Elite Opposition-based Artificial Bee Colony Algorithm for Global Optimization

Z. Guo*, S. Wang, X. Yue, D. Jiang, K. Li

*School of Science, JiangXi University of Science and Technology, Ganzhou, China
School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang, China
State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan, China
School of Mathematics and Informatics, South China Agricultural University, Guangzhou, China

Abstract

Numerous problems in engineering and science can be transformed into optimization problems. Artificial bee colony (ABC) algorithm is a newly developed stochastic optimization algorithm and has been successfully used in many areas. However, due to the stochastic characteristics of the solution search equation, the traditional ABC algorithm often suffers from poor exploitation. Aiming at this weakness of the traditional ABC algorithm, in this paper, we propose an enhanced ABC algorithm with elite opposition-based learning strategy (EOABC). In the proposed EOABC, it executes the elite opposition-based learning strategy with a preset learning probability to enhance the exploitation capacity. In the experiments, EOABC is tested on a set of numerical benchmark test functions, and is compared with some other ABC algorithms. The comparisons indicate that EOABC can obtain competitive results on the majority of the test functions.

doi: 10.5829/idosi.ije.2015.28.09c.03

1. INTRODUCTION

Optimization problems widely exist in engineering applications [1]. Therefore, it is of significance to develop effective and efficient optimization algorithms for practical problems [2-4]. Evolutionary algorithm (EA) is a very promising approach for optimization algorithms, which has been successfully applied to many practical applications [5-8]. Artificial bee colony algorithm (ABC), recently developed by Karaboga and Basturk [9], is a kind of EA that mimics the foraging behavior of the honey bee swarm in nature. Like other EAs [10, 11], ABC has a very simple structure. As ABC is easy to implement and has exhibited encouraging performance in many problems from various fields, it has been quickly developed in recent years and has been successfully applied in solving diverse real-world optimization problems [12]. In practice, ABC has been compared with many other optimization techniques such as Genetic Algorithm (GA) [13], Particle Swarm Optimization (PSO) [14, 15] and Differential Evolution (DE) [16]. The comparison results reveal that the performance of ABC can outperform other optimization techniques in many optimization problems [12].

Although ABC has successfully solved a wide variety of optimization problems from various areas, it has several weaknesses when solving complex optimization problems. One of the weaknesses of the traditional ABC is that it may suffer from poor exploitation when solving complex optimization problems [12]. As known, both exploration and exploitation are very critical for EAs [17]. Therefore, in some cases, the traditional ABC cannot find satisfactory results. Accordingly, various ABC variations have been proposed to promote the search capability of the traditional ABC. In order to enhance the exploitation ability, Zhu et al. [18] proposed a gbest-guided ABC (GABC), which combines the information of the global best solution into the solution search equation. Gao and Liu [19] presented an improved ABC with two improved solution search equations which are inherited from DE. El-Abd [20] introduced an opposition-based ABC (OABC), which incorporates the opposition-based learning strategy. Banharnsakun et al. [21] proposed an
improved ABC with best-so-far selection. In the best-so-far ABC, all the solutions in the current population share the best feasible solutions found so far. Akay and Karaboga [22] presented a modification on ABC by introducing the perturbation frequency and the magnitude of the perturbation. To improve the exploitation ability, Gao and Liu [23] introduced an improved ABC, which utilizes the best solution of the previous iteration.

Karaboga et al. [24] presented a symbolic regression approach using ABC programming (ABCP). By incorporating the generalized opposition-based learning strategy, El-Abd [25] proposed an enhance ABC (GOABC) for global optimization. To improve the exploration power, Chen et al. [26] introduced the simulated annealing (SA) algorithm into ABC. Luo et al. [27] enhanced the solution search equation in the onlooker phase using the best solution of the previous generation to direct the search process. Gao et al. [28] proposed a modified ABC with improved search equation and the orthogonal learning (OL) strategy. Xiang and An [29] introduced an efficient and robust ABC (ERABC), which utilizes a combinatorial solution search equation to accelerate the convergence speed. Kang et al. [30] proposed a hybridized ABC, which embeds the pattern search scheme. Bansal et al. [31] presented a modified ABC with memetic search strategy. Karaboga and Gorkemli [32] proposed a quick ABC, which introduces the neighborhood radius into the search step of the onlooker bees. Gao et al. [33] presented an enhanced ABC utilizing more information-based search equations. Xiang et al. [34] introduced a hybrid ABC embedding the search operation of DE. Kiran and Findik [35] proposed a directional ABC (dABC) which employs the directional information to improve the search ability of ABC.

In our previous work [36], we proposed an elite opposition-based learning (EOBL) strategy and employed it to enhance the exploitation ability of the traditional DE. Our previous experimental results [36] indicated that EOBL can significantly promote the exploitation ability of the traditional DE. Therefore, it is expected that the exploitation capability of the traditional ABC can also be enhanced by EOBL. Motivated by these considerations, we propose an enhanced ABC (EOABC) through the utilization of EOBL. In the proposed EOABC, at each generation, the EOBL strategy is performed with a preset learning probability to enhance the exploitation ability. In addition, EOABC has a very simple framework and thus is easy to implement.

The rest of the paper is structured as follows. Section 2 describes the traditional ABC algorithm. The elite opposition-based ABC is presented in section 3. Numerical results and comparisons are reported in section 4. Finally, section 5 concludes the paper.

2. ARTIFICIAL BEE COLONY

ABC is a newly developed meta-heuristic algorithm, which simulates the intelligent foraging behavior of the honey bee swarm in nature [9, 12]. In ABC, three kinds of bees, namely employed bees, onlooker bees and scout bees, are used to seek the global optimal solutions for a optimization problem at hand [12]. Moreover, each employed bee is associated with a solution, and the employed bees aim to exploit its associated solution and gather the information of the exploited regions to the onlooker bees. The onlooker bees focus on selecting the excellent solutions to be further exploited through the information provided by the employed bees. In the search process, if the quality of a solution has not been enhanced through a preset number of cycles, this solution is assumed to be abandoned by its employed bee, and then the associated employed bee becomes a scout bee that starts to generate a new solution by randomly sampling from the feasible search space.

Without loss of generality, we suppose in this study that ABC is for solving the minimization problem Min \( f(X) \), where \( X = [x_1, x_2, \ldots, x_D] \), and the search space is:

\[
\Omega = \prod_{j=1}^{D} [LB_j, UB_j]
\]

where \( D \) is the dimension of the minimization problem, \( LB_j \) and \( UB_j \) are the lower and upper boundaries of the search space, respectively.

Like other EAs, ABC also consists of a very simple procedure. At the initialization stage, an initial population \( P(t) = \{X'_1, \ldots, X'_t\} \) is randomly generated from the domain of the minimization problem:

\[
x'_i = LB_j + \text{rand}(0,1) \times (UB_j - LB_j)
\]

where \( X' = [x'_1, x'_2, \ldots, x'_D] \), \( i = 1, 2, \ldots, SN \); \( j = 1, 2, \ldots, D \); \( t \) represents the generation, and \( SN \) is the number of solutions in the population, which is also equal to the size of the employed bees or onlooker bees [9]. and \( \text{rand}(0,1) \) is a random real number in the range [0, 1].

Following the initialization stage, ABC executes a loop of search operations until the termination condition is satisfied. In loop of the search operations, each employed bee creates a neighborhood solution of its associated solution according to the following equation [9]:

\[ v'_{i,j} = x'_{i,j} + \phi_{i,j} \cdot (x'_{j,k} - x'_{i,k}) \]

where \( j \) is an integer randomly selected from the set \( \{1, 2, \ldots, D\} \), index \( k \) is an integer randomly selected from the set \( \{1, 2, \ldots, SN\} \setminus \{i\} \), and \( \phi_{i,j} \) is a random real number uniformly distributed within [-1, 1] [12]. After creating the neighborhood solution \( V'_{i} \), its fitness value \( Fit'_{i} \) is evaluated by [9]:
\[
F_{i}^{*} = \begin{cases} 
1 & \text{if } f(V_{i}^{*}) \geq 0 \\
1 + \text{abs}(f(V_{i}^{*})) & \text{if } f(V_{i}^{*}) < 0 
\end{cases}
\] (4)

where \( f(.) \) is the objective function of the minimization problem. Then, each employed bee selects the better one between its associated solution and the corresponding created neighborhood solution to enter the next generation [12]. Once all employed bees have finished their search operations, they will provide the information of the exploited solutions to the onlooker bees. After that, each onlooker bee randomly selects a solution for further exploitation according to the selection probability of each solution. Moreover, the selection probability of each solution is calculated by [9]:

\[
p_{i} = \frac{F_{i}^{*}}{\sum_{i=1}^{39} F_{i}^{*}}
\] (5)

Subsequently, each selected solution is further exploited by the corresponding onlooker bee using Equation (3). Each onlooker bee also utilizes the greedy selection scheme to select the better one between its selected solution and exploited solution to enter the next generation.

After all onlooker bees complete their search procedure, ABC finds the solution whose quality cannot be improved through a preset number of cycles, called limit, then the corresponding employed bee associated with the found solution becomes a scout bee which will reinitialize its associated solution by randomly sampling from the feasible search space.

3. PROPOSED ALGORITHM

In this section, we propose an enhanced ABC algorithm with elite opposition-based learning strategy (EOABC). First, we introduce the notations of the traditional opposition-based learning strategy. Then, the elite opposition-based learning strategy is presented. Finally, the detailed computation steps of EOABC are elaborated at the end of this section.

3.1. Opposition-based Learning

Opposition-based learning (OBL) strategy is a relatively new intelligent computation technique, which is firstly proposed by Tizhoosh [37]. Since its introduction, OBL has attracted many researchers in recent years and has been successfully incorporated in several EAs [38]. According to extensively reported theoretical and experimental studies, OBL is an effective approach for enhancing the performance of EAs [39]. As known, the opposition concept can be widely observed in real-life, such as opposite particles in physics, and opposition parties in politics [40]. Inspired by the opposition concept in real-life, the opposition idea is introduced into the evolutionary computation fields. The definition of opposite solution is described as follows.

Let \( X_{t} = [x_{1,t}, x_{2,t}, \ldots, x_{D,t}] \) represent the \( i \)th solution in the current population at generation \( t \). Its corresponding opposite solution \( O_{t} = [o_{1,t}, o_{2,t}, \ldots, o_{D,t}] \) is calculated by [38]:

\[
o_{i,j} = A_{i} + B_{i}' - x_{i,j} \\
A_{i}' = \min(x_{i,j}, B_{i}') \\
B_{i}' = \max(x_{i,j}) \\
o_{i,j} = \text{rand}(A_{i}', B_{i}'), \text{ if } o_{i,j} < LB_{i} \parallel o_{i,j} > UB_{i}
\] (6)

where \( x_{i,j} \) is the \( j \)th component of the \( i \)th solution in the current population, \( o_{i,j} \) is the opposite value of \( x_{i,j} \), \( A_{i}' \) and \( B_{i}' \) are the minimum and maximum values of the \( j \)th dimension of the current population at generation \( t \), respectively.

As known, due to lack of priori information about the optimum solution, many EAs utilize random guess strategies to generate solutions. However, random guess strategies are often time-consuming and inefficient. In fact, it is a potential scheme to increase the chance of finding a better solution by simultaneously considering the opposite solution. In terms of the probability theory, 50% of the time, a guess is further from the solution than its opposite guess [38]. Therefore, simultaneously considering each solution and its corresponding opposite solution can accelerate the convergence speed [38]. Based on the fundamental idea of OBL, Wang et al. have presented a generalized opposition-based learning strategy (GOBL), which introduces a random generalized coefficient to enhance the search capability of OBL [39]. Accordingly, many researchers have employed GOBL to enhance the performance of EAs [39]. The concept of GOBL is expressed as follows.

For solution \( X_{i}^{t} \), its generalized opposition-based solution \( GO_{i}^{t} = [go_{1,i}^{t}, go_{2,i}^{t}, \ldots, go_{D,i}^{t}] \) is defined by [39]:

\[
go_{i,j} = k \cdot (A_{i} + B_{i}') - x_{i,j} \\
A_{i}' = \min(x_{i,j}, B_{i}') \\
B_{i}' = \max(x_{i,j}) \\
go_{i,j} = \text{rand}(A_{i}', B_{i}'), \text{ if } go_{i,j} < LB_{i} \parallel go_{i,j} > UB_{i}
\] (7)

where \( go_{i,j} \) is the generalized opposition-based value of \( x_{i,j} \), and \( k \) is the random generalized opposition-based coefficient, which is newly generated for each \( i \).

3.2. Elite Opposition-based Learning

Based on the idea of GOBL, in our previous work [36], we have extended the generalized opposition-based strategy, and proposed an elite opposition-based learning (EOBL) strategy. In EOBL, \( EN \) elite solutions are firstly selected from the current population, and then the beneficial information is extracted from these selected elite
solutions. The elite opposition-based solutions can be obtained by taking advantage of the extracted beneficial information. By using this manner, EOBL can further enhance the search ability of GOBL.

For solution \( X_{i,j} \), its elite opposition-based solution \( EO_{i,j} = [eo_{i,1}^{j}, eo_{i,2}^{j}, \ldots, eo_{i,m}^{j}] \) is defined by [36]:

\[
eo_{i,j}^{k} = k \cdot (A_{i}^{j} + B_{j}^{i}) - x_{i,j}^{k},
\]

\[
EO_{i,j}^{j} = \min(eo_{m}^{j}), EB_{j}^{i} = \max(eo_{m}^{j})
\]

\[
\text{if } eo_{i,j}^{k} < LB_{i} \| eo_{i,j}^{k} > UB_{i}\quad(8)
\]

\[i = 1,2, \ldots; SN; j = 1,2, \ldots; D; \]

\[m = 1,2 \ldots EN; k = \text{rand}(0,1)
\]

where \( EX_{i,j}^{m} = [ex_{m,1}^{j}, ex_{m,2}^{j}, \ldots, ex_{m,D}^{j}] \), \( m = 1, 2, \ldots, EN \) are the selected elite solutions used to extract the beneficial information, \( eo_{i,j}^{j} \) is the elite opposition-based value of \( X_{i,j} \), \( EN \) is the size of the selected elite solutions, which is set to \( SN \times 0.1 \), as recommended in our previous work [36] and \( EA_{i}^{j} \) and \( EB_{j}^{i} \) are the minimum and maximum values of the \( j \)th dimension of the selected elite solutions, respectively.

### Table 1. The 15 benchmark test functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Initial Range</th>
<th>Initial Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere Problem</td>
<td>[-100, 100]^D</td>
<td>0</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.22</td>
<td>[-10, 10]^D</td>
<td>0</td>
</tr>
<tr>
<td>Schwefel’s Problem 1.2</td>
<td>[-100, 100]^D</td>
<td>0</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.21</td>
<td>[-100, 100]^D</td>
<td>0</td>
</tr>
<tr>
<td>Rosenbrock’s Function</td>
<td>[-30, 30]^D</td>
<td>0</td>
</tr>
<tr>
<td>Step Function</td>
<td>[-100, 100]^D</td>
<td>0</td>
</tr>
<tr>
<td>Quartic Function with Noise</td>
<td>[-1.28, 1.28]^D</td>
<td>0</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.26</td>
<td>[-500, 500]^D</td>
<td>0</td>
</tr>
<tr>
<td>Ackley’s Function</td>
<td>[-32, 32]^D</td>
<td>0</td>
</tr>
<tr>
<td>Griewank Function</td>
<td>[-600, 600]^D</td>
<td>0</td>
</tr>
<tr>
<td>Penalized Function 1</td>
<td>[-50, 50]^D</td>
<td>0</td>
</tr>
<tr>
<td>Penalized Function 2</td>
<td>[-50, 50]^D</td>
<td>0</td>
</tr>
<tr>
<td>Noncontinuous Rastrigin’s Function</td>
<td>[-5.12, 5.12]^D</td>
<td>0</td>
</tr>
<tr>
<td>Alpine Function</td>
<td>[-10, 10]^D</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 2. Experimental results of ABC, OABC, GOABC, dABC and EOABC over 30 independent runs for the 15 benchmark test functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>ABC</th>
<th>OABC</th>
<th>GOABC</th>
<th>dABC</th>
<th>EOABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere Problem</td>
<td>0.10E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.22</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Schwefel’s Problem 1.2</td>
<td>0.10E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.21</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Rosenbrock’s Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Step Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Quartic Function with Noise</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Schwefel’s Problem 2.26</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Ackley’s Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Griewank Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Penalized Function 1</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Penalized Function 2</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Noncontinuous Rastrigin’s Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
<tr>
<td>Alpine Function</td>
<td>0.50E+04±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
<td>0.00E+00±0.00E+00</td>
</tr>
</tbody>
</table>
Algorithm 1. The algorithmic description of EOABC

1: \[ t = 0; \]
2: \[ FEs = 0; \]
3: Initialize the population;
4: while \( FEs < \text{Max}_FEs \) do
5: \[ Pr = \text{rand}(0, 1); \]
6: \[ \text{if } Pr < Pe \text{ then} \]
7: Choose \( EN \) elite solutions from the current population;
8: Calculate the lower and upper boundaries of the chosen elite solutions;
9: \[ \text{EO} = \{ \}; \]
10: \[ \text{for } i = 1 \text{ to } SN \text{ do} \]
11: \[ k = \text{rand}(0, 1); \]
12: Create the elite opposition-based solution \( EO^i \) for the \( i \)th solution \( X^i \);
13: Evaluate solution \( EO^i \);
14: \[ \text{EO} = \text{EO} \cup \{ EO^i \}; \]
15: \[ \text{FEs} = \text{FEs} + 1; \]
16: \[ \text{end for} \]
17: Choose the top best SN solutions from \( \{ P, EO \} \) for the next generation population;
18: Execute the computation procedure of the traditional ABC;
19: \[ t = t + 1; \]
20: \[ \text{end while} \]

3.3. EOABC

The basic ABC is good at exploration but poor at exploitation, which often results in slow convergence when solving complicated practical problems [12]. Aiming at this weakness of the basic ABC, the EOBL strategy is utilized to improve the exploitation ability. In the search process, the distribution information of the elite solutions in the current population is used to create the opposition-based solution of each solution. Therefore, the EOBL strategy can guide the search towards the promising area and thus improve the exploitation ability.

Like EOBL embedded in DE [36], EOABC has the similar framework. EOABC starts with a random initial population. After initialization, it executes a loop of search process. In the loop of search process, a random real number \( Pr \) is generated. If \( Pr \) is less than the EOBL probability \( Pe \), EOABC performs the steps of EOBL; Otherwise, it executes the computation procedure of the traditional ABC. In the steps of EOBL, \( EN \) elite solutions are firstly chosen from the current population. Then, the lower and upper boundaries of these chosen elite solutions are calculated. Subsequently, the elite opposition-based solution of each solution in the current population is created to constitute an elite opposition-based population. Finally, the elite opposition-based population is competed with the current population to choose the top best SN solutions for the next generation. The algorithmic description of EOABC is shown in Algorithm 1, where \( FEs \) is the number of fitness evaluations, \( \text{Max}_FEs \) is the maximum number of evaluations, \( Pe \) denotes the EOBL probability, \( P(t) = \{ X^i \} \) is the current population, and \( EOOP(t) = \{ EO^i \} \) is the elite opposition-based population.

4. NUMERICAL EXPERIMENTS

4.1. Experimental Settings

In order to evaluate the effectiveness of the proposed EOABC, 15 classical benchmark test functions widely used in the evolutionary computation community are employed in the experiments [41, 42], which are described in Table 1. The dimension of these 15 benchmark test functions are set to \( D = 30 \). In the experiments, the proposed EOABC is compared with ABC [9], OABC [20], GOABC [25], and dABC [35]. For a fair comparison, the parameter settings related to ABC are set as the same as reported values [25], and the learning probabilities of OBL, GOBL and EOBL are set to 0.3, as recommended [25]. Due to the stochastic characteristics of EAs, 30 independent runs for each algorithm and each test function are executed with 300,000 function evaluations (FEs) as the stopping criterion. Moreover, the average and standard deviation of the function error values are recorded for measuring the performance of the ABC algorithms. In order to obtain statistically sound conclusions, two-tailed \( t \)-test at a significance level of 0.05 is done on the experimental results [39].

4.2. Results and Discussions

The mean and standard deviation of the function error values achieved by each algorithm for f1-f15 are presented in Table 2. For convenient analysis, the best results among the algorithms are shown in bold. The summary of the comparison results are shown in the last three rows of Table 2. "Mean" and "SD" indicate the mean and standard deviation of the function error values obtained by 30 independent runs, respectively.

TABLE 3. Average rankings of the five ABC algorithms for the 15 benchmark test functions obtained by the Friedman test

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOABC</td>
<td>1.70</td>
</tr>
<tr>
<td>dABC</td>
<td>3.00</td>
</tr>
<tr>
<td>GOABC</td>
<td>3.17</td>
</tr>
<tr>
<td>OABC</td>
<td>3.50</td>
</tr>
<tr>
<td>ABC</td>
<td>3.63</td>
</tr>
</tbody>
</table>
The symbols ‘+’, ‘−’, and ‘≈’ denote that EOABC obtains better, worse, and similar results than the corresponding algorithms in terms of the two-tailed t-test, respectively.

From Table 2, it can be known that EOABC exhibits better performance than all the other four ABC algorithms on the majority of the 15 benchmark test functions. Specifically, EOABC is significantly better than ABC, OABC, GOABC, and dABC on test functions f1, f2, f3, f4, f5, f7, f8, f10, and f15 according to the two-tailed t-test. In contrast, ABC, OABC, GOABC, and dABC can not outperform EOABC on any test function. In addition, on test functions f6, f9, f12, f13, f14, ABC, OABC, GOABC, dABC, and EOABC all exhibit the similar performance. On test functions f11, both EOABC and dABC yield similar results, while they are significantly better than ABC, OABC, and GOABC on this test function. Overall, EOABC performs better than ABC, OABC, GOABC, and dABC on 10, 10, 10, and 9 out of 15 test functions, respectively.

In order to compare the total performance of the five ABC algorithms on the all 15 benchmark test functions, the average ranking of Friedman test is performed on the experimental results following the suggestions in [39, 43]. Table 3 presents the average ranking of the five ABC algorithms on the all 15 benchmark test functions. These five ABC algorithms can be sorted by the average ranking into the following order: EOABC, dABC, GOABC, OABC, and ABC. Therefore, EOABC achieves the best average ranking, which indicates that the total performance of EOABC is better than that of the other four ABC algorithms on the all 15 benchmark test functions. This can be because the EOBL strategy can significantly enhance the exploitation capacity of the basic ABC, and EOBL is more efficient than OBL and GOBL for improving the performance of the basic ABC.

5. CONCLUSIONS
EOBL is an effective strategy to enhance the performance of EAs. In this study, we employ the EOBL strategy to promote the performance of the traditional ABC, and thus propose a modified ABC, called EOABC. In the experiments, EOABC is compared with ABC, OABC, GOABC, and dABC on 15 benchmark test functions. The experimental results show that EOABC can significantly surpass ABC, OABC, GOABC, and dABC on the majority of the benchmark test functions. The comparison results also reveal that EOBL is more efficient than OBL and GOBL for promoting the search ability of the traditional ABC. In the future, we will apply EOABC to other complex optimization problems, such as multi-objective and dynamic optimization problems.

6. ACKNOWLEDGMENTS
This work was supported in part by the National Natural Science Foundation of China (Nos. 61402481, 61462036, and 41261093), by Natural Science Foundation of Jiangxi, China (Nos. 20151BAB217010, 20151BAB201015) and by the Natural Science Youth Foundation of Hebei Educational Committee (No. QN20131053).

7. REFERENCES


Elite Opposition-based Artificial Bee Colony Algorithm for Global Optimization

Z. Guo¹, S. Wang², X. Yue³, D. Jiang³, K. Li⁴

¹School of Science, JiangXi University of Science and Technology, Ganzhou, China
²School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang, China
³State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan, China
⁴School of Mathematics and Informatics, South China Agricultural University, Guangzhou, China

Abstract: Many problems in engineering and science can be transformed into optimization problems. The Artificial Bee Colony (ABC) algorithm is a recently developed stochastic optimization algorithm that has been successfully applied in various fields. However, due to the weakness of the ABC algorithm, in this paper, we propose an advanced ABC algorithm based on opposition learning (EOABC). In EOABC, the opposition learning strategy is combined with the existing ABC algorithm to improve the algorithm's performance. The algorithm is then tested on a set of benchmark functions and compared with other ABC algorithms. The results show that EOABC can obtain competitive results on most test functions.

Keywords: Evolutionary Algorithm, Artificial Bee Colony, Opposition-Based Learning, Elite Strategy

doi: 10.5829/idosi.ije.2015.28.09c.03