Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Artificial Bee Colony with Different Mutation Schemes: A comparative study

Iyad Abu Doush, Basima Hani F. Hasan,
Mohammed Azmi Al-Betar, Eslam Al Maghayreh,
Faisal Alkhateeb, Mohammad Hamdan

Abstract

Artificial Bee Colony (ABC) is a swarm-based metaheuristic for continuous optimization. Recent work hybridized this algorithm with other metaheuristics in order to improve performance. The work in this paper, experimentally evaluates the use of different mutation operators with the ABC algorithm. The introduced operator is activated according to a determined probability called mutation rate (MR). The results on standard benchmark function suggest that the use of this operator improves performance in terms of convergence speed and quality of final obtained solution. It shows that Power and Polynomial mutations give best results. The fastest convergence was for the mutation rate value (MR=0.2).

Keywords: Artificial Bee Colony, Evolutionary Algorithms, Mutation, Meta-heuristic algorithm, Polynomial mutation.

1 Introduction

Meta-heuristic and evolutionary algorithms are computational methods that solve a problem by iteratively trying to improve a candidate solution with regard to a given fitness function. While meta-heuristics do not guarantee reaching an optimal solution if one is available, they are useful for solving combinatorial optimization problems in both science and engineering. Many algorithms that mimic natural phenomena

such as genetic algorithms [21], simulated annealing [5], ant colony optimization [8], and particle swarm optimization [8] have shown significant efficiency in solving many real-world problems.

The Artificial Bee Colony (ABC) is an optimization algorithm based on the intelligent behavior of honey bees [1, 16]. In ABC, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. In the ABC algorithm, the goal of the bees (employed bees, onlookers and scouts) is to discover the places of food sources with high nectar amount and finally the one with the highest nectar. The algorithm basically works as follows: employed bees go to their food source and return to hive. The employed bee whose food source has been abandoned becomes a scout and starts searching a new food source. Onlookers choose food sources depending on dances. Later on, the abandoned food sources are replaced with the new food sources discovered by scouts and the best food source found so far is registered. These steps are repeated until reaching the stop condition.

The ABC has been used to solve several optimization problems (e.g., forecasting stock markets [3, 18, 12], capacitated vehicle routing problem [29, 3, 9], multiproduct manufacturing system [2], and combat air vehicle path planning [13, 22, 30]). Other researchers work on modifiying the original ABC (e.g., [4, 17]).

In ABC, some scouts choose the food sources randomly based upon a randomized mutation function. In this paper, we explore the use of different mutation functions as a replacement for the original random mutation used in the ABC algorithm, specifically in the scout bees' phase, to enhance the convergence speed. In particular we used five mutation schemes (non-uniform mutation, Makinen, Periaux and Toivanen (MPT) mutation, power mutation, polynomial mutation and best-based mutation). The effectiveness of the updated ABC algorithms are evaluated by using eight benchmark functions with different characteristics (Sphere function, Step function, Schwefel's problem 2.26, Six-Hump Camel-Back, Shifted Sphere, Shifted Schwefel's problem, Shifted Rosenbrock and Shifted Rastrigin).

The remainder of this paper is organized as follows: Section 2 introduces the Artificial Bee Colony (ABC) Algorithm. Section 3 presents the different mutation methods. The experimental environment is presented in Section 4. Finally, we conclude in Section 5.

2 The Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm was first proposed by Karaboga in [14, 16]. In a real bees colony, there are different types of specialized bees performing different tasks. The main goal of the bees is to maximize the amount of nectar stored in the hive.

According to the ABC algorithm, the bees' colony involves three different types of bees. Employed bees, onlooker bees and scouts. Half of the colony are employed bees and the reset are onlookers. Employed bees exploit food sources visited previously and provide the onlooker bees with information regarding the quality of the food sources they are exploiting. The onlooker bees use the information shared with employed bees to decide where to go. Scout bees explore the environment randomly looking for new sources of food. When a food source exhausted, the corresponding employed bee becomes a scout. The main steps of the ABC algorithm are described below.

Step1: Initialize the food source positions.

In the ABC algorithm the position of a food source represents a possible solution of the optimization problem and the amount of nectar at each food source represents the fitness of the corresponding solution. In this step of the algorithm, solutions x_i (where $i = 1 \dots n$) are randomly generated within the ranges of the problem's parameters, where n is the number of food sources (one source for each employed bee). Each solution is a vector of d dimensions, where d is the number of the problem's parameters.

Step 2: Each employed bee generates a new food source and exploits the better one.

The new food source (solution) is generated according to the following formula:

$$y_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}),$$

where φ_{ij} is a random number in the range [-1,1], k is the index of the solution chosen randomly and $j = 1, \ldots, d$.

After generating the new solution y_i, the employed bee compares between it and the original solution x_i and exploits the better one.

Step 3: Each onlooker bee chooses a food source based on its quality, generates a new food source and then exploits the best one.

An onlooker bee selects a food source based on the probability value associated with it. The probability value is calculated as follows:

$$p_i = \frac{Fitness_i}{\sum_{k=1}^{n} Fitness_k},$$

where $Fitness_i$ is the fitness of solution i, and n is the number of food sources which is equal to the number of employed bees.

Step 4: Stop the exploitation of the food sources exhausted and convert its employed bees into scouts.

A solution (represented by a food source) is said to be exhausted if it has not been improved after a predetermined number of execution cycles. The employed bee of each exhausted food source is converted into a scout that performs a random search for another solution (food source) based on the following formula:

$$x_{ij} = x_j^{min} + (x_j^{max} - x_j^{min}) * rand,$$

where x_j^{max} and x_j^{min} are the upper and the lower bounds of parameter (decision variable) j.

Step 5: Keep the best solution (food source) found so far.

Step 6: Repeat steps 2 to 5 until the termination condition is satisfied.

3 Mutation Methods

The mutation method is an essential operator in EAs [7]. It normally provides a mechanism to explore unvisited regions in the search space. It operates with less consideration to the natural principle of the 'survival of the fittest'. Any successful EA should have a mutation mechanism to ensure the diversification of the search space while makes use of the accumulative search.

Genetic Algorithm can be seen as the most popular EA algorithm that is widely used for optimization problems [25, 26]. It begins with population of individuals generated randomly. Evolutionary, it selects, recombines and mutates the current population to come up with a new population hopefully better. It exploits the current population using selection and recombination operators. It also explores the search space using mutation. This is necessary to prevent getting stuck in the local optima and increasing the chance of finding the global optima.

The way of diversifying the individuals in the population have been widely studied [11, 10, 7]. The mutation operator in GA randomly and structurally changes some genes in the individual without considering the characteristics of their parents. In order to implement a mutation operator, two issues should be watched: i) the probability of using mutation over population and ii) the power of mutation represented by the perturbation obtained in an individual.

Similarly to GA, other population-based method have a mutation operator to diversify the search, e.g., the Random consideration in Harmony Search Algorithm, the scout bee in Artificial Bee Colony (ABC). Particularly in ABC, the scout bee provides a mechanism to diversify the individuals and therefore to prevent the search to fault down in a local optima trap.

Figure 1 flowcharts the scout bee process. At each iteration, all individuals, $\boldsymbol{x}_i, \forall i \in 1 \dots SN$, in the population will be examined using $\mathsf{Scout}[.]$ vector. Note that the $\mathsf{Scout}[.]$ vector contains an accumulative counter information about each individual regarding if they improved. In case the individual (say $\mathbf{x}_i = (x_{i,1}, x_{i,2} \dots, x_{i,N})$, where N is the number of genes) is improved in such iteration, the $\mathsf{Scout}[i]$ will be

initialized by 0, otherwise it will be incremented by 1 until a certain *limit* exceeded, then a Do_mutation() operator will be applied.

In general, the continuous optimization problem is formulated as follows

$$\min\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbf{X}\},$$

where $f(\boldsymbol{x})$ is the objective function; $\boldsymbol{x} = \{x_i \mid i = 1, \ldots, N\}$ is the set of decision variables (or genes). $\mathbf{X} = \{\boldsymbol{X}_i \mid i = 1, \ldots, N\}$ is the possible value range for each gene, where $\boldsymbol{X}_i \in [L_{x_i}, U_{x_i}]$, where L_{x_i} and U_{x_i} are the lower and upper bounds for the gene x_i respectively and N is the number of genes.

In this paper, five mutation operators have been investigated in the scout bee operator. These mutation operators are controlled by a Mutation Rate (MR). The purpose of mutation is to diversify the search direction and to prevent the convergence into local optimum. Algorithm 1 shows that each gene in the selected individual will be examined for whether or not it will be changed randomly. In each mutation type, the change process is different as we will discuss below.

Algorithm 1 Scout Bee Procedure

```
for i = 1, ··· , SN do
  if Scout[i] < limit then
    for j = 1, ··· , N do
      if U(0, 1) < MR then
        DO_Mutation()
      end if
    end for
  end if
end for
```

3.1 Original mutation

The original mutation is proposed by [15], which is called random mutation. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows

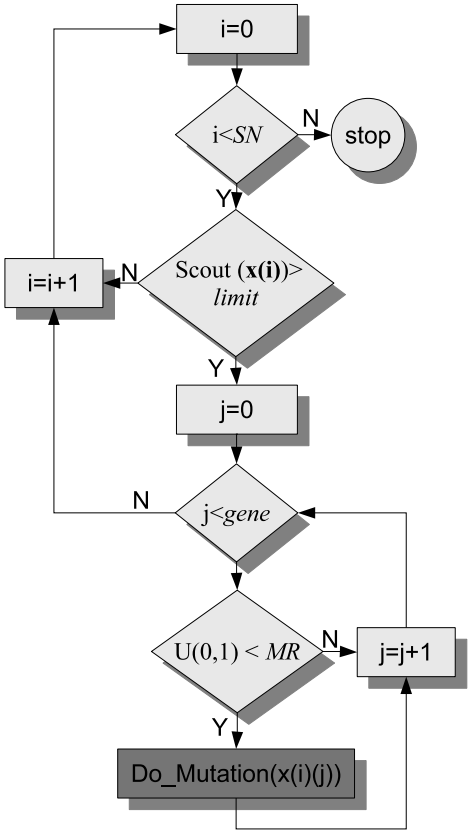


Figure 1. The flowchart Scout Bee Operation

$$x'_{i,j} = L_{x_j} + U(-1,1)(U_{x_j} - L_{x_i}).$$

In this type of mutation, the value of the gene is replaced randomly with a value within the range of decision variable [21] in the abandoned solution i. Note that $U(-1,1)$ generates a random number between -1 and 1.

3.2 Non-uniform random mutation

The non-uniform random mutation is one of popular mutation types that is widely used in GA [21, 20]. In non-uniform mutation, as the generations increase, the step size decreases, therefore making a uniform search in the initial stage of search and very little at later stages. In this type of mutation, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{i,j} \times (1 - U(0,1)^{(\frac{1-t}{MSN})^b}),$$

where b is a system parameter determining the degree of dependency of iteration number (in this study, the value of b is fixed to 5 as recommended by a previous study [21]), t is the generation (or iteration) number. And MSN refers to the maximum number of iterations in ABC. Note that the gene $x'_{i,j}$ is assigned with a value in a range $[0, x_{i,j}]$.

3.3 Makinen, Periaux and Toivanen (MPT) Mutation

Makinen, Periaux and Toivanen mutation, proposed by [19], is a relatively new mutation and has been applied to solve multidisciplinary shape optimization problem in addition to a large set of optimization problems with constrained nature. In this type of mutation, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = (1 - \hat{t}_j) \times L_{x_j} + \hat{t}_j \times U_{x_j},$$

where

$$\hat{t}_j \leftarrow \begin{cases} t_j - (t_j) \times (\frac{t_j - r_j}{t_j})^b & r_j < t_j \\ t_j & r_j = t_j, \\ t_j + (1 - t_j) \times (\frac{r_j - t_j}{1 - t_j})^b & r_j > t_j \end{cases}$$

and

$$t_j = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - x_{i,j}}.$$

Normally, the value of $b = 1$.

3.4 Power Mutation

This type of mutation operator is based on power distribution, and proposed by [7]. It is an extended version of MPT mutation. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows:

$$x'_{i,j} \leftarrow \begin{cases} x_{i,j} - s \times (x_{i,j} - L_{x_j}) & t < r \\ x_{i,j} - s \times (U_{x_j} - x_{i,j}) & otherwise, \end{cases}$$

where

$$t = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - x_{i,j}},$$

and s is a random number generated according to the final distribution, and r is a uniform random number generated in the range 0 and 1, $r \in [0, 1]$.

3.5 Polynomial mutation

Polynomial mutation was first introduced by Deb and Agrawal in [6] which has been successfully applied for single and multi-objective optimization problems. In this type of mutation, the gene ($x_{i,j}$) that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{i,j} + \delta_q \times (U_{x_j} - L_{x_j}),$$

where

$$\delta_q \leftarrow \begin{cases} [(2r) + (1-2r) + (1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1 & r < 0.5 \\ 1 - [(2(1-r)) + (2(r-0.5)) + (1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} & otherwise, \end{cases}$$

$$\delta_1 = \frac{x_{i,j} - L_{x_j}}{U_{x_j} - L_{x_j}},$$

$$\delta_2 = \frac{U_{x_j} - x_{i,j}}{U_{x_j} - L_{x_j}}.$$

Note that r is a uniform random number generated in the range 0 and 1, $r \in [0, 1]$.

3.6 Best-based Mutation

This type of mutation is initially proposed by [27]. It is effective and powerful mutation type for unconstrained large scaled optimization problems. In this type of mutation, four individuals are randomly selected (i.e., $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}, \boldsymbol{x}_{r3}, \boldsymbol{x}_{r4})$ from the entire population. After that, the gene $(x_{i,j})$ that met the probability of MR is changed as follows:

$$x'_{i,j} = x_{best,j} + F \times (x_{r1,j} - x_{r2,j}) + F \times (x_{r3,j} - x_{r4,j}),$$

where $x_{best,j}$ is the gene in the best individual from the entire population. F is an important parameter that ensures the balance between exploration and exploitation which normally is experimentally determined, and takes a value range between 0 and 1.

4 Experimental results

We used 8 global minimization benchmark functions (Table 2) to evaluate different mutation methods used in ABC (see Table 1). These benchmark functions have been selected as one function for each set of unique characteristics (e.g, unimodal, multi-modal, and separable), as it is shown in the last column of Table 2. The benchmark functions

were implemented with a multi-dimension (N=100), with the exception to Six-Hump Camel-Back function which is two-dimensional.

We conducted four different experiements, each one with different mutation rate (MR = 0, 0.2, 0.5, and 0.8). In each experiment we tested six different mutation methods: ABC original, Non-uniform, MPT, Power, Polynomial, and Best mutation. Each experiment was repeated 30 times with different random seeds. The average of the best values obtained by the algorithms is calculated. The obtained results of the mean best values and standard deviation are shown in tables 3, 4, 5, and 6.

The experiments were executed on a P4 machines with 1 GB of RAM using C++ under Microsoft Visual Studio environment. In all the experiments the values for common parameters are as follows: the population size (NP) was 100, the food sources was 50, the stopping criteria = 10,000, and the algorithm ran for 30 times. The limit is defined using the formula D* NP*0.5 which uses the dimension of the problem and the colony size to determine the limit value. These values are similar to what has been suggested in the state of the art methods.

The results in tables 3, 4, 5, and 6 show that for the sphere function the mutation rate (MR=0.2) gives the best result, and the power mutation (M4) gives the best result. The best result for sphere function using other mutation rate values (MR= 0.5 and 0.8) was given by the polynomial mutation (M5).

On the other hand, for the Schwefel problem function the mutation rate (MR=0.2) gives the best result with MPT mutation (M3). The shifted rosenbrock gives the best result using the mutation rate (MR=0.5) with MPT mutation (M3). The best result for shifted rosenbrock function using other mutation rate values (MR= 0.2 and 0.8) was given by the Non-uniform mutation (M2).

For the rest of the functions (i.e., step, camel-back, shifted sphere, shifted schwefel, and shifted rastrigin) all the mutation methods (M1-M6) with all mutation rates reached the global optimal solution.

Figure 3 shows the effect of using the mutation rate value (MR=0.8) on the convergence speed of the six different mutation methods. The compared benchmark functions are: sphere and shifted rosenbrock.

For the two functions polynomial mutation (M5) has the fastest convergence speed. The slowest convergence speed for the sphere function was for Power mutation (M4). On the other hand, for the shifted rosenbrock, Best mutation (M6) has the slowest convergence speed.

Figure 2 compares the effect of using different mutation rate values on the convergence speed of the mutation method used (i.e., M1 to M6 in Table 1). Generally speaking the mutation rate (MR=0.2) has the fastest convergence speed for the mutation methods M1, M4, and M6 (Original ABC, Power, and Best). For the mutation methods M3 and M5 (MPT and Polynomial), the mutation rate (MR=0.8) has the fastest convergence speed. These observations confirm the mean best results obtained, which are usually using mutation rate value (MR=0.2). This value allows diversifying the population without changing the population to be far from optimal solution.

Table 1. The different mutation schemes used in the experiments.

Original ABC	Non-uniform	Makinen, Periaux and Toivanen (MPT)	Power	Polynomial	Best
M1	M2	M3	M4	M5	M6

5 Conclusion and Further Work

It is common in swarm-based algorithms to hybridize with operators from evolutionary algorithms such as mutation. Normally, the hybridization comes by adding a common to an existing algorithm. This paper investigated the performance of ABC using different mutation schemes. The mutation happened in the ABC algorithm after reaching the limit and calling the scout bees. We updated the original ABC algorithm with six different mutation schemes. The application of the mutation is performed according to a defined parameter called mutation rate (MR). We evaluated the updated algorithms of ABC with 8 global benchmark functions used in the literature. We investigated the mean best value and studied the convergence speed using the six different mutation methods parametrized with four different mutation rates.

Table 2: Benchmark functions used to evaluate HS variations

Function Name	Expression	Search Range	Optimum Value	Category [24]
Sphere function [23]	$f_1(x) = \sum_{i=1}^{N} x_i^2$	$x_i \in [-100, 100]$	$\min(f_1) = f(0, \ldots, 0) = 0$	unimodal
Step function [23]	$f_2(x) = \sum_{i=1}^{N} (\lfloor x_i + 0.5 \rfloor)^2$	$x_i \in [-100, 100]$	$\min(f_2) = f(0, \ldots, 0) = 0$	discontinuous unimodal
Schwefel's problem 2.26 [31]	$f_3(x) = -\sum_{i=1}^{N} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right) \right)$	$x_i \in [-500, 500]$	$\min(f_3) = f(420.9687, \ldots, 420.9687) = -12569.5$	difficult multimodal
Six-Hump Camel-Back function [23]	$f_4(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$x_i \in [-5, 5]$	$\min(f_4) = f(-0.08983, 0.7126) = -1.0316285$	low dimensional

Continuation of Table 2

Function Name	Expression	Search Range	Optimum Value	Category [24]
Shifted Sphere function [28]	$f_5(x) = \sum_{i=1}^{N} z_i^2 + f\_bias_1$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_5) = f(o_1, \ldots, o_N) = f\_bias_1 = -450$	unimodal, shifted, separable, and scalable
Shifted Schwefel's problem 1.2 [28]	$f_6(x) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} z_j \right)^2 + f\_bias_2$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_6) = f(o_1, \ldots, o_N) = f\_bias_6 = -450$	unimodal, shifted, non-separable, and scalable
Shifted Rosenbrock [28]	$f_7(x) = \sum_{i=1}^{N-1} (100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2) + f\_bias_6$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-100, 100]$	$\min(f_7) = f(o_1, \ldots, o_N) = f\_bias_6 = -390$	multi-modal, shifted, non-separable, and scalable
Shifted Rastrigin [28]	$f_8(x) = \sum_{i=1}^{N} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_9$, where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$	$x_i \in [-5, 5]$	$\min(f_8) = f(o_1, \ldots, o_N) = f\_bias_9 = -330$	multi-modal, shifted, separable, and scalable

Generally speaking the results show that Power and Polynomial mutations give best results. The mutation rate value (MR=0 and 0.8) gives the slowest convergence. On the other hand, the fastest convergence was for the mutation rate value (MR=0.2).

The future work can be experimenting different mutation schemes after modifying the original ABC algorithm to apply mutation on early stages of the algorithm. Currently, the algorithm uses mutation after exceeding the limit of reaching constant optimal solution. We could benefit more from the different mutation methods presented in this paper by applying them early in the ABC algorithm.

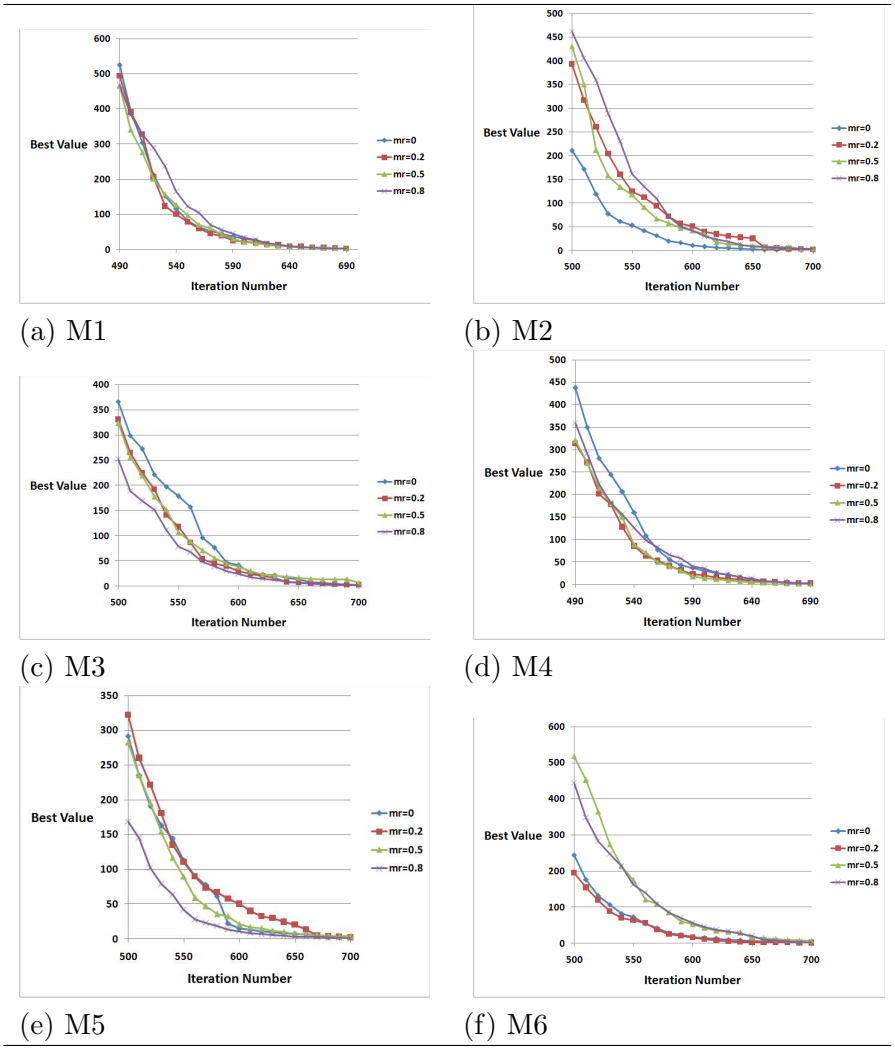
It might be interesting to adaptively select the mutation operator based on algorithm performance. But the algorithm needs to use a single best mutation rate, and mutation operator should be recommended, as it is not allowed to change this operator for each benchmark. For example, if the ABC algorithm is stuck in local optima, then use another operator in the hope to improve search capabilities and reach the global optima.

Table 3. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0

Sphere	**Step**	**Schwefel's 2.26**	**Camel-Back**	**Shifted Sphere**	**Shifted Schwe-fel's**	**Shifted Rosen-brock**	**Shifted Rast-rigin**
2.15E-15	0	-4.199724E+04	-1.03163E+00	-450	-450	3.90054E+02	-330
(1.60935E-16)	(0)	(0.25795393)	(0)	(0)	(0)	(0.102094052)	(0)

References

[1] A. Aderhold, K. Diwold, A. Scheidler, and M. Middendorf. Artificial bee colony optimization: A new selection scheme and its performance. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 283–294, 2010.



(a) M1 (b) M2

(c) M3 (d) M4

(e) M5 (f) M6

Figure 2. The convergence speed for the sphere function using different mutation rates (10,000 iterations). The figures show the portions with noticable difference, here it is within the range 0 – 600.

Table 4. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.2

	M1	M2	M3	M4	M5	M6
Sphere	1.60E-15 (2.4049E-16)	1.63E-15 (2.50432E-16)	1.60E-15 (2.95416E-16)	**1.56E-15 (2.32707E-16)**	1.93E-15 (2.07427E-16)	1.66E-15 (2.41625E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199731E+04 (0.223375361)	-4.199734E+04 (0.256613668)	**-4.199724E+04 (0.258221147)**	-4.199731E+04 (0.217112715)	-4.199730E+04 (0.204236734)	-4.199736E+04 (0.160781139)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90025E+02 (0.088819745)	**3.90018E+02 (0.033415342)**	3.90035E+02 (0.092817557)	3.90031E+02 (0.03157629)	3.90039E+02 (0.081382741)	3.90023E+02 (0.05924405)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

[2] S. Ajorlou, I. Shams, and M.G. Aryanezhad. Optimization of a multiproduct conwip-based manufacturing system using artificial bee colony approach. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2, 2011.

[3] B. Akay and D. Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, pages 1–14, 2010.

[4] Nebojsa Bacanin and Milan Tuba. Artificial bee colony (abc) algorithm for constrained optimization improved with genetic oper-

Table 5. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.5

	M1	M2	M3	M4	M5	M6
Sphere	2.10E-15 (1.84796E-16)	2.16E-15 (1.95236E-16)	2.09E-15 (2.65087E-16)	2.13E-15 (2.10979E-16)	**1.85E-15** (**1.61943E-16**)	2.14E-15 (2.2148E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199731E+04 (0.171671967)	-4.199736E+04 (0.195965045)	-4.199733E+04 (0.291429296)	-4.199731E+04 (0.214449308)	**-4.199728E+04** (**0.241189581**)	-4.199735E+04 (0.247377045)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90025E+02 (0.056847244)	3.90036E+02 (0.0972493)	**3.90016E+02** (**0.031588391**)	3.90019E+02 (0.03157629)	3.90039E+02 (0.066354577)	3.90032E+02 (0.083040386)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

ators. *Studies in Informatics and Control*, 21(2):137–146, 2012.

[5] Thomas Back. *Evolutionary Algorithms in Theory and Practice*. OXFORD UNIVERSITY PRESS, New York, Oxford, 1996.

[6] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[7] Kusum Deep and Manoj Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 193(1):211 – 230, 2007.

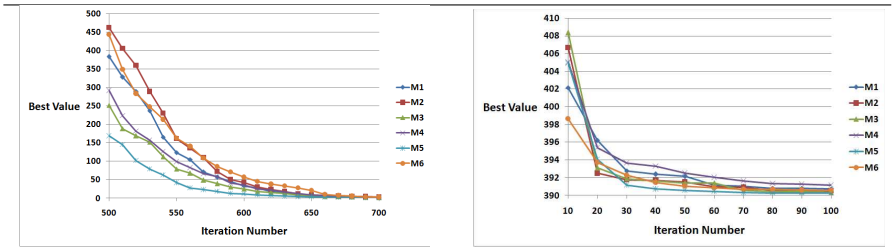
Table 6. Average and standard deviation (±SD) of the benchmark function results ($N = 100$), mr=0.8

	M1	M2	M3	M4	M5	M6
Sphere	2.08E-15 (2.32587E-16)	2.10E-15 (1.97095E-16)	2.18E-15 (1.55962E-16)	2.16E-15 (1.62883E-16)	**1.86E-15 (1.65775E-16**)	2.09E-15 (2.15497E-16)
Step	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)	**0** (**0**)
Schwefel's problem 2.26	-4.199741E+04 (0.228840154)	-4.199729E+04 (0.165952493)	-4.199728E+04 (0.277530158)	-4.199734E+04 (0.158621939)	**-4.199726E+04 (0.252550211**)	-4.199728E+04 (0.175545128)
Camel-Back	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)	**-1.03163** (**0**)
Shifted Sphere	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Schwefel's problem 1.2	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)	**-450** (**0**)
Shifted Rosenbrock	3.90034E+02 (0.074387028)	**3.90020E+02 (0.041508398**)	3.90031E+02 (0.05580612)	3.90042E+02 (0.143667564)	3.90027E+02 (0.06110496)	3.90061E+02 (0.18150363)
Shifted Rastrigin	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)	**-330** (**0**)

[8] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[9] M. El-Abd. Black-box optimization benchmarking for noiseless function testbed using artificial bee colony algorithm. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1719–1724. ACM, 2010.

[10] Mohammad Hamdan. A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms. *International Journal on Artificial Intelligence Tools*, 20(1):209–219, 2011.



(a) Sphere (b) Shifted Rosenbrock

Figure 3. The convergence speed using different mutation methods (10,000 iterations and MR=0.8).

[11] F. Herrera and M. Lozano. Two-loop real coded genetic algorithms with adaptive control of mutation step sizes. *Applied Intelligence*, 13:187–204, 2002.

[12] T.J. Hsieh, H.F. Hsiao, and W.C. Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2), 2010.

[13] F. Kang, J. Li, and Z. Ma. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 2011.

[14] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.

[15] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing*, pages 789–798, 2007.

[16] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

[17] Dervis Karaboga and Bahriye Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 11(3):3021 – 3031, 2011.

[18] S. Kockanat, T. Koza, and N. Karaboga. Cancellation of noise on mitral valve doppler signal using iir filters designed with artificial bee colony algorithm. *Current Opinion in Biotechnology*, 22:S57–S57, 2011.

[19] Raino A.E. Makinen, Jacques Periaux, and Jari Toivanen. Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2):149–159, 1999.

[20] Z Michalewicz, T Logan, and S Swaminathan. Evolutionary operators for continuous convex parameter space. In *Proceedings of Third Annual Conference on Evolutionary Programming*. 1994.

[21] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[22] SN Omkar, J. Senthilnath, R. Khandelwal, G. Narayana Naik, and S. Gopalakrishnan. Artificial bee colony (abc) for multi-objective design optimization of composite structures. *Applied Soft Computing*, 2009.

[23] M. G. H. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[24] Quan-Ke Pan, P.N. Suganthan, M. Fatih Tasgetiren, and J.J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216(3):830 – 848, 2010.

[25] J.E. Smith and T.C. Fogarty. Operators and parameter adaptation in genetic algorithms. *Soft computing*, 1(2):81–87, 1997.

[26] W.M. Spears. *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.

[27] Nadezda Stanarevic. Comparison of different mutation strategies applied to artificial bee colony algorithm. In *Proceedings of the 5th European conference on European computing conference*, ECC'11, pages 257–262, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS).

[28] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report#2005005, IIT Kanpur, India, Nanyang Technological University, Singapore, 2005.

[29] WY Szeto, Y. Wu, and S.C. Ho. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, 2011.

[30] C. Xu, H. Duan, and F. Liu. Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning. *Aerospace Science and Technology*, 2010.

[31] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82 –102, 1999.

Iyad Abu Doush[1], Basima Hani F. Hasan[1], Mohammed Azmi Al-Betar[2], Eslam Al Maghayreh[1], Faisal Alkhateeb[1]

Received December 17, 2012

[1] Yarmouk University
Computer Science Department, Irbid, Jordan

Iyad Abu Doush
E–mail: *iyad.doush@yu.edu.jo*

[2]Jadara University
Computer Science Department, Irbid, Jordan