Journal of Engineering and Applied Sciences 9 (10-12): 405-413, 2014
ISSN: 1816-949X

An Efficient Artificial Bee Colony Algorithm for Constrained Optimization Problems

Soudeh Babaeizadeh and Rohanin Ahmad
Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, UTM 81310, Johor, Malaysia

Abstract: Artificial Bee Colony (ABC) algorithm is a relatively new swarm intelligence algorithm which has shown a competitive performance with respect to other population-based algorithms. However, there are still some insufficiencies in ABC algorithm such as slow convergence and easily trapped in local optima. These drawbacks can be even more challenging when constraints are also involved. To address these issues an Efficient Constrained Artificial Bee Colony (eABC) algorithm is proposed where two new solution search equations are introduced respectively to employed bee and onlooker bee phases. In addition, smart flight operator is employed to be used in scout bee phase. This algorithm is tested on several constrained benchmark problems. The numerical results demonstrate that the eABC is competitive with other state of the art constrained ABC algorithms under consideration.

Key words: Artificial bee colony, swarm intelligence, constrained optimization, chaotic search, smart hight

INTRODUCTION

Optimization methods are frequently applied to address various real world problems. However, finding optimal solutions for these problems are very challenging using traditional optimization methods. Because there have always been many real world problems where derivatives are unavailable or unreliable. Swarm Intelligence (SI) algorithms have shown considerable success in solving nonconvex, discontinuous, non-differentiable optimization problems and attracted more attention in recent years.

The most prominent Evolutionary Algorithms (EAs) have been suggested in the literatures are Genetic Algorithm (GA) (Holland, 1975), Particle Swarm Optimization (PSO) (Kennedy, 2011), Ant Colony Optimization (ACO) (Dorigo and Blum, 2005), Differential Evaluation (DE) (Storn and Price, 1997) and Artificial Bee Colony (ABC) algorithm (Karaboga, 2005) and so on.

ABC is a recently proposed SI algorithm which simulates the foraging behavior of honey bee swarms (Karaboga, 2005). Numerical performance demonstrate that ABC algorithm is competitive to other population-based algorithms such as GA, PSO, DE with an advantage of employing fewer control parameters and the need for fewer function evaluations to arrive at an optimal solution (Karaboga and Basturk, 2007a, b, 2008; Karaboga and Akay, 2009). Due to its simplicity and ease of implementation, ABC has captured much attention from researchers and it has been applied to solve many numerical as well as practical optimization problems (Gao *et al*., 2014; Aydin *et al*., 2014; Xiang and An, 2013; Li *et al*., 2012), since its invention.

Among optimization problems, the ones tackled in this paper are Constrained Optimization Problems (COPs) for Nonlinear Programming (NLP) which can be formulated as in the following problem:

$$\begin{aligned} &\min f(x) \\ &\text{s.t } g_j(x) \le 0,\ j = 1,2,...,m \\ &\quad h_j(x) = 0,\ j = m+1,...,l \end{aligned} \tag{1}$$

Where:

$x = [x_1, x_2,..., x_D] \in R^n$ = D-dimensional decision vector and each R^n = Bounded in the interval $[x_{min}, x_{max}]$
x_{min} = The lower bound
x_{max} = The upper bound
$f(x)$ = The objective function defined in D-dimensional search space in R^n

In general, most of the optimization problems have been primarily designed to address unconstrained optimization problems. In order to solve constrained problem, constraint handling techniques are employed to direct the search towards the feasible regions of the search space. Constraint handling methods are categorized into four groups by Koziel and Michalewicz

Corresponding Author: Rohanin Ahmad, Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, UTM 81310, Johor, Malaysia

(1999): methods based on penalty functions which penalize constraints to solve a constrained problem as an unconstrained problem, methods based on reservation of feasible solutions by transforming infeasible solutions to feasible solutions with some operators, methods that separate feasible and infeasible solutions, other hybrid methods.

However, ABC algorithm similar other evolutionary algorithms faces up with some challenging problems related with the solution search equation of ABC. This method has good exploration but poor exploitation (Karaboga, 2005) which results in the poor convergence. In this sudy, a new constrained ABC algorithm is proposed by employing two new solution search mechanisms for employed bee and onlooker bee phases respectively. Furthermore, smart flight operator is employed into scout bee phase instead of purely random generation of solutions in general ABC.

MATERIALS AND METHODS

Artificial bee colony: ABC algorithm is a recently introduced population-based method by Karaboga (2005) which simulate the foraging behavior of honey bee colonies. In ABC, the colony of artificial bees is classified into three groups, employed bees, onlooker bees and scout bees. Half of the colony includes employed bees and the other half consists of onlooker bees. Employed bees search around the food source and gathering required information. Then, they carry the information about the position of food source back to the hive and share this information with onlooker bees.

Onlooker bees choose food source with better quality from those found by employed bees using probability selection mechanism as a proportional of the quality of food source. Therefore, the food sources with good quality attract more onlooker bees compared to food source with lower quality. If the quality of the food source is not improved through a predetermined number of iteration, the food source will be abandoned by its employed bee and employed bee becomes a scout and starts to search for a new food source randomly in the neighborhood of the hive. Through, the search process, scout bees are responsible for exploration while exploitation is done using employed and onlooker bees.

In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of each food source corresponds to their fitness of the related solution. The number of employed bees or the onlooker bees is equal to the number of solutions SN in the population. At initialization step, a population of SN solutions are randomly generated using the following equation:

$$x_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \quad (2)$$

where, $x_i = (x_{i1}, x_{i2},..., x_{iD})$ and i = 1, 2,..., SN, j = 1, 2,..., D and D is a number of optimization parameters, $x_{min,j}$ and $x_{max,j}$ are the lower and upper bounds for the dimension j, respectively.

After initialization, the population of solution is evaluated and then is repeated in a cycle of the employed bees, onlookers and scouts. Each employed bee generates a new food source in their neighborhood using Eq. 3:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) \quad (3)$$

where, k0{1, 2, ..., SN} and j0{1, 2, ...D} are randomly chosen indexes, k has to be different from i, N_{ij} is a random number in the range [-1, 1]. Once $v_i = \{v_{i1}, v_{i2}, ..., v_{iD}\}$ is obtained, it will be evaluated and compared with x_i using greedy selection mechanism. If the fitness of v_i is better than fitness value of x_i, the v_i will be replaced with x_i and x_i will be removed, otherwise x_i is retained in population.

After all employed bees complete their searches, they share their information about fitness and position of solutions with the onlooker bees. An onlooker bee chooses a solution using probability value associate with the solution where p_i is defined as follows:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_i} \quad (4)$$

where, fit_i is the fitness value of solution i. Obviously, the higher the value of fit_i has more probability that the ith solution is selected. Then as in the case of employed bee a new solution is generated using Eq. 3. If a new solution has better quality than the old solution, the old solution is replaced with new solution otherwise the old solution remained in the population.

If a solution cannot be improved further through a predetermined number of trials (limit) the solution is assumed to be abandoned and the corresponding employed bee becomes a scout. The scout produces a solution randomly using Eq. 2. The detailed pseudo code of original ABC algorithm is presented in the Algorithm 1.

Algorithm 1; Original artificial bee colony algorithm:

Initialize the population of solution
Evaluate the initial population
cycle = 1
Repeat
Employed bee phase
Apply greedy selection process
Calculate the probability values
Onlooker bee phase
Scout bee phase
Memorize the best solution achieved so far
cycle = cycle+1
until cycle = maximum cycle number

Previous work on constrained artificial bee colony: ABC algorithm has been originally introduced to address unconstrained optimization problems (Karaboga, 2005). Then, this method is adapted to deal with constrained optimization problems. The presence of various constraints and interferences between them makes COPs more challenging than unconstrained optimization problems. In this study, we briefly present the available constrained ABC algorithms in the literature.

The first attempt to apply ABC algorithm to solve COPs is done by Karaboga and Bastruck (Karaboga and Basturk, 2007a, b). To cope with constraints, Deb (2000)'s mechanism is employed to be used instead of the greedy selection mechanism due to its simplicity, computational cost and fine tuning requirement over other constraint handling methods. Because initialization with feasible solutions is very time consuming and in some situation, impossible to generate a feasible solution randomly, the constrained ABC algorithm does not consider the initial population to be feasible. As alternative Deb's rules are employed to direct the solutions to feasible region of search space. In addition, the scout bee phase of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population. Scouts are generated at a predetermined period of cycles for discovering new solution randomly. This period is another control parameter called Scout Production Period (SPP). At each SPP cycle, it is controlled if there is an abandoned solution or not. If there is a scout production process is executed. The numerical performance of the proposed ABC algorithm is evaluated and compared with constrained PSO and DE algorithms and results show that ABC algorithm can be effectively applied for solving constrained optimization problems.

Mezura-Montes *et al*. (2010) presented Smart Flight ABC (SF-ABC) algorithm to improve the performance of constrained ABC where smart flight operator is applied in scout bee phase to direct search towards promising region of the search space. Therefore if the best solution is infeasible, the trial solution has the chance to be located near the boundaries of the feasible region of search space. However if the best solution is infeasible, the smart flight will generate a solution in promising region of search space. In addition, the combinations of two dynamic tolerances are also applied into SF-ABC to transform the original COP into unconstrained optimization. The numerical results demonstrate the competitive performance of SF-ABC with constrained ABC (Karaboga and Basturk, 2007a, b).

A modified ABC was introduced by Karaboga and Akay (2011) to solve COPs. In this algorithm, a new probability selection mechanism is presented to enhance diversity by allowing infeasible solutions in the population where infeasible solutions are introduced inversely proportional to their constraint violations and feasible solution defined based on their fitness values. To recognize this algorithm through this paper the abbreviation MABC will be used.

Another modified constrained ABC was developed by Subotic (2011) where Multiple Onlooker Bees (MO-ABC) are applied into original constrained ABC. In this algorithm three trial solutions are applied to form a new solution. The numerical performance of the algorithm when compared with the original ABC shows comparative results.

Mezura-Montes and Cetina-Dominguez presented a Modified ABC (M-ABC). This algorithm consist of four modifications on the selection mechanism, the equality and boundary constraints and scout bee operators compared to the original constrained ABC. The mechanisms to handle equality and boundary constraints are enhanced with the aim to support a more appropriate approach to the feasible region of the search space. A binary tournament selection based on feasibility is supplanted with the fitness selection of solutions applied in the original ABC. In addition, smart flight operator is employed to be used in scout bee instead of the uniformly random approach in constrained ABC (Karaboga and Basturk, 2007a, b). The numerical results show that M-ABC provides comparable results with respect to algorithms under comparison (Mezura-Montes and Cetina-Dominguez, 2012).

A Genetic Inspired ABC algorithm (GI-ABC) was introduced to adopt GA in the process of replacement of exhausted solutions (Bacanin and Tuba, 2012). In this algorithm, uniform crossover and mutation operators from GAs are applied to improve the performance of ABC algorithm.

Stanarevic, Tuba and Bacanin (Stanarevic *et al*., 2011) suggested Smart Bee ABC algorithm (SB-ABC) to solve constrained problems. In this algorithm, smart bee is used to memorize the solutions and their fitness. Then, the best solution is replaced with a new random solution if the new solution is unfeasible or if the new solution is feasible but it does not have better fitness. The numerical experiments show efficiency of the method.

ABC-BA is a hybrid algorithm presented by Tsai (2014) that integrates ABC and Bee Algorithm (BA) to solve COPs. In this algorithm, individuals can perform as an ABC individual in ABC sub-swarm or a BA individual in the BA sub-swarm. In addition, the population size of the ABC and BA sub-swarms change stochastically based on current best fitness values

achieved by the sub-swarms. Experimental results demonstrate that ABC-BA outperforms ABC and BA algorithms, respectively.

Constrained ABC algorithm was also applied to solve many real-world engineering problems in recent years. Brajevic *et al*. (2011) proposed a Constrained Artificial Bee Colony (SC-ABC) and applied on several standard engineering benchmark problems of discrete and continuous variables. The numerical results then were compared to results obtained from Simple Constrained Particle Swarm Optimization algorithm (SiC-PSO) which show a very good performance. Akay and Karaboga (2012) used ABC to solve large scale optimization problems as well as engineering design problems. The numerical results show that the performance of ABC algorithm is comparable to those of state of the art algorithms under comparison. Upgraded Artificial Bee Colony (UABC) algorithm was also introduced for constrained optimization problems by Brajevic and Tuba (2013) to improve fine-tuning features of the modification rate parameter and applying modified scout bee phase of the ABC algorithm. This algorithm was then tested on several engineering benchmark problems and the performance was compared with the performance of the Akay and Karaboga (2012) algorithm. The numerical results show that the UABC produces better results.

Efficient constrained artificial bee colony: In this study, we proposed an efficient constrained Artificial Bee Colony (eABC) algorithm. In the first step of this algorithm, initial population of SN solutions is generated randomly using Eq. 2. After initialization, the population is evaluated and a cycle of the search procedures of the employed bees, the onlooker bees and scout bees is repeated.

A new solution search equation is proposed for employed bee phase using Eq. 5:

$$v_{ij} = \begin{cases} v_{ij} = x_{rj} + \gamma_{ij}(x_{ij} - x_{rj}) & \text{if } R_j < MR \\ x_{ij} & \text{otherwise} \end{cases} \quad (5)$$

Where:

r = Randomly chosen index has to be different form

i and $(_{i,j}$ = A random number between [-1, 1]

R_j = Uniformly distributed random number in the range [-1, 1]

MR = A control parameter which controls the number of parameters to be modified

Equation 5 it is obvious that when the difference between the parameters x_{ij} and x_{rj} decreases, the perturbation on the position x_{ij} decreases. Therefore as the search moves toward the optimum solution in the search space, the step size is adaptively reduced.

After generating a new solution using Eq. 5 Deb's ruels are applied in selection process to direct the individuals to feasible region of search space. Using Deb's mechanism, either the new solution is memorized and the current solution is removed or the current solution is remand.

Algorithm 2; Employed bee phase for eABC:

for I = 1: SN
for j = 1: D
Produce a new solution u_i using Eq. 5
end for
If no parameter is changed, choose a parameter randomly and change it form solution x_i using Eq. 5.
Evaluate the solution v_i
Apply the selection process between v_i and x_i based on Deb's Method
If solution x_i does not improve $trial_i = trail_i+1$, otherwise $trail_i = 0$
end if

Deb's Method utilizes a tournament selection mechanism where two solutions are compared using following rules:

- Feasible solution is preferred to infeasible solution
- Among two feasible solutions, the one having better objective function value is preferred
- Among one feasible and one infeasible solution, the one having smaller constant violation is preferred

The framework of employed bee phase is given in Algorithm 2. After all employed bees complete their searches, they share their information related to the fitness values and the positions of their solutions with the onlooker bees. An onlooker bee chooses a solution using a probability mechanism as follows:

$$p_i = \begin{cases} 0.5 + \left(\frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}\right) \times 0.5 & \text{if solution is feasible} \\ \left(1 - \frac{violation_i}{\sum_{j=1}^{SN} violation_j}\right) \times 0.5 & \text{if solution is infeasible} \end{cases} \quad (6)$$

where, $violation_i$ is the constraint violation of solution x_i and $finess_i$ is the fitness value of the solution x_i. Based on this equation infeasible solutions are allowed to consider in population as well as feasible solutions. The fitness value also is defined in Eq. 7:

$$\text{fitness}_i = \begin{cases} 1/(1+f_i) & \text{if } f_i \geq 0 \\ 1+\text{abs}(f_i) & \text{if } f_i < 0 \end{cases} \quad (7)$$

where, f_i is the objective function of solution i. An onlooker bee evaluates the information shared by employed bees and selects a solution with a probability associated with its nectar amount. After solution selection, onlooker bees produce modification on the position of the selected solution by taking advantage of the global best and random solution to guide the candidate solution toward promising region of search space using Eq. 8:

$$v_j = \begin{cases} x_{ij} + \phi_{ij}(x_{best,j} - x_{r1,j}) + \kappa_{i,j} r \text{ and}(x_{r1,j} - x_{r2,j}) & \text{if } R_j < MR \\ x_{ij} & \text{otherwise} \end{cases} \quad (8)$$

where, $x_{r1,\,j}$ and $x_{r2,\,j}$ are uniformly random solution and $j = \{1, 2, ..., D\}$, $N_{i,\,j}$ and $k_{i,\,j}$ are random number in the range [-1, 1] and [0.3, 0.6], respectively. Similar with employed bee phase after generating new solution using Eq. 8, the new solution is compared with current solution using Deb's rules. If the new solution has better equality it will remained in the population and the current solution removed otherwise the current solution is remained. The framework of onlooker bee phase is given in Algorithm 3.

After distribution of all onlooker bees if a solution can not improve further through predetermined number of cycles (limit) it is abandoned and replaced with a new solution discovered by scout bees. The scout produces a new solution using smart flight operator defined in Eq. 9:

$$v_j = x_{ij} + j_{ij}(x_{rj} - x_{ij}) + (1 - j_{ij})(x_{best,\,j} - x_{ij}) \quad (9)$$

where, N_{ij} is random number in [-1,1] and r is random index that have to be different from i, x_{best} is the best solution found so far.

Algorithm 3; Onlooker bee phase for eABC:
t = 0, i = 1
Repeat
if random<p_i then
t = t+1
for j = 1: D
Produce a new solution v_i using Eq. 8
end for
If no parameter is changed, choose a parameter randomly and change it form solution x_i using Eq. 8.
Evaluate the solution v_i
Apply the selection process between v_i and x_i based on Deb's Method
If solution x_i does not improve $trail_i = trail_i+1$, otherwise $trail_i = 0$
end if
i = i+1
i = i mod (SN+1)
until t = SN

Numerical experiments and comparisons: In order to evaluate the performance of ABC algorithm and show the efficiency and superiority of the proposed algorithm, 24 well-known benchmark problems form CEC2006 (Liang *et al*., 2006) are applied.

The proposed algorithm is evaluated and compared with five state of the art constrained ABC algorithms. The eABC algorithms as well as other algorithms in comparison are coded in MALAB environment. Each problem runs 30 times and statistical results are provided including the best, median, mean, worst results and the standard deviation which can be seen in Table 1 and 2.

Table 1: The Numerical results obtained by ABC, MABC, M-ABC, MO-ABC and eABC

Problems	Parameters	ABC	MABC	M-ABC	SF-ABC	MO-ABC	eABC
g01	Best	-1.5000000	-1.5000000	-15.000000	-15.0000000	-15.000000	-1.5.00000
	Mean	-1.5000000	-1.5000000	-15.000000	-14.163245	-15.000000	-1.500000
	Worst	-1.5000000	-1.5000000	-15.000000	-12.525128	-15.000000	-1.500000
	SD	0.0000000	0.0000000	0.000000	0.92321	0.000000	0.000000
g02	Best	0.8035669	0.8035383	0.803614	-0.708944	-0.803610	-0.8036150
	Mean	-0.7917445	-0.792927	-0.799450	-0.471249	-0.793510	-0.8021544
	Worst	-0.7529237	-0.750302	-0.778176	-0.319535	-0.74458	-0.7990689
	SD	0.013292	0.011051	-0.006440	0.010823	0.016310	0.001263
g03	Best	-1.004657	-1.004817	-1.000	-1.000	-1.000	-1.00409
	Mean	-1.000096	-1.001941	-1.000	-1.000	-1.000	-1.00313
	Worst	-0.979659	-0.989160	-1.000	-1.000	-1.000	-1.00104
	SD	0.00597911	0.0003752	0.000	0.000	0.000	0.001336
g04	Best	-30665.542	-3066.542	-30665.5	-30665.539	-30665.539	-30665.54
	Mean	-30665.542	-3066.542	-30665.539	-30665.539	-30665.539	-30665.54
	Worst	-30665.542	-3066.542	-30665.539	-30665.539	-30665.539	-30665.54
	SD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g05	Best	5126.489	5127.099	5126.734	5126.506	5126.657	5126.394
	Mean	5177.239	5263.991	5178.178	5126.527	5162.506	5299.670
	Worst	5307.988	5802.318	5317.183	5126.859	5229.119	5968.589
	SD	57.86021	156.0343	56.0001	0.079343	47.8234	248.4226
g06	Best	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
	Mean	-6961.814	-6961.814	-6961.814	-6961.813	-6961.813	-6961.814

Table 1: Continue

Problems	Parameters	ABC	MABC	M-ABC	SF-ABC	MO-ABC	eABC
	Worst	-6961.814	-6961.814	-6961.814	-6961.805	-6961.804	-6961.814
	SD	0.0000000	0.0000000	0.000000	0.0002	0.0001	0.000000
g07	Best	24.46138	24.47032	24.3121	24.16452	24.32317	24.55848
	Mean	24.70718	24.68698	24.41643	24.65842	24.45625	24.80918
	Worst	25.16577	25.36005	24.794131	25.55104	24.92918	25.10102
	SD	0.1813943	0.1786124	0.127124	0.326125	0.135021	0.1286075
g08	Best	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.0958250
	Mean	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.0958250
	Worst	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.0958250
	SD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g09	Best	680.6381	680.6371	680.6331	680.6321	680.6312	680.6387
	Mean	680.6506	680.6515	680.6472	680.6450	680.6353	680.6512
	Worst	680.6757	680.6760	680.6763	680.8582	680.6362	680.6776
	SD	0.008074	0.009610	0.054320	0.041213	0.004123	885.1586
g10	Best	7160.631	7220.554	7051.775	7049.517	7053.32	7304.817
	Mean	7364.940	7347.843	7233.810	7116.824	7167.801	7445.860
	Worst	7691.303	7924.128	7604.129	7362.741	7418.334	7647.175
	SD	129.8405	134.1410	132.1284	82.12413	83.00823	87.75224
g11	Best	0.749000	0.749000	0.750000	0.750000	0.750000	0.7490001
	Mean	0.749002	0.749003	0.750000	0.750000	0.750000	0.7490035
	Worst	0.749010	0.749014	0.750000	0.750000	0.750000	0.7490247
	SD	0.000002	0.000003	0.000000	0.000000	0.000000	0.0000005
g12	Best	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Mean	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Worst	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	SD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g13	Best	0.555124	0.489597	0.05389	0.053986	0.454213	0.491074
	Mean	0.949781	0.957689	0.157791	0.263854	0.456375	0.943945
	Worst	1.492954	1.437534	0.441978	1.000000	0.489036	1.325281
	SD	0.146915	0.161358	0.017232	0.216234	0.021103	0.171501

Table 2: The Numerical results obtained by ABC, MABC, M-ABC, MO-ABC and eABC

Problems	Parameters	ABC	MABC	M-ABC	SF-ABC	MO-ABC	eABC
g14	Best	-45.11878	-45.32082	-47.64541	-46.6651370	-46.450835	-45.97169
	Mean	-42.68215	-42.65421	-47.27156	-46.468243	-45.998013	-42.20681
	Worst	-40.60165	-40.05962	-46.53698	-43.87123	-45.316798	-39.11480
	SD	1.171236	1.195831	0.245761	0.520124	0.257124	1.461280
g15	Best	941.2191	951.4375	961.7152	961.7151	961.7151	940.1215
	Mean	958.8476	960.8922	961.7188	961.7155	961.8831	957.7468
	Worst	972.9578	970.6846	961.7912	961.7201	964.3398	970.6761
	SD	7.512742	4.87894	0.014319	0.1592	0.542672	7.754218
g16	Best	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155
	Mean	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155
	Worst	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155	-1.905155
	SD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g17	Best	8886.685	8879.576	8866.599	8927.598	8939.125	8871.616
	Mean	9053.597	9053.567	8987.459	8928.865	8946.134	9052.462
	Worst	9249.174	9215.365	9165.2543	8938.617	8956.127	9.259310
	SD	123.0898	122.6397	95.6532	3.1213	9.52825	1.320255
g18	Best	-0.840568	-0.859365	-0.866023	-0.866025	-0.865976	-0.842418
	Mean	-0.840573	-0.710702	-0.795019	-0.740748	-0.767198	-0.741887
	Worst	-0.0508290	0.0677663	0.093789	0.145231	0.096120	0.061626
	SD	0.0508290	0.0677663	0.093789	0.145231	0.096120	0.061626
g19	Best	36.77401	37.58086	33.2547	32.66271	33.76983	37.08395
	Mean	39.29784	39.83492	34.2656	33.10714	35.31478	39.70235
	Worst	42.70161	42.42735	35.7368	34.91401	37.36458	42.47243
	SD	1.457124	1.174349	0.63124	0.51321	0.687514	1.363563
g23	Best	-	-	-159.754	-350.126	-	-704.385
	Mean	-	-	-35.2847	-121.375	-	-221.191
	Worst	-	-	109.1275	276.0038	-	57.88116
	SD	-	-	82.76981	157.8952	-	196.7631
g24	Best	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013
	Mean	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013
	Worst	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013
	SD	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Benchmark test problems and parameter settings: The main characteristics of 24 benchmark functions are shown in Table 3. Table 3 describes various kinds of these test functions (linear, nonlinear, polynomial, quadratic and cubic) with different numbers of decision variables, different types (linear inequalities, linear equalities,

nonlinear inequalities and nonlinear equalities) and numbers of constraints. Table 1 D is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints, LE is the number of linear equality constraints, NE is the number of nonlinear equality constraints, a is the number of constraints active at the optimal solution and n is the number of variables of the problem. However as all the algorithms considered in comparison were not able to obtain feasible solutions for g20-22 we exclude these problems from our experiments. In addition, the value of each parameters used are given in Table 4.

Table 3: The main characteristics of the test problems

Function	Type	n	D	LI	NI	LE	NE	a
g01	Quadratic	13	0.0111	9	0	0	0	6
g02	Nonlinear	20	99.9971	1	1	0	0	1
g03	polynomial	10	0.0000	0	0	0	1	1
g04	Quadratic	5	52.1230	0	6	0	0	2
g05	Cubic	4	0.0000	2	2	0	3	3
g06	Cubic	2	0.0066	0	5	0	0	2
g07	Quadratic	10	0.0003	3	2	0	0	6
g08	Nonlinear	2	0.8560	0	4	0	0	0
g09	polynomial	7	0.5121	0	3	0	0	2
g10	linear	8	0.0010	3	0	0	0	3
g11	polynomial	2	0.0000	0	1	0	1	1
g12	Quadratic	3	4.7713	0	1	0	0	0
g13	Quadratic	5	0.0000	0	0	0	3	3
g14	Nonlinear	10	0.0000	0	0	3	0	3
g15	Quadratic	3	0.0000	0	0	1	1	2
g16	Nonlinear	5	0.0204	4	34	0	0	4
g17	Nonlinear	6	0.0000	0	0	0	4	4
g18	Quadratic	9	0.0000	0	13	0	0	6
g19	Nonlinear	15	33.4761	0	5	0	0	0
g20	linear	24	0.0000	0	6	2	12	16
g21	linear	7	0.0000	0	1	0	5	6
g22	linear	22	0.0000	0	1	8	11	19
g23	linear	9	0.0000	0	2	3	1	6
g24	linear	2	79.6556	0	2	0	0	2

Table 4: Parameters setting

Parameters	Symbols	Values
Solutions number	SN	20
Maximum cycle number	MCN	6000
Modification rate	MR	0.8
Population size	PS	40
Limit	Limit	150
Scout production period	SPP	150
Epsilon	g	0.001

RESULTS AND DISCUSSION

The numerical performance of eABC algorithm was compared with original ABC (Karaboga and Basturk, 2007a, b), MABC (Karaboga and Akay, 2011), M-ABC (Mezura-Montes and Cetina-Dominguez, 2012), SF-ABC (Mezura-Montes *et al*., 2010) and MO-ABC (Subotic, 2011) algorithms. From Table 1 and 2 it is obvious that the eABC algorithm in problems g02, g03, g04, g08, g11, g23 outperforms compare with other algorithms. For g05, g10, g15, g17, g19 the performance of SF-ABC was superior to all other algorithms. However, MO-ABC is superior in problems g09 and g14. In problem g01, g06, g12, g16 and g24 all the algorithm can find the optimal results. The numerical performance showed that eABC provided comparable result with respect to other state of the art algorithms in solving COPs. In order to compare the convergence ability of eABC with the other state of the art algorithms, three sample plots are presented in Fig. 1-3 which clearly show that eABC was able to converge faster than other algorithms which confirms that the new search equations can accelerate the constrained ABC convergence.

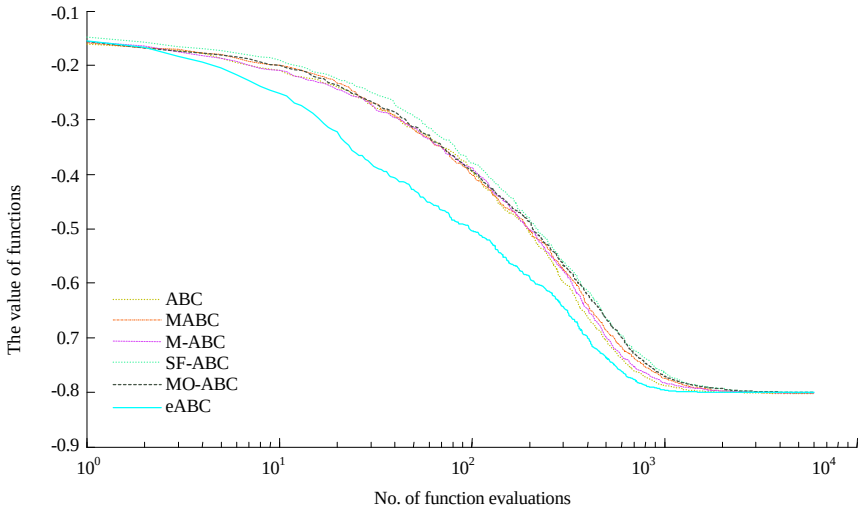

Fig. 1: Iterations to convergence for problem g02

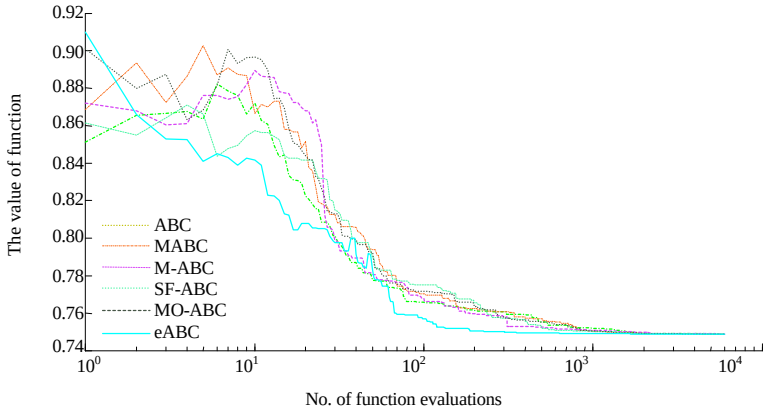

Fig. 2: Iterations to convergence for problem g11

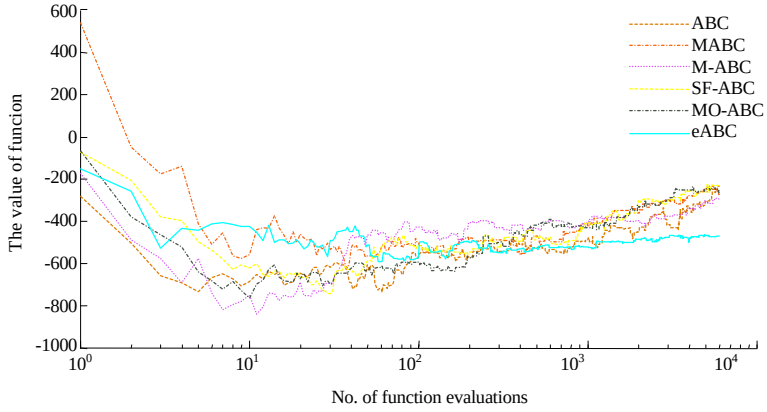

Fig. 3: Iterations to convergence for problem g23

CONCLUSION

In this study, we have introduced a modified constrained ABC called eABC algorithm to solve constrained optimization problems in which two new search equations proposed for employed bee and onlooker bee phases to enhance the global convergence of ABC algorithm to solve COPs. In addition, smart flight mechanism was applied to generate new solution in scout bee phase. The experimental results were tested on 24 benchmark functions and show that eABC is competitive with state of the art constrained ABC algorithms. The Future researches include testing other constraint handling mechanisms and using local search operators to improve ABC algorithm further.

ACKNOWLEDGEMENT

The researchers would like to thank Universiti Teknologi Malaysia for the financial funding through the grant RUG 08H47.

REFERENCES

Akay, B. and D. Karaboga, 2012. Artificial bee colony algorithm for large-scale problems and engineering design optimization. J. Intell. Manuf., 23: 1001-1014.

Aydin, D., S. Ozyon, C. Yasar and T. Liao, 2014. Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem. Int. J. Electr. Power Energy Syst., 54: 144-153.

Bacanin, N. and M. Tuba, 2012. Artificial Bee Colony (ABC) algorithm for constrained optimization improved with genetic operators. Stud. Inform. Control, 21: 137-146.

Brajevic, I. and M. Tuba, 2013. An upgraded Artificial Bee Colony (ABC) algorithm for constrained optimization problems. J. Intell. Manuf., 24: 729-740.

Brajevic, I., M. Tuba and M. Subotic, 2011. Performance of the improved artificial bee colony algorithm on standard engineering constrained problems. Int. J. Math. Comput. Simul., 5: 135-143.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Comput. Methods Applied Mech. Eng., 186: 311-338.

Dorigo, M. and C. Blum, 2005. Ant colony optimization theory: A survey. Theor. Comput. Sci., 344: 243-278.

Gao, W.F., S.Y. Liu and L.L. Huang, 2014. Enhancing artificial bee colony algorithm using more information-based search equations. Inform. Sci., 270: 112-133.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. University of Michigan Press, Ann Arbor, MI., USA., ISBN: 9780472084609, Pages: 183.

Karaboga, D. and B. Akay, 2009. A comparative study of artificial bee colony algorithm. Applied Math. Comput., 214: 108-132.

Karaboga, D. and B. Akay, 2011. A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. Applied Soft Comput., 11: 3021-3031.

Karaboga, D. and B. Basturk, 2007a. A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. J. Global Optim., 39: 459-471.

Karaboga, D. and B. Basturk, 2007b. Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problem. Proceedings of the 12th International Fuzzy Systems Association World Congress, June 18-21, 2007, Cancun, Mexico, pp: 789-798.

Karaboga, D. and B. Basturk, 2008. On the performance of Artificial Bee Colony (ABC) algorithm. Applied Soft Comput., 8: 687-697.

Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey, October 2005. http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf.

Kennedy, J., 2011. Particle Swarm Optimization. In: Encyclopedia of Machine Learning, Sammut, C. and G.I. Webb (Eds.). Springer Science and Business Media, New York, USA., ISBN: 9780387307688, pp: 760-766.

Koziel, S. and Z. Michalewicz, 1999. Evolutionary algorithms, homomorphous mappings and constrained parameter optimization. Evol. Comput., 7: 19-44.

Li, G., P. Niu and X. Xiao, 2012. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Applied Soft Comput., 12: 320-332.

Liang, J.J., T.P. Runarsson, E. Mezura-Montes, M. Clerc, N. Suganthan, C.A.C. Coello and K. Deb, 2006. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. J. Applied Mech., 41: 2-24.

Mezura-Montes, E. and O. Cetina-Dominguez, 2012. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. Applied Math. Comput., 218: 10943-10973.

Mezura-Montes, E., M. Damian-Araoz and O. Cetina-Domingez, 2010. Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. Proceedings of the IEEE Congress on Evolutionary Computation, July 18-23, 2010, Barcelona, pp: 1-8.

Stanarevic, N., M. Tuba and N. Bacanin, 2011. Modified artificial bee colony algorithm for constrained problems optimization. Int. J. Math. Models Methods Applied Sci., 5: 644-651.

Storn, R. and K. Price, 1997. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim., 11: 341-359.

Subotic, M., 2011. Artificial bee colony algorithm with multiple onlookers for constrained optimization problems. Proceeding of the European Computing Conference, April 28-30, 2011, Paris, France, pp: 251-256.

Tsai, H.C., 2014. Integrating the artificial bee colony and bees algorithm to face constrained optimization problems. Info. Sci., 258: 80-93.

Xiang, W.L. and M.Q. An, 2013. An efficient and robust artificial bee colony algorithm for numerical optimization. Comput. Operat. Res., 40: 1256-1265.