Research Journal of Applied Sciences, Engineering and Technology 10(5): 537-546, 2015
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: November 13, 2014 Accepted: January 27, 2015 Published: June 15, 2015

Performance Comparison of Constrained Artificial Bee Colony Algorithm

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

Abstract: This study is aimed to evaluate, analyze and compare the performances of available constrained Artificial Bee Colony (ABC) algorithms in the literature. In recent decades, many different variants of the ABC algorithms have been suggested to solve Constrained Optimization Problems (COPs). However, to the best of the authors' knowledge, there rarely are comparative studies on the numerical performance of those algorithms. This study is considering a set of well-known benchmark problems from test problems of Congress of Evolutionary Computation 2006 (CEC2006).

Keywords: Artificial bee colony, constrained optimization, nature inspired algorithms, swarm intelligence

INTRODUCTION

Population-based optimization algorithms are nature-inspired algorithms that can generate near-optimal solutions for hard optimization problems. A general characteristic of all population-based algorithms is that the population of possible solutions is improved iteratively by applying some search operators on the individuals depending on the quality of their fitness. As a result, the population is steered towards more promising region of the search space. Population-based optimization algorithms can be classified in two main categories, Evolutionary Computation (EC) and Swarm Intelligence (SI).

Genetic Algorithm (GA) (Tang *et al*., 1996), Evolution Strategy (ES) (Rechenberg, 1978), Genetic Programming (GP) (Koza, 1992), Differential Evolution (DE) (Storn and Kenneth, 1997) and Evolutionary Programming (EP) (Fogel *et al*., 1966) have been at the core of recent advancements on EC class.

Swarm intelligence has also captured attention of many researchers in recent years. The most prominent SI algorithms proposed in the literatures are as follow. Ant Colony Optimization (ACO) (Dorigo and Stützle, 2010), Particle Swarm Optimization (PSO) (Kennedy, 2010), Bee Swarm Optimization (BSO) (Drias *et al*., 2005), Bacterial Foraging Optimization (BFOA) (Passino, 2002), Biogeography-Based Optimization (BBO) (Simon, 2008), Artificial Immune Systems (AIS) (Farmer *et al*., 1986) and Artificial Bee Colony (ABC) (Karaboga, 2005) and so on.

ABC is one of the most successful SI algorithm inspired from the foraging behavior of honey bee swarm and was firstly suggested by Karaboga (2005) in 2005 to solve unconstrained optimization problems. Numerical studies show that this algorithm is competitive to other population-based algorithms with an advantage of simplicity and employing fewer control parameters (Karaboga and Basturk, 2007a, 2008; Karaboga and Akay, 2009). This method was then extended to solve Constrained Nonlinear Optimization Problems (CNOPs) which can be formulated as in Problem:

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

where, $x = [x_1, x_2, \cdots, x_n] \in R^n$ is n-dimensional decision vector and each R^n is bounded by lower and upper bounds as $x_{min} \le x_i \le x_{max}$. The objective function $f(x)$ is defined on s an n-dimensional search space in R^n.

This study presents a comprehensive comparative study on the performances of available constrained ABC algorithms.

Artificial bee colony: ABC is a recently proposed population-based algorithms by Karaboga (2005) being inspired by foraging behavior and waggle dance of honey bee colonies. This algorithm has been widely used to solve optimization problems (Gao *et al*., 2014; Aydina *et al*., 2014; Xiang and An, 2013; Li *et al*., 2012). ABC algorithm simulates the cooperative intelligence in foraging behavior of a honey bee swarm. In this algorithm, the position of food source denotes a possible solution to the optimization problem and the nectar amount of food source represents fitness value of the associated solution. In ABC algorithm, half of the

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

colony includes employed bee and the other half consist of onlooker bees. The number of employed bees is equal to the Number of Solutions (SN) in the population. At initialization step, ABC generates a randomly distributed initial population of SN solutions using following Equation:

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

where, each solution $x_i, i = 1,2,\cdots,SN$ is d-dimensional vector for $j = \{1,2,\cdots,d\}$, $x_{min,j}$ and $x_{max,j}$ are the lower and upper bounds for the dimension j, respectively. After initialization, the fitness value of associated solutions are evaluated and saved. Then, the population of the solutions is subjected to predetermined number of cycles called Maximum Cycle Number (MCN) of the search processes of the employed bees, the onlooker bees and the scout bees' phases.

An employed bee produces a modification on the solution x_i defined as:

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

where, $k \in \{1,2,\cdots,SN\}$ and $j \in \{1,2,\cdots,d\}$ are randomly chosen indexes, k has to be different from $i, \emptyset_{i,j}$ is a random number in the range $[-1,1]$. After each v_i is calculated the fitness value of this solution is evaluated and a greedy selection mechanism is applied comparing x_i and v_i.

If the fitness value of the new solution is higher than the current solution the bee is replaced with the new solution, otherwise the current solution remains. After the employed bee phase, the solution information is transferred to the onlooker bee phase. Onlooker bees choose a solution depending on the probability value p_i associated with that solution calculated using following equation:

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

where, fit_i is the fitness value of solution i. Once the onlooker has selected solution x_i a modification is done on the solution using Eq. (3). Then fitness values of generated solutions are evaluated and similar to employed bees phase, greedy selection mechanism is employed. If new solution has better fitness value than current solution, the new solution remains in population and the old solution is removed. In the scout bee phase, if solution x_i cannot be improved further through a predetermined number of trails (limit), then that solution is abandoned and replaced with a new solution generated randomly in scout phase. This process employs Eq. (2).

The ABC algorithm consists of three control parameters, the Number of Solutions (SN), the total Number of Cycles (MCN) as well as the number of cycles that a non-improving solution will be kept before being abandoned and replaced by a new solution generated by the scout bee mechanism (limit). In ABC algorithm the employed and onlooker bees are responsible for exploitation processes and scout bees control the exploitation process. In recent years the vast majority of unconstrained ABC algorithm have been presented and applied for practical problems (Akay and Karaboga, 2012; Gao *et al*., 2013; Banitalebi *et al*., 2015; Karaboga and Gorkemli, 2014; Kiran *et al*., 2015).

METHODOLOGY

Constrained artificial bee colony: ABC algorithm has been originally proposed to deal with unconstrained optimization problems (Karaboga, 2005). The ABC algorithm is then adapted to tackle constrained optimization problems. The presence of various constraints and interferences between constraints makes COPs more difficult to tackle than unconstrained optimization problems. In this section, several constrained ABC algorithm are concisely reviewed to facilitate experimental studies, analysis and comparison.

Original constrained artificial bee colony: ABC algorithm for the first time was adapted by Karaboga and Basturk (2007b) to solve constrained optimization problems. In this algorithm (Deb, 2000) mechanism is employed to cope with constraints, due to its simplicity, computational cost and fine tuning requirement over other constraint handling methods. Deb's method applies a tournament selection operator, comparing two solutions at a time with assumption that any feasible solution is preferable than any infeasible solution. Therefore, based on Deb's method the following criteria are always enforced:

- Any feasible solution is preferred to any infeasible solution
- Among two feasible solutions, the one having better fitness value is preferred
- If both solutions are infeasible the one having lowest constraint violation is preferred

As initialization with feasible solutions is very time consuming mechanism and in some situation, impossible to generate a feasible solution randomly, the constrained ABC algorithm does not consider the initial population to be feasible. Alternatively, Deb's rules were employed instead of greedy selection to direct the solutions to feasible region of search space. In addition, scout bee phase of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population.

In order to produce a candidate solution from the current solution in memory, the constrained ABC algorithm uses following equation:

$$v_{ij} = \begin{cases} x_{i,j} + \emptyset_{i,j}\left(x_{i,j} - x_{k,j}\right) & R_j < MR \\ x_{ij}, & Otherwise \end{cases} \quad (5)$$

where, $k \in \{1,2,\cdots,SN\}$ is randomly chosen index which has to be different from i, R_j is randomly chosen real number in the range $(0,1)$ and $j \in \{1,2,\cdots,d\}$. MR, is parameter that controls the probability of modification of the coefficient of the new solution.

Scout Production Period (SPP) is another parameter that controls the non-improving solution. If a solution cannot be improved further after certain trials new solution is generated randomly by scout bees. According to what is described above, the Original ABC algorithm (Karaboga and Basturk, 2007a) is explained as Algorithm 1.

Algorithm 1: Original Constrained ABC
Initialize the population of solution
Evaluate the initial population
cycle = 1
Repeat
Employed bee phase
Apply selection process based on Deb's method
Calculate the probability values for each solution
Onlooker bee phase
Scout bee phase
Memorize the best solution achieved so far
cycle = cycle+1
until cycle = maximum cycle number

Smart flight artificial bee colony algorithm: In scout bee phase, there is no guarantee a solution generated using uniform random search attracted towards a promising region of search space. Then, to overcome this difficulty, Flight ABC (SF-ABC) algorithm (Mezura-Montes *et al*., 2010) was introduced which employ smart flight operator in scout bee phase to direct the search towards the best-so-far solution. Based on this algorithm the new the features of this method of authentication can be listed as follows (Abhishek *et al*., 2013):

- Easy to implement
- Requires no special equipment
- Easy to lost or forget
- Vulnerable to shoulder surfing
- Security based on password strength
- Cost of support increases
- Familiar with a lot of users

solution search equation for scout bee is generated using following Eq. (6):

$$v_{i,j} = x_{i,j} + \emptyset_{i,j}\left(x_{i,j} - x_{k,j}\right) + \left(1 - \varphi_{ij}\right)\left(x_{b,j-}x_{ij}\right) \quad (6)$$

where, $k \in \{1,2,\cdots,SN\}$ and $j \in \{1,2,\cdots,d\}$, x_{kj} is a randomly chosen solution that has to be different form $x_{i,j}$ and x_{bj} where $x_{b,j}$ is the best-so-far solution. The $\varphi_{i,j}$ is a random number in $(-1,1)$.

If the best solution x_{bj} is infeasible, the trial solution has a chance to be located near the boundaries of the feasible region of search space. Even if the best solution is feasible, the smart flight will generate a solution in promising region of search space. In addition, two dynamic tolerances ε-constrained method and tolerance for equality constraints are applied into SF-ABC as constrained handling mechanism. In this algorithm instead of using Deb's rules applied in Deb (2000) ε-constrained method is employed to transform the original CNOP into unconstrained optimization problem and use the sum of constraint violation defined using following equation:

$$\Phi(x_k) = \sum_{i=1}^{m} \max\left(0, g_i(\boldsymbol{x})\right) + \sum_{j=m+1}^{l} \max\left(0, |h_i(x)| - \delta\right) \quad (7)$$

A tolerance ε is used for comparison between two solutions x_i and x_j based on their objective function values and their constraint violations. Based on this method x_i is preferred to x_j if and only if any of the following conditions are satisfied:

- If constraint violations of solutions x_i and x_j are less than the tolerance value and $f(i) < f(j)$.
- If constraint violations of both solutions are the same and $f(i) < f(j)$.
- If constraint violation of x_i is less than constraint violation of x_j.

This process allows some infeasible solutions located in the boundary of feasible region to be considered to improve the preservation of solutions placed in the promising areas in the boundaries of the feasible region. The ε value varies by:

$$\varepsilon(\mathrm{c}) = \begin{cases} \Phi(x_b)(1 - c/cn)^{cp}, & \text{if g} < CN \\ 0, & \text{Otherwise} \end{cases} \quad (8)$$

where, c is the cycle number and cn is the cycle number where, the ε value will become zero and cp is a parameter to control the speed of reducing the tolerance value. The initial value for ε corresponds to the sum of constrain violations of the best solution in the initial population. The other dynamic tolerance has been designed for equality constraints based on a tolerance value δ. In this mechanism equality constraints are transformed into inequality constraint and the value of δ reduces with increasing time until a value e^4 is reached where δ is defined by:

$$\delta(c+1) = \delta(c)/dec \quad (9)$$

Modification on artificial bee colony algorithm with multiple onlookers: Subotic (2011) proposed a

Modified Constrained ABC (MO-ABC) which applies multiple onlooker bees to improve original constrained ABC. In this algorithm after initialization by means of Eq. (2) and the corresponding function evaluation, the main loop contains employed bee, onlooker bee and scout bee phases. Solution search equations for employed bees are generated using Eq. (3) and evaluated. Deb's rules are then employed to choose better solution among current and trial solution. Probability for each solution is also calculated using Eq. (4) and onlooker bees generates new solution using equation below:

$$v_{i,j} = \begin{cases} x_{i,j} + a_1\varphi_{i,j}\left(x_{i,j} - x_{k,j}\right) \\ +a_2\varphi_{i,j}\left(x_{i,j} - x_{k-1,j}\right) \\ +a_3\varphi_{i,j}\left(x_{i,j} - x_{k-2,j}\right),\ R_j < MR \\ x_{i,j},\ Otherwise \end{cases} \quad (10)$$

where, $v_{i,j}$ is new solution, $x_{k,j}$, $x_{k+1,j}$ and $x_{k+2,j}$ are uniformly random solutions, R_j is randomly chosen parameter in the range of $(0,1)$ and $a_1, a_2, a_3$are quotients which show that how much every particular individuals partitioned in new solution. MR is control parameter that controls the probability of the modification on the parameters of the new solution. After evaluation of solutions generated by onlooker bees, in scout bee phase, abandoned solution is replaced with a new generated solution by using (10). Finally, the algorithm memorizes the best solution found so-far and moves to the next iteration.

Improved artificial bee colony algorithm: Karaboga and Akay (2012) presented a modified ABC algorithm for constrained optimization problems. To recognize this algorithm through this study the abbreviation MABC is used to refer to this algorithm. In this algorithm, similar to constrained ABC (Karaboga and Basturk, 2007b) the solution search equations used in employed bee and onlooker bee phases are generated using Eq. (5). Furthermore, the scout bee phase generates a random solution using Eq. (2). In addition, the Deb's rules applied to be used as constraint handling method. However, the main difference between these two algorithms is related with the probability selection mechanism. Based on suggested probability method several infeasible solutions in the population are allowed to enhance diversity. Probability values of infeasible solutions are introduced inversely proportional to their constraint violations. The probability selection mechanism is defined as in follows:

$$p_j = \begin{cases} 0.5 + 0.5\left(\frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}\right)\ violation_i = 0 \\ 0.5\left(1 - \frac{violation_i}{\sum_{j=1}^{SN} violation_i}\right)\ Otherwise \geq 0 \end{cases} \quad (11)$$

where, $violation_i$ is the value of the violation function corresponding to the solution x_i. The violation is defined using Eq. (7) and $fitness_i$ is the fitness value of the solution x_i which is determined by:

$$fitness_i = \begin{cases} 1/1 + f_i, if\ f_i \geq 0 \\ 1 + |f_i|, if\ f_i < 0 \end{cases} \quad (12)$$

Furthermore, a statistical parameter analysis is handled and the appropriate value for each control parameters is obtained.

Modified artificial bee colony algorithm: A Modified ABC algorithm (M-ABC) was introduced by presenting some modifications to the selection mechanism, the equality and boundary constraints and scout bee operators presented (Mezura-Montes and Cetina-Domínguez, 2012). The mechanisms to handle equality and boundary constraints are enhanced to direct the search towards a more appropriate region. A dynamic parameter control mechanism is proposed as described by:

$$\varepsilon(n+1) = \varepsilon(n)/dec \quad (13)$$

where, n is current cycle number dec is the decreasing rate value of each cycle and dec should be greater than one. The control mechanism works to satisfy equality constraints at the beginning of the process and then with the progresses of algorithm through the cycles. This tolerance is reduced so that the constraint violation of the generated solutions is lower than those of the solutions obtained in the first cycles. In addition, in contrast with original ABC, in this algorithm the information translation process is done using tournament selection mechanism and Deb's rules were applied as selection principle in the tournament. Consequently, no modified fitness values or probabilities must be computed as in the original constrained ABC. As suitable attractors in a constrained search space, the smart flight operator was suggested to enhance the location of solution. This technique is employed in the scout bee instead of the uniform random generation of solutions employed by general ABC while the new solution v_i is generated using Eq. (6). Finally, to prevent the generation of values outside the boundary of search space a mechanism is introduced to the employed, onlooker and scout bees phases by:

$$v_{i,j} = \begin{cases} 2x_{min,j} - v_{i,j}, if\ v_{i,j} < x_{min,j} \\ 2x_{max,j} - v_{i,j}, if\ v_{i,j} > x_{max,j} \end{cases} \quad (14)$$

Scout behavior modified artificial bee colony algorithm: Scout behavior Modified Artificial Bee Colony (SM-ABC) algorithm introduced as a powerful algorithm for problem with high dimensionality and

when the best solution lies in the boundaries of the feasible region of search space. In this algorithm the behavior of the scout bee is improved in order to get the ability to exploit the vicinity of the current best solution using Eq. (7). In addition, the way to control the tolerance for equality constraints is revised (Mezura-Montes and Cetina-Domínguez, 2009).

Improved artificial bee colony algorithm using genetic operators: GI-ABC is a genetically inspired ABC algorithm presented in Bacanin and Tuba (2012). The modifications have been made by adopting genetic algorithm in the process of replacement of abandoned solutions. In this algorithm uniform crossover and mutation operators from genetic algorithms applied to improve the exploitation ability of ABC algorithm (Karaboga and Basturk, 2007a) Based on the proposed method, after certain number of iterations called break point, the randomly generated solution in scout bee phase is replaced with directed search towards the best solution where exhaustive exploitation is performed around the best-so-far solution using uniform crossover operator. Then, to prevent each variable to be fixed, mutation operator is applied and each solution parameter is mutated using following Eq. (15):

$$v_{i,j} = x_{i,j} + \psi_{i,j}\left(x_{k,j} - v_{i,j}\right) \quad (15)$$

where $i = 1, 2, \cdots, SN$, $x_{k,j}$ is randomly generated solution, $v_{k,j}$ is a new generated solution with best fitness value, $\psi_{i,j}$ is random number in $(-1, 1)$. Based on above description the scout bee phase works as follows:

- If $iteration < BP$ generate a new random solution using equation (2).
- If $iteration < BP < SBP$ and if RPcondition is satisfied, using Eq. (15) otherwise use Eq. (2).
- If $SBP < iteration$ and if RP condition is satisfied, then select two solutions with best fitness value otherwise, use Eq. (2).

where, BP is break point SBP is the second break point and *it* is iteration and Replacement Rate RP is parameter.

Modified constrained artificial bee colony using smart bee: Smart Bee ABC (SB-ABC) algorithm is another modification to the original constrained ABC which applies its historical memories to improve the quality of solutions (Stanarevic *et al*., 2011). The motivation of present smart bee ABC algorithm come from the fact that, ABC algorithm does not consider the initial population to be feasible, therefore a smart bee mechanism is proposed to ABC algorithm to exploit its historical memories to enhance fitness value of solutions. This mechanism is applied instead of Deb's rules in the selection process. In SB-ABC algorithm, the position of the best-so-far solution and its fitness value is memorized and is replaced with the new randomly generated solution in two cases:

- If the new solution is infeasible solution
- If the new solution is feasible solution but it does not have better fitness value

EXPERIMENTAL RESULTS

This section is to evaluate, analyze and compare the performance of the aforementioned eight constrained ABC algorithms. At this aim several constrained benchmark functions are considered form CEC 2006 (Liang *et al*., 2006). Table 1 describes various kinds of these test functions (linear, nonlinear, polynomial, quadratic and cubic) with different numbers of decision variables, different kinds (linear inequalities, linear equalities, nonlinear inequalities and nonlinear equalities) and numbers of constraints. In this table ρ 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. In addition, the value of parameters used in each algorithm is given in Table 2.

DISCUSSION AND CONCLUSION

Experimental results: In this study the performances of several constrained ABC algorithms are evaluated and compared. In Experiment (1) each test function was repeated 30 times for each algorithm and then the best, worst, mean solution as well as standard deviation is recorded and the results are listed in Table 3. From the results in this table, ABC, MABC, M-ABC, Mo-ABC, SM-ABC, GI-ABC algorithm have found optimal minima for the seven test functions g01, g03, g04, g06, g08, g11, g12. SF-ABC can generate near optimal solutions for two functions g06, g01 while produce optimal solutions for five functions g03, g04, g08, g11, g12.

The SB-ABC can find near optimal solution for one function g06 and for six functions can find the optimal solution g01, g03, g04, g08, g11, g12. As can be seen from Table 3 on function g02, GI-ABC, M-ABC and SM-ABC performed better than other algorithms while SF-ABC is not successful on this function. SF-ABC and GI-ABC can provide better result for function g05 where SB-ABC cannot perform well. GI-ABC, SM-ABC outstands in solving function g07 compare to other algorithms. The M-ABC algorithm is more successful than MABC on this

Table 1: The main characteristics of the test problems

Problem	Type	n	ρ	LI	NI	LE	NE	α
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

Table 2: Parameters setting

Algorithm	Parameters	Reference
ABC	NP = 40; SN = 20; MCN = 6000; limit = SN×d; MR = 0.8; SPP = SN ×d; ε = 0.0001	Karaboga and Basturk (2007a)
MABC	NP = 40; SN = 20; MCN = 6000; limit = 0.5 ×SN ×d; e = 0.0001; MR = 0.8; SPP = 0.5 × SN ×d	Karaboga and Akay (2012)
M-ABC	NP = 40; SN = 20; MCN = 6000; limit = 145; MR = 0.8; ε = 0.0001	Mezura-Montes and Cetina-Domínguez (2012)
SM-ABC	NP = 40; SN = 20; MCN = 6000; limit = 145; MR = 0.8; ε = 0.0001; dec = 1.002	Mezura-Montes and Cetina-Domínguez (2009)
MO-ABC	NP = 40; SN = 20; MCN = 6000; limit = 145; a_2 = 0.4; a_3 = 0.3; MR = 0.8; ε = 0.0001; a_1= 0.3	Subotic (2011)
SB-ABC	NP = 40; SN = 20; MCN = 6000; limit = SN × d; MR = 0.8; SPP = SN × d; ε = 0.0001	Stanarevic *et al*. (2011)
GI-ABC	NP = 40; SN = 20; MCN = 6000; limit = 150; MR = 0.923; ε = 0.0001	Bacanin and Tuba (2012)
SF-ABC	NP = 40; SN = 20; MCN = 6000; limit = 145; dec = 1.002; PC = 46; gc = 1160; δ = 1:0; MR = 0.923; ε = 0.0001	Mezura-Montes *et al*. (2010)

Table 3: Comparisons based on best, mean, worst and standard deviation

Problem		ABC	MABC	M -ABC	SM-ABC	GI-ABC	SF-ABC	MO-ABC	SB-ABC
g01	Best	-15.0000000	-15.0000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000
	Mean	-15.0000000	-15.0000000	-15.000000	-15.000000	-15.000000	-14.163000	-15.000000	-15.000000
	Worst	-15.0000000	-15.0000000	-15.000000	-15.000000	-15.000000	-12.525000	-15.000000	-15.000000
	Std. dev	0.0000000	0.0000000	0.000000	0.000000	0.000000	0.923000	0.000000	0.000000
g02	Best	0.8035989	0.8035995	0.803614	-0.803617	-0.803617	-0.708944	-0.803610	-0.803500
	Mean	-0.7935470	-0.7935120	-0.799450	-0.799421	-0.800213	-0.471249	-0.793510	-0.792980
	Worst	-0.7496490	-0.7498960	-0.778176	-0.805767	-0.796850	-0.319535	-0.744580	-0.748990
	Std. dev	0.0141230	-0.0122300	-0.006440	0.006840	0.003400	0.010800	0.016000	0.025000
g03	Best	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Mean	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Worst	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Std. dev	-0.0000000	0.0000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000
g04	Best	-30665.539000	-30665.5390000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000
	Mean	-30665.539000	-30665.5390000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000
	Worst	-30665.539000	-30665.5390000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000	-30665.539000
	Std. dev	0.0000000	0.0000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g05	Best	5126.4890000	5126.4900000	5126.734000	5126.678000	5126.502000	5126.506000	5126.657000	5126.532000
	Mean	5158.6220000	5182.6790000	5178.178000	5178.392000	5264.654000	5126.527000	5162.506000	5185.752000
	Worst	5438.1830000	5324.3570000	5317.183000	5120.885000	5126.345000	5126.859000	5229.119000	5438.423000
	Std. dev	75.3370000	68.3490000	56.000000	56.100000	38.275000	0.079300	47.820000	75.367000
g06	Best	-6961.8140000	-6961.8140000	-6961.814000	-6961.814000	-6961.814000	-6961.814000	-6961.814000	-6961.814000
	Mean	-6961.8140000	-6961.8130000	-6961.814000	-6961.814000	-6961.814000	-6961.813000	-6961.813000	-6961.816000
	Worst	-6961.7940000	-6961.8050000	-6961.814000	-6961.814000	-6961.814000	-6961.805000	-6961.804000	-6961.813000
	Std. dev	0.0050000	0.0002000	0.000000	0.000000	0.000000	0.000200	0.000100	0.000700
g07	Best	24.3286100	24.3250000	24.312000	24.317000	24.303000	24.164500	24.323000	24.326000
	Mean	24.4698800	24.4690000	24.416000	24.415000	24.376000	24.658000	24.456000	24.485000
	Worst	25.1875400	24.6100000	24.794000	24.289000	24.343000	25.551000	24.929000	25.227000
	Std. dev	0.1854900	0.1175000	0.127000	0.124000	0.033000	0.326000	0.135000	0.195000
g08	Best	-0.0958250	-0.0958250	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
	Mean	-0.0958250	-0.0958250	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
	Worst	-0.0958200	-0.0958250	-0.095825	-0.095825	-0.095820	-0.095825	-0.095825	-0.095825
	Std. dev	0.0000000	0.0000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g09	Best	680.6329710	680.6300000	680.633000	680.631000	680.630500	680.632000	680.631000	680.643000
	Mean	680.6401260	680.6360000	680.647000	680.656000	680.630000	680.645000	680.635000	680.685000
	Worst	680.6498300	680.6570000	680.676000	680.632000	680.559000	680.858000	680.636000	680.678000
	Std. dev	0.0039981	0.0038700	0.0543000	0.015500	0.060900	0.041200	0.004000	0.014000
g10	Best	7053.129000	7058.8430000	7051.775000	7051.694000	7049.279000	7049.516600	7053.320000	7053.958000
	Mean	7224.547000	7220.3950000	7233.810000	7233.646000	7192.423000	7116.823600	7167.801000	7224.425000
	Worst	7389.195000	7362.7406000	7604.129000	7121.456000	7389.195000	7362.740600	7418.334000	7604.239000
	Std. dev	132.9420000	122.6700000	1013.000000	112.000000	81.859000	82.124000	83.008000	133.937000
g11	Best	0.7500000	0.7500000	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
	Mean	0.7500000	0.7500000	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
	Worst	0.7500000	0.7500000	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
	Std. dev	0.0000000	0.0000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g12	Best	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Mean	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Worst	-1.0000000	-1.0000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
	Std. dev	0.0000000	0.0000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
g13	Best	0.7594810	0.7587400	0.053890	0.053967	0.053950	0.053986	0.454000	0.767900
	Mean	0.9679549	0.9679500	0.157791	0.158489	0.248700	0.263854	0.456000	0.972000
	Worst	1.0000000	1.0000000	0.441978	0.104522	0.058850	1.000000	0.489000	1.000000
	Std. dev	0.0549543	0.0568900	0.017200	0.172760	0.203000	0.216000	0.021000	0.056900

Table 4: Comparison by means of gap based on mean solution

Problems	ABC	MABC	M-ABC	SM-ABC	MO-ABC	GI-ABC	SF-ABC	SB-ABC
g01	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	5.58300	0.0000
g02	1.25330	1.25768	0.518778	0.52239	1.25790	0.4238	41.35900	1.3238
g03	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	0.00000	0.0000
g04	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	0.00000	0.0000
g05	0.62662	1.09590	1.012270	1.01227	1.5605e-04	2.6949	5.656e-04	1.1558
g06	0.00000	1.436e-06	0.000000	0.00000	0.00000	0.0000	1.436e-5	2.873e-05
g07	0.67420	1.346e-06	0.000000	0.00000	0.00000	0.2879	1.44820	0.7364
g08	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	0.00000	0.0000
g09	1.488e-03	8.815e0-4	2.498e0-3	3.8199e-03	7.346e-04	0.0000	2.20e-03	8.0807e-03
g10	2.48670	2.42780	2.618150	2.61580	1.68175	2.0310	0.95859	2.4850
g11	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	0.00000	0.0000
g12	0.00000	0.00000	0.000000	0.00000	0.00000	0.0000	0.00000	0.0000
g13	1694.17030	1694.16170	192.476300	193.77010	745.22710	360.9823	389.07130	1701.6682

Table 5: Holm-bonferroni procedure for the 8 algorithms over the 13 test problems under consideration

j	Algorithm	R_j	p_j	α_j	SED	Significant
7	SB-ABC	77e+00	0.000	7.1429e-03	-0.1733e+02	Reject
6	SF-ABC	93e+00	0.5000e+00	8.33e-03	0.0000	Accept
5	ABC	94e+00	0.8607e+00	1.000e-02	0.10833e+01	Accept
4	SM-ABC	96e+00	0.9999e+00	1.250e-02	0.32500e+01	Accept
3	M-ABC	98e+00	1.0000e+00	1.667e-02	0.54167e+01	Accept
2	MO-ABC	101e+00	1.0000e+00	2.500e-02	0.86667e+01	Accept
1	GI-ABC	105e+00	1.0000e+00	5.000e-02	0.13000e+02	Accept
	MABC	93e+00				

problem while SF-ABC cannot provide satisfactory results. It is obvious from the table MO-ABC and MABC outperformed on problem g09. GI-ABC, SF-ABC and MO-ABC achieved superior results on problem g10 while SB-ABC generates poor results in this problem. GI-ABC, MO-ABC and M-ABC algorithms outperformed the other considered algorithms on the solution of the problem g13 while SB-ABC has shown the poorest performance.

Comparison based on error: In this section the solution quality is measured by the gap between the optimal solution and the best solution where gap is defined as:

$$\left|\frac{(best\ solution - optimal\ solution)}{optimal\ solution}\right| \times 100$$

Table 4 demonstrated that in all algorithms the big gap between the optimal solution and the best solution is for problem g13. In GI-ABC the best gap is obtained by g01, g03, g04, g06, g08, g09, g11, g12 problems is zero. However, for ABC the gap for g01, g02, g03, g04, g06, g08, g11, g12 problem is zero. MABC algorithm approximately behaves like ABC algorithm. SM-ABC, M-ABC and MO-ABC algorithms generate zero-gap for g01, g03, g04, g06, g07, g08 and g11 and g12 problems. SF-ABC algorithm has zero-gap for problems g03, g04, g08, g11, g12. Algorithm has zero-gap for problems g03, g04, g08, g11 and g12. SB-ABC obtain zero- gap for problems g01, g03, g04, g08, g11, g12.

Ranking by means of holm-bonferroni method: A statistical study on the performance of the considered algorithms regarding the performance of MABC algorithm (Karaboga and Akay, 2012) carried out by employing the Holm-Bonferroni mechanism (Liang *et al*., 2006; Holm, 1979). The results of using Holm-Bonferroni method is tabulated in Table 5. In this method, the 8 algorithms under analysis have been ranked based on their average performance calculated over 13 problems. Then, a score R_j has been assigned to each algorithm for $j = 1, 2, \cdots, N_A$ where N_A is the number of algorithms under consideration. The score is calculated as follows. In this mechanism first the performance of each algorithm for each function is ranked from one to eight.

A score of eight is assigned to the problem with the best performance, 7 is assigned to the second best and so on. While the algorithm presenting the worst performance scores is marked as 1. Then, after obtaining score for each algorithm, they are summed up and then based on the obtained results the score are sorted. The SED shows the standard error of difference. Using the values of SED, the corresponding cumulative normal distribution values p_j have been calculated. These p_j then compared with the corresponding value of $\alpha = 0.05/j$. If $p < \alpha$ then a test is reported as significant otherwise results are reported as insignificant. From Table 5 it is obvious that reference algorithm MABC significantly outperforms SB-ABC algorithm and is comparative with other algorithms.

Convergence analysis: This subsection shows the comparison of the convergence speed (Iterations) in term of number of iterations between the original constrained ABC, MABC, M-ABC, SM-ABC, GI-ABC, SF-ABC, MO-ABC and SB-ABC for problems g02, g05, g10 and g13. It can be observed that in Fig. 1. The SF-ABC has lowest convergence comparing to the

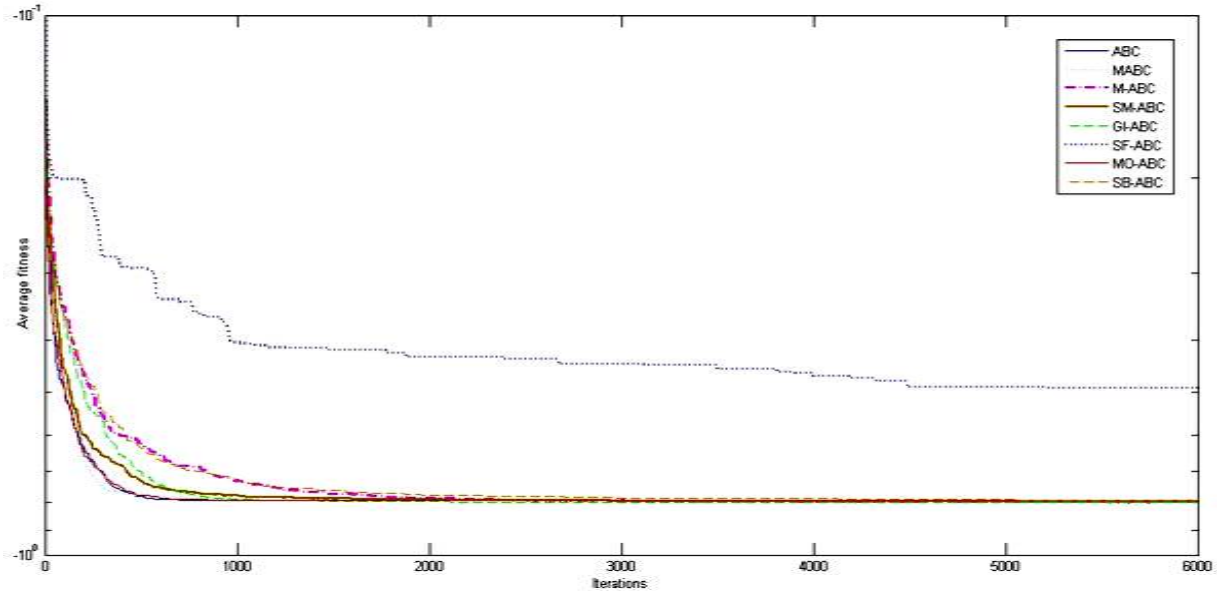

Fig. 1: Iterations to convergence for problem g02

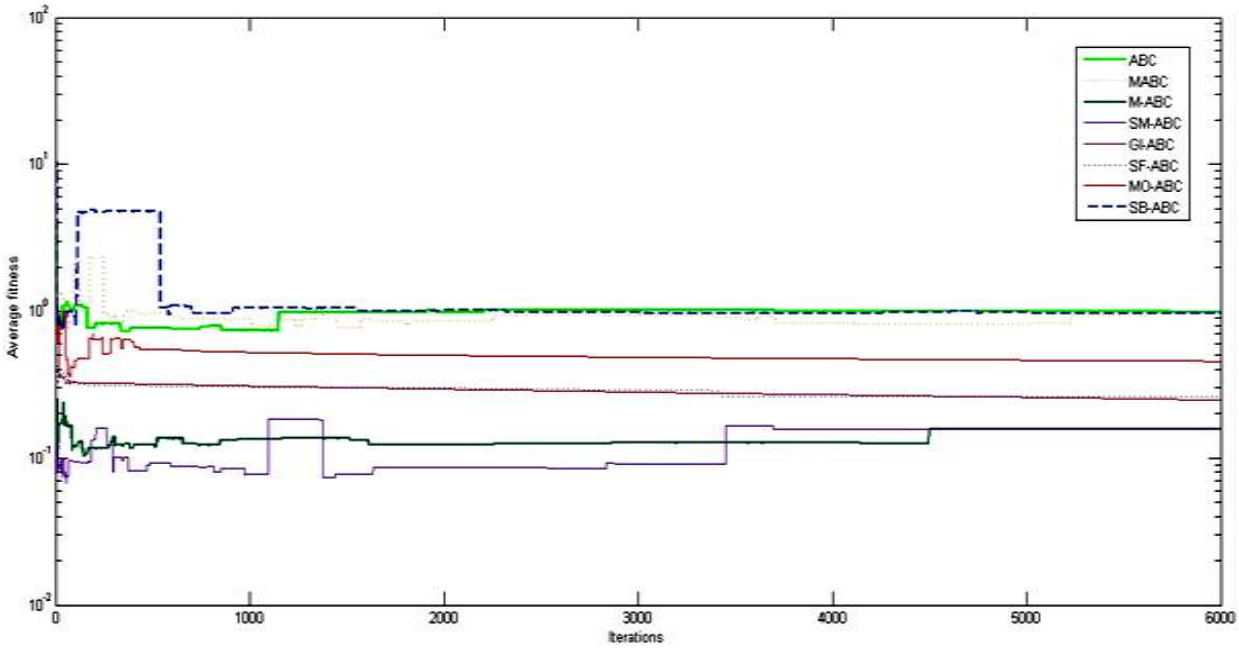

Fig. 2: Iterations to convergence for problem g13

other algorithms under consideration. Figure 2 the SB-ABC obtains smallest convergence speed and ABC algorithm stands in the second place.

ACKNOWLEDGMENT

The authors would like to thank Universiti Teknologi Malaysia and the Ministry of Education, Malaysia for the financial funding through RUG 08H47 grant.

REFERENCES

Akay, B. and D. Karaboga, 2012. A modified artificial bee colony algorithm for real-parameter optimization. Inform. Sciences, 192: 120-142.

Aydina, D., S. Özyön, C. Yaşar and T. Liao, 2014. Artificial bee colony algorithm with dynamic population size to combine economic and emission dispatch problem. Int. J. Elec. Power., 45: 144-153.

Bacanin, N. and M. Tuba, 2012. Artificial Bee Colony (ABC) algorithm for constrained optimization improved with genetic operators. Stud. Inform. Control, 21(2): 137-146.

Banitalebi, A., M.I.A. Aziz, A. Bahar and Z.A. Aziz, 2015. Enhanced compact artificial bee colony. Inform. Sciences, 298(20): 491-511.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Comput. Method Appl. M., 186(2-4): 311-338.

Dorigo, M. and T. Stützle, 2010. Ant Colony Optimization: Overview and Recent Advances. In: Gendreau, M. and J.Y. Potvin (Eds.), Handbook of Metaheuristic. International Series in Operations Research and Management Science, Springer US, New York, pp: 227-263.

Drias, H., S. Souhila and S. Yahi, 2005. Cooperative bees swarm for solving the maximum weighted satisfiability problem. In: Cabestany, J., A. Prieto and D.F. Sandoval (Eds.), IWANN 2005. LNCS 3512, Springer-Verlag, Berlin, Heidelberg, pp: 318-325.

Farmer, J.D., N.H. Packard and A.S. Perelson, 1986. The immune system, adaptation and machine learning. Physica D, 22(1): 187-204.

Fogel, L.J., A.J. Owens and M.J. Walsh, 1966. Artificial intelligence through simulated evolution. John Wiley, New York, pp: 227-296.

Gao, W.F., S.Y. Liu and L.L. Huang, 2013. A novel artificial bee colony algorithm with Powell's method. Appl. Soft Comput., 13(9): 3763-3775.

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

Holm, S., 1979. A simple sequentially rejective multiple test procedure. Scand. J. Stat., 2(6): 65-70.

Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Vol. 200.

Karaboga, D. and B. Akay, 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput., 214(1): 108-132.

Karaboga, D. and B. Basturk, 2007a. A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. J. Global. Optim., 39(3): 459-471.

Karaboga, D. and B. Basturk, 2007b. Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. In: Melin, P. *et al*. (Eds.), Foundations of Fuzzy Logic and Soft Computing, IFSA, 2007. LNAI 4529, Springer-Verlag, Berlin, Heidelberg, pp: 789-798.

Karaboga, D. and B. Basturk, 2008. On the performance of Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput., 8(1): 687-697.

Karaboga, D. and B. Akay, 2012. A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. Appl. Soft Comput., 11(3): 3021-3031.

Karaboga, D. and B. Gorkemli, 2014. A quick Artificial Bee Colony (qABC) algorithm and its performance on optimization problems. Appl. Soft Comput., 23: 227-238.

Kennedy, J., 2010. Particle Swarm Optimization. In: Encyclopedia of Machine Learning. Springer, US, pp: 760-766,

Kiran, M.S., H. Hakli, M. Gunduz and H. Uguz, 2015. Artificial bee colony algorithm with variable search strategy for continuous optimization. Inform. Sciences, 300: 140-157.

Koza, J.R., 1992. Genetic programming: On the programming of computers by means of natural selection. MIT Press, Cambridge, MA, USA.

Li, G., N. Peifeng and X. Xiao, 2012. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Appl. Soft. Comput., 12(1): 320-332.

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

Mezura-Montes, E. and O. Cetina-Domínguez, 2009. Exploring promising regions of the search space with the scout bee in the artificial bee colony for constrained optimization. Proceeding of the Artificial Neural Networks in Enginnering Conference (ANNIE'2009), ASME Press.

Mezura-Montes, E. and O. Cetina-Domínguez, 2012. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. Appl. Math. Comput., 218(22): 10943-10973.

Mezura-Montes, E., M. Damián-Araoz and O. Cetina-Domingez, 2010. Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. Proceeding of IEEE Congress on Evolutionary Computation (CEC), pp: 1-8.

Passino, K.M., 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Contr. Syst., 22(3): 52-67.

Rechenberg, I., 1978. Evolutionary Strategies. In: Schneider, B. and U. Ranft (Eds.), Simulationsmethoden in der Medizin und Biologie. Springer-Verlag, Berlin, Heidelberg, pp: 83-114.

Simon, D., 2008. Biogeography-based optimization. IEEE T. Evolut. Comput., 12(6): 702-713.

Stanarevic, N., M. Tuba and N. Bacanin, 2011. Modified artificial bee colony algorithm for constrained problems optimization. Int. J. Math. Model. Method. Appl. Sci., 5(3): 644-651.

Storn, R. and P. Kenneth, 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim., 11(4): 341-359,

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

Tang, K.S., K.F. Man, S. Kwong and Q. He, 1996. Genetic algorithms and their applications. IEEE Signal Proc. Mag., 13(6): 22-37.

Xiang, W.L. and M.Q. An, 2013. An efficient and robust artificial bee colony algorithm for numerical optimization. Comput. Oper. Res., 40(5): 1256-1265.