = x_{\text{new}}^i$
    End if (student phase ends)
    End for.
    Set $k = k + 1$;
End while.
Output the best result

Figure 5. Teaching learning-based optimization pseudocode.

Teaching–learning based optimization

TLBO is a recently developed population-based optimization algorithm that searches for solutions through each learner's deliberate effort to attain the knowledge level of his/her teacher[7,24]. The algorithm approximates the view of teachers as the most knowledgeable individuals in the society. In TLBO, therefore, the teacher represents the optimum solution and a group of learners constitutes the population. The design variables in TLBO are the different subjects being offered to the learners and the learners' results represent the fitness. The search process of TLBO is divided into two phases – 'teacher phase' and 'learner phase'. In the former phase, the learners learn from the teacher, but in the 'latter phase' learners interact among themselves. Therefore, a learner with the minimum objective function value is deemed a teacher for the subsequent iteration. The algorithm obtains good results through a careful interplay of the teacher and learner phases. Figure 5 presents the pseudocode of TLBO[25].

TLBO has proven to be efficient and has enjoyed wide application since its development. The algorithm has input capacities that ensures wide exploration of the search space. TLBO has been applied to solve the travelling salesman problem, constrained and unconstrained global optimization test functions, robotics, motif discovery problems, vehicle routing, etc.[26] with good results. Though TLBO has been found to be good at exploration, it performs poorly in exploitation, resulting in premature convergence in complex problems[27].

The African buffalo optimization algorithm

Figure 6 presents the ABO algorithm, where w_k represents the *waaa* (move on/explore) signals of the buffalos with particular reference to buffalo k; m_k is the *maaa* signal (stay to exploit); w_k' is the request for further exploration; m_k' represents requests for further exploitation; lp1 and lp2 are the learning parameters; r is a random number that takes a value of 0 and 1 depending on the problem under investigation – higher the value, more the exploitation and less the exploration.

Since its design, ABO has been successfully applied to solve several optimization problems, such as the travelling salesman problem, strategic management[27], numerical function optimization, parameter-tuning of PID controllers for automatic voltage regulators, etc.[28].

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

Update the buffalos' exploitation using equation:

$$m'_k = m_k + \text{lp1}(bg - w_k) + \text{lp2}(bp.k - w_k)$$

where m_k and w_k represents exploitation and exploration moves respectively, of the kth buffalo ($k = 1, 2 \ldots N$); lp1 and lp2 are learning factors; bg is the herd's best fitness and bp is the individual buffalo's best location

Update the location of buffalos using the equation:

$$w'_k = \frac{(w_k + m_k)}{r}.$$

Is $bg_{\max}$ updating? Yes, go to 5. If No, go to 2

If the stopping criteria is not met, go back to update buffalos, else go to next step

Output best solution.

Figure 6. African buffalo optimization algorithm.

ABO for global optimization

In this study, ABO was implemented on a set of 16 standard global optimization test problems[28]. These 16 benchmark global optimization test functions are described[28]. These functions were chosen because they represent diverse global optimization landscapes and problems ranging from constrained to unconstrained, unimodal to multimodal, separable to non-separable. Moreover, previous studies have used these functions to test and compare various algorithms[29–31]. The experimental results obtained were compared with those from FA and intelligent firefly algorithm (IFA)[32]. These algorithms chosen to be compared with ABO have provided some of the best results in the literature.

ABO solution strategies for global optimization problems

In a global search space, two equations control the movement of buffalos within the solution space. These are the democratic equations:

$$m'_k = m_k + \text{lp1}(bg - w_k) + \text{lp2}(bp.k - w_k), \quad (1)$$

and decision equation:

$$w'_k = \frac{(w_k + m_k)}{\lambda}. \quad (2)$$

Equation (2) provides for the actual movement of the herd using the *maaa* (move on) signal. Eq. (1) provides an avenue for interaction among the competing buffalos tapping into two competing forces bp and bg. The actual movement equation is the decision (eq (2)) taking into cognizance the result obtained from the democratic eq. (1). In practical terms, the solution steps are as follows:

Step 1: Randomly initialize the buffalos within the solution space.

Step 2: Initialize ABO parameters, namely lp1 and lp2.

Step 3: Update buffalo's exploitation and determine the prevailing bp for each buffalo and bg for the entire herd using eq. (1):

Step 4: Update the location of the buffalos using eq. (2).

Step 5: Verify if bg is updating. If yes, go to step 6; if no, return to step 2.

Step 6: Validate stopping criteria. If yes, go to step 7; otherwise, return to step 3.

Step 7: Output the best result.

Experimental setting

The experiments performed were implemented in MATLAB on an Intel Duo Core™ i7-3770 CPU, 3.40 GHz with 4 GB RAM. The 16 benchmark global optimization test functions listed above were investigated. Table 1 presents the experimental parameters.

Benchmark global optimization test functions

To continue our validation process, ABO, FA and IFA were tested on 16 popular but difficult global optimization test functions. Table 2 provides a description of each of the 16 benchmark test functions.

A close examination of Table 2 reveals that the benchmark functions have been carefully chosen to represent diverse search landscapes such as unimodal to multimodal, constrained to unconstrained, separable to non-separable functions. This is necessary to provide a good testbed for the different algorithms under consideration. Table 3 presents the comparative performance results of the three algorithms under investigation. The table reveals that the three algorithms perform exceedingly

Table 1. Experimental parameters

African buffalo optimization		Firefly algorithm		Intelligent firefly algorithm	
Parameter	Value	Parameter	Value	Parameter	Value
Population	40	Population of ants	D^*	Population	D^*
lp1	0.6	γ	2	Mutation rate	0.5
lp2	0.5	MaxGen	1000	Crossover rate	1
N/A	–	α	0.5	γ	2
N/A		Q	200	Wind strength	[0, 1]
N/A	–		0.9	Wind angle	(0°, 360°)
N/A	–	N/A	–	MaxGen	1000
N/A	–	N/A	–	N/A	–
N/A	–	N/A	–	N/A	–
N/A	–	N/A	–	N/A	–
N/A	–	N/A	–	N/A	–
N/A	–	N/A	–	N/A	–
Total no of runs:	50	50	50		

D^* represents dimension of the optimization problem which, in this case is the number of nodes; Q is the pheromone amount; α is the exploitation ratio.

Table 2. Benchmark functions

Objective function	Xter	[n]var	Search domain	Global minimum
Ackley[u]	US	2	[−35, 35]	0
Beale[u]	MS	2	[−4.5, 4.5]	0
Booth[u]	MS	2	[−10, 10]	0
Carrom table[c]	MS	2	[−10, 10]	−24.157
Cross-leg table[c]	MN	2	[−10, 10]	−1
Himmelblau[c]	MS	2	[−5, 5]	0
Levy 13[c]	MS	2	[−10, 10]	0
Schaffer[c]	MN	2	[−100, 100]	0
Powell[u]	UN	4	[−1000, 1000]	0
Cube[c]	UN	5	[−100, 100]	0
Sphere[u]	US	5	[−100, 100]	0
Egg holder[c]	MN	50	[−512, 512]	–959.6407
Griewank[u]	MN	50	[−600, 600]	0
Rastrigin[u]	MS	50	[−5.12, 5.12]	0
Rosenbrock[u]	UN	50	[−50, 50]	0
Zacharov[u]	MN	50	[−5, 10]	0

[n]var, Number of variables required to optimize the function; Search domain, Search range; Global minimum, Benchmark global minimum of the function; Xter, Characteristic of the benchmark function; US, Unimodal and separable; MS, Multimodal and separable; MN, Multimodal and non-separable; UN, unimodal and non-separable. Note that superscripts c or u after each function name indicate whether the benchmark function is constrained or unconstrained respectively.

well, obtaining optimal results in all of the runs as indicated by the standard deviation of zero in some of the global optimization test cases. For instance, all the algorithms obtained optimal solution in $f1$, $f3$, $f7$ and $f11$. This is commendable, especially when one considers the fact that these are varying landscapes of constrained and unconstrained functions. A closer observation, however, reveals that the above four benchmark functions represent three unconstrained and one constrained function. This outcome is in agreement with earlier findings that the constrained function requires a longer procedure for optimization, because it may have to be first converted to unconstrained function[33–35].

Conversely, all the algorithms had their worst performance in the egg-holder function (Figure 7). The global minimum of this function is –959.6407. Mean results of the algorithms show that they are far-off from the global minimum. For example, FA was hovering around –1.5200, IFA around –2.5000 and ABO around –66.7834. From these results, it can be argued that ABO is nearer to the optimum, though still very far-off.

The egg-holder is indeed a difficult function to optimize and has a deceptive landscape that easily misleads search agents because of the large number of local minima[30].

Another difficult benchmark function for all the algorithms under investigation here is the cross-leg table function. The global optimum of this multimodal function is –1. Again, due the flat nature of this artificial landscape (Figure 8), it provides misleading information leading to poor results.

Table 3. Simulation results

	FA		ABO		IFA	
Objective function	Mean	SD	Mean	SD	Mean	SD
$f1$ Ackley	0	0	0	0	0	0
$f2$ Beale	0	0	6.3446	4.42	0	0
$f3$ Booth	0	0	0	0	0	0
$f4$ Carrom table	−23.43	1.75	−18.18	0	−24.16	0
$f5$ Cross-leg table	−0.066	0.10	−0.80433	0	−746	0.26
$f6$ Himmelblau	0	0	92.9882	79.63	0	0
$f7$ Levy 13	0	0	0	0	0	0
$f8$ Schaffer	0.085	0.034	0	0	0.0035	0.034
$f9$ Powell	1.8400	2.7000	0	0	0.000017	0.00003
$f10$ Cube	88.69	77.05	0	0	41.39	65.09
$f11$ Sphere	0	0	0	0	0	0
$f12$ Egg holder	−1.5200	1.600	−66.7834	0.003	−2.5000	0.021
$f13$ Griewank	0.0162	0.00061	0	0	0	0
$f14$ Rastrigin	45.47	12.7	0	0	66.63	16.44
$f15$ Rosenbrock	296.8	652.3	0	0	70.57	109.6
$f16$ Zacharov	36.68	13.2	0	0	0	0

Mean, The mean score after 30 independent runs of the algorithms; SD, Standard deviation from the mean of the run.

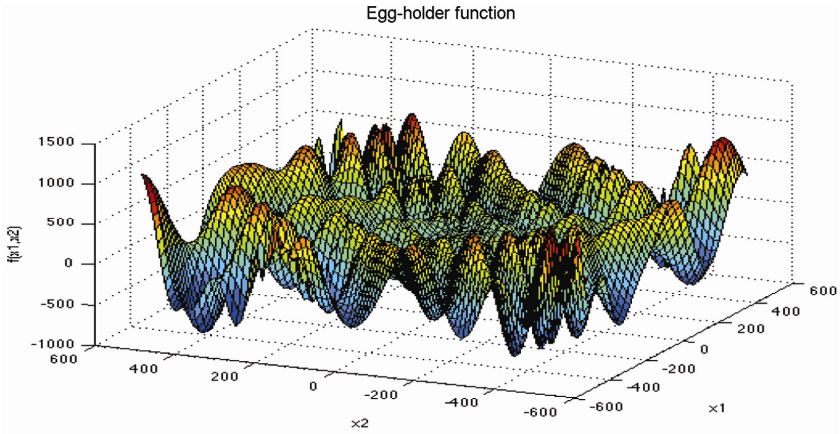

Figure 7. Egg-holder function.

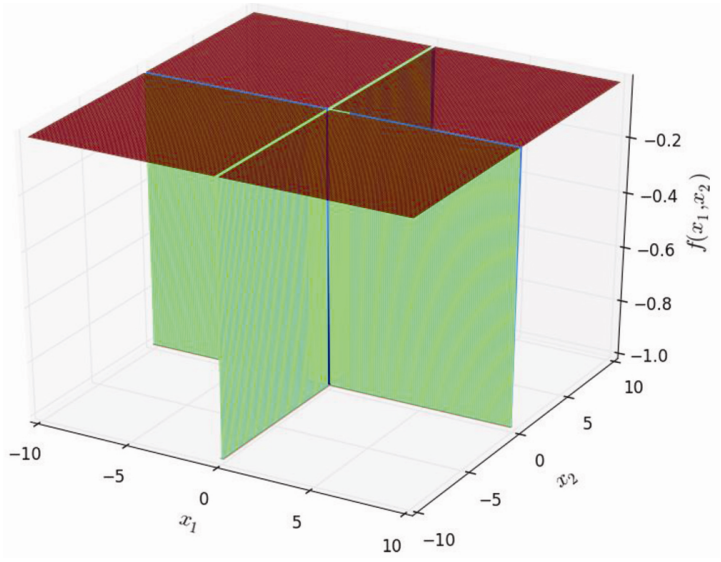

Figure 8. Cross-leg table function.

From the simulation outcomes, again, ABO was closer to the optimum with –0.80433 to –0.066 of FA and –746 of IFA. Actually, IFA was totally off-the-mark in tracking the global minimum of this function; however, the performance of ABO here is commendable.

On algorithm-by-algorithm basis, ABO showed the best performance with 11 optimal solutions out of the 16 test cases; followed by IFA with nine optimal solution and then FA with six optimal solutions. Since effectiveness in obtaining the optimal solution is one of the aspects of a good algorithm[36], ABO can be rightly adjudged as a better algorithm here. Similarly, in terms of standard deviation from the mean results, ABO has the minimum deviation among the three algorithms with total standard deviation of 84.053 to 191.45 of IFA and 671.43 of FA.

Table 4. Chi-square test output from FA, ABO and IFA

	Observed *N*	Expected *N*	Residual
FA			
–2.4300	1	1.5	–0.5
–1.5200	1	1.5	–0.5
–0.0660	1	1.5	–0.5
0.0000	6	1.5	4.5
0.0162	1	1.5	–0.5
0.0850	1	1.5	–0.5
1.8400	1	1.5	–0.5
36.6800	1	1.5	–0.5
45.4700	1	1.5	–0.5
88.6900	1	1.5	–0.5
296.8000	1	1.5	–0.5
Total	16		
ABO			
–66.7834	1	2.7	–1.7
–18.1800	1	2.7	–1.7
0.0000	11	2.7	8.3
0.8043	1	2.7	–1.7
6.3446	1	2.7	–1.7
92.9882	1	2.7	–1.7
Total	16		
IFA			
–24.1600	1	1.8	–0.8
–2.5000	1	1.8	–0.8
0.0000	8	1.8	6.2
0.0000	1	1.8	–0.8
0.0035	1	1.8	–0.8
41.3900	1	1.8	–0.8
66.6300	1	1.8	–0.8
70.5700	1	1.8	–0.8
746.0000	1	1.8	–0.8
Total	16		

Table 5. Test statistics results

	FA	ABO	IFA
Chi-square	15.625[a]	31.250[b]	24.500[c]
df	10	5	8
Asymp. sig.	0.111	0.000	0.002

[a]11 cells (100.0%) have expected frequencies less than 5. The minimum expected cell frequency is 1.5. [b]Six cells (100.0%) have expected frequencies less than 5. The minimum expected cell frequency is 2.7. [c]Nine cells (100.0%) have expected frequencies less than 5. The minimum expected cell frequency is 1.8.

The output of the three comparative algorithms were statistically computed using Chi-square test (Tables 4 and 5).

H_o: The output of ABO is significantly better statistically than those of FA and IFA.

H_A: There is no significant difference between the outputs of ABO, FA and IFA.

Statistically, as can be seen from Table 5, with 0.5 degree of freedom, showed a value of ABO 0.000 to 0.002 of IFA and 0.111 of FA. From this result, it is obvious that there is significant difference in the performance of ABO in obtaining solutions to the 16 benchmark test cases under investigation; so we reject the null hypothesis.

More experimental evaluations

In view of the competitive outcomes of the three swarm-based techniques in the first set of experiments, it is necessary to assess the performance of ABO with other population-based algorithms on global optimization functions. The comparative population-based algorithms represent some of the most successful optimization algorithms, namely GA and PSO as well as some newly designed optimization methods like ABC, TLBO and JA. The benchmark functions investigated are the sphere, beale, Easom, Matyas, Zakhorov, Schwefel1.2, Rosenbrock, Branin, Bohachevsky1, Booth, Michalewicz2, Michalewicz5, Bohachevsky2, Bohachevsky3, Goldstein Price, Perm and Langerman2 functions. The comparative results are obtained from Rao[8]. Table 6 presents experimental outcomes. It records the performances of some of the best algorithms in the literature. Out of the 17 benchmark functions considered, JA produced the best result since it was able to obtain optimal solutions in all test cases, except in *f*14 and *f*18, followed by ABC with 13 and TLBO with excellent results in 12 out of the 17 instances. TLBO did not do so well in *f*9, *f*11, *f*14, *f*18 and *f*22. ABO followed closely with excellent results in nine out the 17 instances, except in *f*3, *f*4, *f*10, *f*17, *f*18, *f*21, *f*22 and *f*23. PSO and GA did well too, obtaining optimum outcomes in eight instances each.

Conclusion

In this study we examined the application of ABO to solving global optimization benchmark functions. A number of experiments on different landscapes ranging from unimodal to multimodal, constrained to unconstrained, separable to non-separable functions were carried

Table 6. ABO with genetic algorithm, particle swarm optimization, artificial bee colony, teaching learning-based optimization and Jaya algorithms

Functions	Statistics	GA	PSO	ABO	ABC	TLBO	Jaya
$f1$	M	1.11E+03	0	0	0	0	0
	SD	74.214474	0	0	0	0	0
$f3$	M	0	0	6.34	0	0	0
	SD	0	0	4.42	0	0	0
$f4$	M	–1	–1	–0.9	–1	–1	–1
	SD	0	0	0	0	0	0
$f5$	M	0	0	0	0	0	0
	SD	0	0	0	0	0	0
$f9$	M	0.013355	0	0	0.0002476	0	0
	SD	0.004532	0	0	0.000183	0	0
$f10$	M	7.40E+03	0	830	0	0	0
	SD	1.14E+03	0	0	0	0	0
$f11$	M	1.96E+05	15.088617	0	0.0887707	1.62E–05	0
	SD	3.85E+04	24.170196	0	0.07739	3.64E–05	0
$f14$	M	0.998004	0.9980039	0.5424	0.9980039	0.9980039	0.998004
	SD	0	0	0	0	0	0
$f15$	M	0	0	0	0	0	0
	SD	0	0	0	0	0	0
$f16$	M	0	0	0	0	0	0
	SD	0	0	0	0	0	0
$f17$	M	–1.8013	–1.5728692	–0.9824	–1.8013034	–1.801303	–1.801303
	SD	0.00E+00	0.11986	0.0159	0	0	0
$f18$	M	–4.64483	–2.4908728	1.000	–4.6876582	–4.6726578	–4.680138
	SD	0.09785	0.256952	0	0	4.74E–02	1.58E–02
$f19$	M	0.06829	0	0	0	0	0
	SD	0.078216	0	0	0	0	0
$f20$	M	0	0	0	0	0	0
	SD	0	0	0	0	0	0
$f21$	M	5.870093	3	23.9691	3	3	3
	SD	1.071727	0	14.2657	0	0	0
$f22$	M	0.302671	0.0360516	0.2751	0.0411052	0.0006766	0
	SD	0.193254	0.048927	0.2058	0.023056	0.0007452	0
$f23$	M	–1.08094	–0.679268	–4.1271	–1.0809384	–1.080938	–1.080938
	SD	0	0.274621	0	0	0	0
$f24$	M	14.67178	0.1646224	0	0	0	0
	SD	0.178141	0.493867	0	0	0	0

out. The results thus obtained were compared with those from other techniques. The outcome revealed the capacity of ABO to obtain competitive results in solving global optimization test cases. It is, therefore, recommended that ABO be applied to solve other optimization problems such as graph-colouring problem, *N*-queen and urban transportation problems to further validate its search capacity.

1. Pricopie, A. and Costache, A., In *The 1940 Vrancea Earthquake. Issues, Insights and Lessons Learnt*, Springer, New York, 2016, pp. 363–375.
2. Kennedy, J. In *Encyclopedia of Machine Learning*, Springer, New York, 2011, pp. 760–766.
3. Karaboga, D., Artificial bee colony algorithm. *Scholarpedia*, 2010, **5**, 6915.
4. Yang, X.-S., Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput*., 2010, **2**, 78–84.
5. Yang, X.-S., In *Nature Inspired Cooperative Strategies for Optimization* (NICSO 2010), Springer, New York, 2010, pp. 65–74.
6. Gandomi, A. H., Yang, X.-S. and Alavi, A. H., Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Eng. Comput*., 2013, **29**, 17–35.
7. Rao, R. V., Savsani, V. J. and Vakharia, D., Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Comput-Aided Des*., 2011, **43**, 303–315.
8. Rao, R., Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput*., 2016, **7**, 19–34.

9. Odili, J. B., Kahar, M. N. M. and Anwar, S., African buffalo optimization: a swarm-intelligence technique. *Proc. Comput. Sci*., 2015, **76**, 443–448.
10. Odili, J. B. and Mohmad Kahar, M. N., Solving the traveling salesman's problem using the African buffalo optimization. *Comput. Intell. Neurosci*., 2015, **501**, 1–12.
11. Yeomans, J. S. and Yang, X.-S., Municipal waste management optimisation using a firefly algorithm–driven simulation-optimisation approach. *Int. J. Process Manage. Benchmark*., 2014, **4**, 363–375.
12. Matsushita, H., In *IEEE Congress on Evolutionary Computation*, Sendai, Japan, 2015, pp. 2672–2677.
13. Fister, I., Yang, X.-S. and Brest, J., A comprehensive review of firefly algorithms. *Swarm Evol. Comput*., 2013, **13**, 34–46.
14. Yang, X.-S. Nature-inspired metaheuristic algorithms: success and new challenges. arXiv preprint arXiv:1211.6658, 2012.
15. Kavousi-Fard, A., Samet, H. and Marzbani, F., A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. *Expert Syst. Appl*., 2014, **41**, 6047–6056.
16. Yang, X.-S. and Deb, S., In IEEE World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), Coimbatore, India, 2009, pp. 210–214.
17. Kamat, S. and Karegowda, A., A brief survey on cuckoo search applications. *Int. J. Innov. Res. Comput. Commun. Eng.*, 2014, **2**, 7–14.
18. Ouaarab, A., Ahiod, B. and Yang, X.-S., Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Comput. Appl*., 2014, **24**, 1659–1669.
19. Marichelvam, M., Prabaharan, T. and Yang, X.-S., Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Appl. Soft Comput*., 2014, **19**, 93–101.
20. Fister, I., Rauter, S., Yang, X.-S. and Ljubič, K., Planning the sports training sessions with the bat algorithm. *Neurocomputing*, 2015, **149**, 993–1002.
21. Yang, X.-S. and He, X., Bat algorithm: literature review and applications. *Int. J. Bio-Inspired Comput*., 2013, **5**, 141–149.
22. Warid, W., Hizam, H., Mariun, N. and Abdul-Wahab, N. I., Optimal power flow using the Jaya algorithm. *Energies*, 2016, **9**, 678.
23. Pandey, H. M., Cloud System and Big Data Engineering (Confluence), In 6th IEEE International Conference, Noida, India, 2016, pp. 728–730.
24. Rao, R. V., Savsani, V. J. and Vakharia, D., Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. *Inf. Sci*., 2012, **183**, 1–15.
25. Baghlani, A. and Makiabadi, M., Teaching–learning-based optimization algorithm for shape and size optimization of truss structures with dynamic frequency constraints. *Iran. J. Sci. Technol. Trans. Civil Eng*., 2013, **37**, 409.
26. Rao, R., Review of applications of TLBO algorithm and a tutorial for beginners to solve the unconstrained and constrained optimization problems. *Decis. Sci. Lett*., 2016, **5**, 1–30.
27. Ge, F., Hong, L. and Shi, L., An autonomous teaching–learning based optimization algorithm for single objective global optimization. *Int. J. Comput. Intel. Syst*., 2016, **9**, 506–524.
28. Ali, M. M., Khompatraporn, C. and Zabinsky, Z. B., A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. Global Optim*., 2005, **31**, 635–672.
29. Hedar, A.-R. and Fukushima, M., Tabu search directed by direct search methods for nonlinear global optimization. *Eur. J. Oper. Res*., 2006, **170**, 329–349.
30. Bingham, D., Virtual library of simulation experiments: test functions and databases, 2015; https://www.sfu.ca/~ssurjano.
31. Mishra, S. K., Some new test functions for global optimization and performance of repulsive particle swarm method, 2006; SSRN 926132.
32. Fateen, S.-E. K. and Bonilla-Petriciolet, A., In *Cuckoo Search and Firefly Algorithm*, Springer, 2014, pp. 315–330.
33. Odili, J. B., Kahar, M. N. M. and Noraziah, A., African buffalo optimization and the randomized insertion algorithm for the asymmetric travelling Salesman's problems. *J. Theoret. Appl. Infor. Technol*., 2016, **87**(3) 356–364.
34. Odili J. B., Kahar, M. N. M. and Noraziah, A., Convergence analysis of the African buffalo optimization algorithm. *Int. J. Simul. Sci. Technol*, United Kingdom Simulation Society, 2016, **17**(33), 44.1–44.7.
35. Odili, J. B., Kahar, M. N. M. and Noraziah, A., African buffalo optimization strategy for tuning parameters of a PID controller in automatic voltage regulators. *Int. J. Simul., Sci. Technol.*, 2016, **17**(33), 45.1–45.6.
36. de Oliveira, J. V., Semantic constraints for membership function optimization. *IEEE Trans. Syst. Man, Cybern. – Part A*, 1999, **29**, 128–138.

ACKNOWLEDGEMENTS. This study was supported by the Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang under Grant GRS 1403118.

Received 30 December 2016; revised accepted 31 August 2017

doi: 10.18520/cs/v114/i03/627-636