Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100211

Comparison of Artificial Bee Colony Algorithms on Engineering Problems

Bekir Afşar[1] *and Doğan Aydın*[2,*]

[1] Department of Computer Engineering, Muğla Sıtkı Koçman University, Muğla, Turkey
[2] Department of Computer Engineering, Dumlupınar University, Kütahya, Turkey

Received: 11 Aug. 2015, Revised: 29 Oct. 2015, Accepted: 30 Oct. 2015
Published online: 1 Mar. 2016

Abstract: Artificial Bee Colony (ABC) algorithm is a swarm intelligence based recent optimization technique that mimics from real bee colonies. After the initial proposal of ABC, several variants of original algorithm have been proposed like as other optimization algorithms. Although performances of well-known ABC algorithms are known and comparison works exist on benchmark test functions, the real performance of these algorithms are not known on engineering problems. In this paper, several ABC algorithms are reviewed and compared on three different engineering optimization problems using default parameter settings and tuned parameter settings. Moreover, the best ABC variants obtained from experimental results are compared with contemporary algorithms in literature. The results have shown that ABC algorithms are also competitive with recent state-of-the-art algorithms on engineering problems.

Keywords: Artificial bee colony, engineering optimization problems, performance comparison, swarm intelligence

1 Introduction

Swarm intelligence (SI) [1] is the joint behavior of the self-organized and decentralized systems. SI systems consist of simple agents which interact with each other and with their environments. Some examples of SI systems can be seen in nature such as ant and bee colonies, flocks of birds, and schools of fish. In recent years, researchers have invited new artificial swarm algorithms inspired by real swarm systems and their problem solving behavior. Particle swarm optimization (PSO) inspired by the behaviors of blocks of birds or schools of fish [2], ant colony optimization (ACO) inspired by the foraging behaviors of ant colonies [3], cuckoo search (CS) inspired by the brooding behaviors of cuckoo species [4], Cat Swarm Optimization (CSO) based on the behavior of cats [5] are some of the examples of SI algorithms. Comparison of such SI methods on engineering problems can be found in literature [7].

Artificial Bee Colony (ABC) algorithm [6] is another recent SI algorithm which is inspired by foraging behavior of honey bees. ABC algorithm is firstly designed for tackling numerical function optimization problems. Due to the presence of small number of control parameters, ease to implementation, its simplicity and efficiency, ABC has widely used to solve many optimization problems [8,9,10].

Even though the original ABC algorithm achieved succesful results for the multimodal and multidimensional basic problems, it was found to be less successful in comparison with state-of-the-art algorithms for composite and non-separable function as well as having slow convergence rate [11]. Therefore, several improvements of the original ABC algorithm were introduced over the years. Unfortunately, experimental tests of the each variant have been made under different conditions. Moreover, we can't see painstaking effort on the fine-tuning of the algorithm parameters. Although some indications on the performance of the different variants may be obtained from the available results, they are not fully conclusive (i) for what concerns the relative performance of the variants and (ii) for what concerns the relative importance of the introduced modifications. To tackle this problem, in our previous comparison work, we have studied on the performance of the well-known ABC algorithms on a comprehensive set of benchmark functions [12]. However, the performances obtained from a set of benchmark functions may not guarantee a similar performance on engineering optimization problems [13]. In addition to this, there is no enough study to show the

* Corresponding author e-mail: dogan.aydin@dpu.edu.tr

performances of these ABC variants on engineering optimization problems.

Aim of this study is to analyze the performance of recent ABC variants, and to compare fairly them with each other and state-of-the-art algorithms. In experimental study, we have used three engineering problems: parameter estimation for frequency-modulated (FM) sound waves problem, spread spectrum radar poly phase code design problem and economic power dispatch problem. At first, we compared ABC algorithms by using default parameter values suggested by the authors. Then the comparison has done with the tuned parameter values found by Iterated F-race, the automatic parameter configuration tool [14,15]. Finally, we compared the results of ABC variants with state-of-the-art algorithms to see the real position of ABC algorithms in the literature.

2 Artificial Bee Colony (ABC) Algorithm

The ABC algorithm, was proposed by [6], is a SI-based optimization algorithm inspired by foraging behaviors of honeybees. There are three types of honeybees in the ABC algorithm: employed, onlooker and scout bees. Employed bees are responsible to calculate nectar amounts of the food sources and the number of employed bees is equal to the number of the food sources at the foraging area. Onlooker bees are responsible to choose the food sources which have good nectar amounts and the number of onlooker bees is equal to the number of employed bees. Scout bees are responsible to discover new food sources. Employed bee becomes a scout, when its food source has been abandoned and this food source is replaced with the new one found by the scout bee.

Possible solution of an optimization problem is represented by the position of a food source. The higher of the nectar amount of the food sources the better solution of the optimization problem. So the quality of the solution is represented by the nectar amount and its name is fitness value in the ABC algorithm. First, an initial population which contains SN (number of food sources) solutions is generated randomly according to following equation:

$$x_{i,j} = x_j^{\min} + \varphi_{i,j}(x_j^{\max} - x_j^{\min}) \tag{1}$$

where $\varphi_{i,j}$ is a uniform random number in [0, 1] for dimension j and food source i, x_j^{min} is the minimum value of the search range, x_j^{max} is the maximum limit of the search range on dimension j. At this time, $limit$ parameter is also initialized for each food source. It is used for when a food source should be abandoned. Each solution x_i ($i = 1,2,3,\ldots,SN$) is an n-dimensional vector and has fitness value. In the ABC algorithm, the fitness value ($fitness_i$) is calculated by the following equation:

$$fitness_i = \begin{cases} \frac{1}{1+f_i}, f_i \geq 0, \\ 1+abs(f_i), f_i < 0 \end{cases} \tag{2}$$

where f_i is the objective value of food source i. After the initialization of the population, employed bees take place and visit the food sources. They search better food sources according to the equation:

$$v_{i,j} = x_{i,j} + \varphi_{i,j}(x_{i,j} - x_{k,j}), i \neq k \tag{3}$$

where k and i can take values of $(1,2,3,\ldots,SN)$, j is a randomly selected dimension ($j = 1,2,3,\ldots,D$), $\varphi_{i,j}$ is a uniform random number in [-1, 1], $x_{i,j}$ and $x_{k,j}$ are the position of the reference food source i and a randomly selected food source k in dimension j, respectively. If the new food source is better than the old one, it is replaced with the old one. After the employed bees stage, onlooker bees take place to evaluate the food sources which are found by the employed bees. They visit a food source and if the food source has a higher nectar amount, it is selected by the onlooker bees according to the selection probability which is determined by any food source i as follows:

$$p_i = \frac{fitness_i}{\sum_{n=1}^{SN} fitness_n} \tag{4}$$

According to the behaviors of the employed and onlooker bees, it can be seen that they search for good food sources and focus to the good solutions area to carry out the exploitation. To escape local optimums, algorithm must explore new solutions. If employer and onlooker bees cannot improve the location of a food source at $limit$ times, this food source has been exhausted. This means that food source is not good enough to explore new solutions. At this time, scout bees take place and try to find a new food source instead of exhausted food source according to the equation 1. Therefore, scout bees are responsible to explore the whole search space for avoiding local optimums.

3 Considered Variants of Artificial Bee Colony Algortitmh

Exploration and exploitation are the two important concepts in the evolutionary based algorithms. Exploration is the ability to search whole search space to find good new solutions [16]. Exploitation is the ability to find the optimum solution and focus to the search space which includes the optimum solution. These two concepts must be balanced for good convergence speed and to avoid local optimums. In ABC algorithm, employed and onlooker bees ensure exploitation, scout bees ensure exploration. In order to improve the convergence characteristics and to avoid to get stuck on the local optimums, some new variants of the original ABC algorithm have been proposed by several researchers [17]. In the following sub-sections, brief explanations of most recent ABC algorithms are given.

3.1 Gbest-guided Artificial Bee Colony (GbABC) Algorithm

The starting point of the GbABC [18,19] is, the solution search equation of the original ABC algorithm is good at exploration but poor at exploitation. Therefore, in order to improve the exploitation of the original ABC, the search of the new candidate solutions is guided by using the information of the global best solution. The solution search equation described by equation (3) of the original ABC is modified as follows:

$$v_{i,j} = x_{i,j} + \varphi_{i,j}(x_{i,j} - x_{k,j}) + \psi_{ij}(x_{gbest,j} - x_{k,j}), i \neq k \quad (5)$$

where $x_{gbest,j}$ is the j^{th} element of the global best solution. $\psi_{i,j}$ is a uniform random number in [0, C], where C is a nonnegative constant which is used for balancing the exploration and exploitation of the candidate solution search. If C takes 0, the equation returns to the original equation. Exploitation of the solution search equation (5) can be increased by increasing the value of C. But, very big values of the C can decrease the exploration. Consequently, very big values of the C parameter are not used for balancing the exploitation and exploration.

3.2 Gbest Distance-guided Artificial Bee Colony (GbdABC) Algorithm

Diwold et al. have proposed two different versions for the ABC algorithm [19]. The first one was aforementioned GbABC and the second one was GbdABC algorithm. GbdABC uses the same search equation in equation (5) but the neighbor food source in mutation function, x_k, is not selected randomly. That means each neighbor has not the same selection probability. Instead, each neighbour food source has a probability to select related to the selected reference food source as follows:

$$p_k = \frac{(1/dist(loc_i, loc_k))}{\sum_{n=1, n\neq i}^{SN}(1/dist(loc_i, loc_j))} \quad (6)$$

where p_k is the probability of neighbor x_k, loc_x is the location of a food source and distance between two food source locations, loc_x and loc_y, is denoted by $dist(loc_x, loc_y)$. The underlying idea of this modification is to prefer nearer neighbors because it is probable to find better locations by searching between two good solutions which are probably close to each other in many type of optimization functions [19].

3.3 Best-so-far Selection Artificial Bee Colony (BABC) Algorithm

In BABC, three major changes were proposed to improve the exploitation and exploration of the original ABC algorithm. These changes are; best-so-far method, adjustable search radius and an objective-value-based comparison method. In order to enhance the exploitation, efficiency of the onlooker bees is improved by using best-so-far method. So, convergence speed of the best-so-far ABC is accelerated [20]. The solution search equation of the original ABC is modified as shown in following equation:

$$v_{i,d} = x_{i,j} + f_b(\varphi_{i,j}(x_{ij} - x_{gbest,j})) \quad (7)$$

where j is a randomly selected dimension, $\varphi_{i,j}$ is a uniform random number [-1, 1], f_b is the fitness value of the best-so-far solution, $x_{gbest,j}$ is the best-so-far food solution in selected dimension j. The important change is that the values in all dimensions of each food source are updated at each iteration.

To avoid local optimums, global search ability is introduced for the scout bees. If the solution gets stuck on the local optimum, the scout bee will generate a new food source by using:

$$v_{i,j} = x_{i,j} + \varphi_{i,j}(w_{\max} - (itr_{current}/itr_{\max})(w_{\max} - w_{\min})) \quad (8)$$

where $v_{i,j}$ is a new feasible solution of a scout bee that is modified from the current position of an abandoned food source, $x_{i,j}$, w_{max} and w_{min} are the control parameters which define the minimum and maximum percentage of scout bee position adjustment, respectively. $itr_{current}$ presents the current iteration executed so far, and itr_{max} is for maximum iteration number for the algorithm [20]. According to the equation (8), founded food sources by the scout bee are far from optimal solution in early iterations but in later iterations it will converge to the optimal solution closely.

Third change is objective value comparison. In the original ABC, greedy selection is used by using the fitness values of the solutions. In the BABC, greedy selection is applied by using directly objective values of the solutions.

3.4 Modified Artificial Bee Colony (MABC) Algorithm

The idea behind the modified ABC algorithm is that the convergence speed of the original ABC algorithm is good at basic functions but poor at hybrid functions. In order to improve convergence speed, two major changes have been applied to the original ABC algorithm [16]. The first one is the modification rate (MR) parameter that determines how many parameters to be modified in search equation (3). The second change is scaling factor (SF) parameter that controls step size of the perturbation adaptively. In the solution search equation of the original ABC (3), the value of the random number $\varphi_{i,j}$ is between [-1,1] but in the modified ABC its value is between [$-SF$, SF]. Bigger values of SF can increase the convergence speed but decrease the exploitation. For this reason its value can be determined adaptively according to the Rechenberg's 1/5 mutation rule in the modified ABC algorithm.

3.5 Improved Artificial Bee Colony (IABC) Algorithm

In Improved ABC [21], new initialization approach and a novel search mechanism have been introduced to improve the convergence speed of the original ABC algorithm. The first change is initialization of the ABC algorithm which affects directly the convergence speed and also the quality of the final solution. Instead of the random initialization approach of the original ABC algorithm, a novel initialization approach which employs chaotic systems and the opposition-based learning method is used in the improved ABC algorithm.

The second change is a new search mechanism which includes two improved solution search equations namely *ABC/best/1* (9) and *ABC/rand/1* (10) which are defined below. While *ABC/best/1* uses the information of the best solution in the current population, *ABC/rand/1* explores the population:

$$v_{i,m} = x_{best,m} + \varphi_{i,j}(x_{i,m} - x_{k1,m}) \tag{9}$$

$$v_{i,m} = x_{k1,m} + \varphi_{i,j}(x_{i,m} - x_{k2,m}) \tag{10}$$

where $k1$ and $k2$ are two different random food source indexes and m is a positive integer that controls how many parameters to be changed.

To use the advantages of the above equations and avoid the shortages of them, two solution search equations are hybridized according to the parameter p which is the selecting probability of the two equations. So, the exploitation and the exploration of the improved ABC algorithm are balanced.

3.6 Chaotic Artificial Bee Colony (CABC) Algorithm

In order to improve the convergence speed of the original ABC algorithm and to prevent the original ABC to get stuck on local solutions, the chaotic ABC algorithm was proposed which uses chaotic maps [22]. Instead of the random number sequences, chaotic sequences are used because of their spread-spectrum characteristic, non-periodic, complex temporal behavior, and ergodic properties.

In the original ABC algorithm, the value of the limit parameter and the generated random numbers do not change at the new iterations. This situation reduces the convergence speed of the original ABC algorithm. So, to take advantages of the chaotic sequences, they are used at the initialization and scout bees stages instead of the random numbers. At the initialization stage, chaotic sequences are used for generating the initial food sources. In the original ABC algorithm, an employed bee becomes a scout bee when its food source cannot be improved after *limit* trails. However, after $limit/2$ trail employed bee becomes a scout bee in the chaotic ABC algorithm and this time scout bee searches new foods by using chaotic search. The chaotic maps; *Logistic*, *Circle*, *Gauss*, *Henon*, *Sinusoidal*, *Sinus*, *Tent* are used for chaotic search in the chaotic ABC algorithm.

3.7 Rosenbrock Artificial Bee Colony (RABC) Algorithm

RABC [23] algorithm was proposed for accurate numerical optimization that combines Rosenbrock's rotational direction method (RM) with an ABC algorithm. RM is used as a local exploitation tool for the original ABC algorithm. Fitness calculation mechanism is changed of the original ABC algorithm; rank-based fitness transformation is adopted:

$$fitness_i = 2 - SP + \frac{2(SP-1)(r_i-1)}{SN-1} \tag{11}$$

where $SP \in [1.0, 2.0]$ is the selection pressure and r_i is the rank of the solution (food source) i in entire population.

In the RABC algorithm, after the initialization stage, the population is evaluated and the best solution is memorized. Then the step size of the modified RM procedures is calculated according to the following equation:

$$\delta_j = 0.1\frac{\sum_{i=1}^{m}(\mathrm{x'}_{\mathrm{i,j}} - x_{gbest,j})}{m} \tag{12}$$

where δ_j is step size of the j^{th} dimension, x'_i is the i^{th} solution after ranking and m is the number of solutions selected to calculate the step size. After that, the modified RM procedure is called for some iterations with the best solution and the new best solution is obtained. After the ranking, the solution in the middle position is changed with the best-so-far solution. Then the exploration phase is executed by the ABC algorithm.

3.8 Incremental Artificial Bee Colony (IncABC) Algorithm

IncABC proposed incremental population size and hybridization with local search procedures to tackle large-scale benchmark functions [24]. These changes have been applied by the authors to particle swarm optimization (PSO) and ant colony optimization (ACO) before and the results have shown that performances of these algorithms were improved. Therefore, growing population is applied to the ABC algorithm, provided by a control parameter g. This means, a new food source is added to the population in every g iterations until a maximum number of food sources (SN_{max}) is reached. IncABC algorithm applies local search procedures to the ABC algorithm. Powell's conjugate directions set [25] and Lin-Yu Tseng's Mtsls1 [26] methods are hybridized as a local search procedure in the incremental ABC algorithm.

At every g iterations, new food source is added according to the (13) which uses the information of the best-so-far solution.

$$X_{new,j} = x_{new,j} + \varphi_{i,j}(x_{gbest,j} - x_{new,j}) \tag{13}$$

where $x_{new,j}$ is the randomly generated new food source location according to (1), $X_{new,j}$ is the updated location of the new food source.

Besides, some extra modifications have been applied to the original ABC algorithm. At the employed bee stage, new food source is generated around the best-so-far solution instead of the randomly selected food source. At the scout bee stage, similar replacement mechanism is used which adds a new control parameter R_{factor} to control the how much the new food source will be closer to the best-so-far food source, is used with the following equation:

$$v_{i,j} = x_{gbest,j} + Rfactor(x_{gbest,j} - x_{new,j}) \tag{14}$$

4 Problem Definitions

In this paper, we aim to compare ABC variants with three engineering optimization problems. The descriptions of these three problems are given in the following subsections.

4.1 Parameter Estimation for Frequency-modulated (FM) Sound Waves Problem

Sound waves are expressed with a six dimensional equations. The problem is to generate a sound (expressed in (15)) similar to target sound (expressed in (16)). The parameters of an FM synthesizer are going to be optimized and this is a six dimensional optimization problem. The equations of the sound waves are:

$$\begin{aligned} y(t) = a_1.\sin(w_1.t.\theta + a_2.\sin(w_2.t.\theta) \\ +a_3.\sin(w_3.t.\theta))) \end{aligned} \tag{15}$$

$$\begin{aligned} y_0(t) = (1.0).\sin((5.0).t.\theta - (1.5).\sin((4.8).t.\theta) \\ +(2.0).\sin((4.9).t.\theta))) \end{aligned} \tag{16}$$

where θ is $2\pi/100$ and the range of parameters is [-6.4, 6.35]. The fitness function of the problem is as follows:

$$f(X) = \sum_{i=0}^{100}(y(t) - y_0(t))^2 \tag{17}$$

The detailed description of this problem can be found in [27].

4.2 Spread Spectrum Radar Poly Phase Code Design Problem

Radar systems and spread spectrum communication systems uses a group of vertical poly phase coded signals which are designed specially in order to improve the system performance. These poly phase coded signals can be formulated as a nonlinear multivariable 20 dimensional optimization problem. The problem can be expressed as follows:

$$global \min_{x\in X} f(x) = \max\{\phi_1(x), ..., \phi_{2m}(x)\} \tag{18}$$

$X = \{(x_1, ...x_n) \in R^n | 0 \le x_i \le 2\pi, j = 1, ..., n\}$, where $m = 2n - 1$ and

$$\phi_{2i-1}(x) = \sum_{j=1}^{n} \cos(\sum_{k=|2i-j-1|+1}^{j} x_k), i = 1, ..., n \tag{19}$$

$$\phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos(\sum_{k=|2i-j|+1}^{j} x_k), i = 1, ..., n-1 \tag{20}$$

$$\phi_{m+i}(x) = -\phi_i(x), i = 1, ..., m \tag{21}$$

The purpose is minimizing the maximum value of coded signals as ϕ at the above equations. The detailed description of this engineering problem can be found in [27].

4.3 Economic Power Dispatch Problem

The economic power dispatch problems are about satisfying power demands of a place by combination of power generation units output while minimizing the total fuel cost. The objective function, f, for the problem is formulated as follows:

$$\min f = \sum_{i=1}^{n} F_i(P_i) \tag{22}$$

where F_i is the total fuel cost for the i^{th} generator (in $\$/h$) and P_i is the power of generator i (in MW). Each generator unit has inequality constraint expressed as each unit should be laid between maximum and minimum limits.

The fuel cost of the generator unit i without valve point effect is presented in polynomial function [13],

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \tag{23}$$

where a_i, b_i and c_i are cost coefficients of generator i. In literature, there are some test functions for the problem. In this paper, we only used 30-generators and six dimensional optimization problem. The detailed description of the problem and the test function can be found in [28].

5 Empirical Analysis

Performances of Artificial Bee Colony algorithms are analyzed in engineering optimization problems in this section. Section 5.1 summarizes the experimental procedure, while section 5.2 and section 5.3 discuss the results obtained from three test problems with using default parameters and tuned parameters respectively. Finally, in section 5.4, the comparison of best performing ABC variants with the contemporary algorithms is represented.

5.1 Experimental Setup

The main objective of the empirical analysis conducted for the purposes of this article is to determine the best performing ABC variant within all considered ABC algorithms. The real position of the best ABC variants compared with state-of-the-art algorithms will be determined as a second objective.

To do fair comparison, we performed two sets of experiments. First, we ran ABC algorithms using their default parameter settings. Then the parameters of each algorithm were tuned, and tuned version of ABC algorithms were ran on three engineering optimization problems. These problems were tackled at IEEE Conference on Evolutionary Competition (CEC) 2011 competition as well. Therefore, all experiments were conducted under the same conditions with CEC 2011 competition. All ABC variants were run 25 times for each problem. At the end, mean, median and the best objective function values are listed after executing the algorithms for 50000, 100000 and 150000 function evaluations (FEs). Experiments are run under C++ on Linux with a Quad-Core machine running @2.40 GHz with 4 GB of RAM.

5.2 Comparison of ABC Variants with Default Parameter Values

Parameter values are the important key for the performances of the ABC algorithms. For this reason, developers of the algorithms aim to find good parameter values for the problems by running their algorithms several times on the problems with different parameter values.

In our study, the parameter values of the ABC algorithms were taken from the original papers and these values are set as the default values of the parameters (Table 1). We ran each ABC algorithm on the three engineering problems with these values. Then, we compared the results of each ABC algorithm and presented the results in Table 3. The best values are shown in bold faces.

As shown in the Table 3, for the first problem, IABC is giving best performance when considering best and mean results. For median values, RABC algorithm is better than other variants. For the second problem, RABC algorithm is the best for all of the values (best, median, mean). Also IABC algorithm reaches the optimum value for the second problem. For the third problem, BABC algorithm gives the best performance for all of the values (best, median, mean). It is interesting that BABC algorithm is the worst for other problems.

5.3 Comparison of ABC Variants with Tuned Parameter Values

As in other meta-heuristics, parameter values affect directly the performance of ABC algorithms. In general parameters should be tuned according to the problem instances. Therefore, for fair comparison, we have obtained the tuned parameters of the algorithms by an offline automatic parameter tuning algorithm, Iterated F-Race [15].

F-Race algorithm is based on finding best parameter set after trying randomly generated candidate parameter values to the problem instances and eliminating the worst ones by statistical tests like as a race. In iterated F-Race, the procedure of F-Race is iteratively continues in a loop and each loop a new candidate sets are generated around the best candidate parameters found in previous step.

In our case, we have used three problems as the instances for Iterated F-Race. We have used default parameters of Iterated F-Race suggested by the authors. The tuned parameters are determined after obtaining the best results from the 5 independent runs of Iterated F-Race for each algorithm. The tuned parameter values are listed in Table 2.

After parameter tuning task, the algorithms were compared with using tuned parameters. The comparison results are shown in Table 4. The results, which are better than the results obtained by default parameters, are in bold faces. Table 4 indicates that the performances of the all considered algorithms were improved after parameter tuning almost all cases.

5.4 Comparison with the Contemporary Algorithms

Finally, best performing ABC variants were compared with the contemporary algorithms which have tackled to

Table 1: Default parameter values of ABC algorithms

	ABC	BABC	CABC	GbABC	GbdABC	IABC	IncABC	MABC	RABC
SN	62	100	10	40	15	25	8	10	25
limit	1	1	1	1	1	1	1.2	1	1
C	-	-	-	1.5	1	-	-	-	-
wMin	-	0.2	-	-	-	-	-	-	-
wMax	-	1	-	-	-	-	-	-	-
SF	-	-	-	-	-	-	-	1	-
m	-	-	-	-	-	1*D	-	0.4	-
p	-	-	-	-	-	0.25	-	-	-
K	-	-	-	-	-	200	-	-	-
rItr	-	-	-	-	-	-	-	-	15
NC	-	-	-	-	-	-	-	-	5
SP	-	-	-	-	-	-	-	-	1.5
R_{factor}	-	-	-	-	-	-	10^{-6}	-	-
SN_{max}	-	-	-	-	-	-	13	-	-
growth	-	-	-	-	-	-	8	-	-

Table 2: Tuned parameter values of ABC algorithms

	ABC	BABC	CABC	GbABC	GbdABC	IABC	IncABC	MABC	RABC
SN	54	21	88	44	82	23	9	62	66
limit	2.85	2.4	1.96	3.34	1.93	0.52	0.95	2.03	1.77
C	-	-	-	1.46	1.54	-	-	-	-
wMin	-	0.37	-	-	-	-	-	-	-
wMax	-	0.64	-	-	-	-	-	-	-
SF	-	-	-	-	-	-	-	0.09	-
m	-	-	-	-	-	0.58*D	-	0.33	-
p	-	-	-	-	-	0.01	-	-	-
K	-	-	90	-	-	76	-	-	-
rItr	-	-	-	-	-	-	-	-	71
NC	-	-	-	-	-	-	-	-	2
SP	-	-	-	-	-	-	-	-	1.88
R_{factor}	-	-	-	-	-	-	10^{-8}	-	-
SN_{max}	-	-	-	-	-	-	76	-	-
growth	-	-	-	-	-	-	2	-	-

same engineering optimization problems. For the Problem I and II, these algorithms are the competitors of CEC 2011 conference. For the Problem III, the recent algorithms which are designed for economic power dispatch are selected for comparison. The results are listed in Table 5, Table 6, and Table 7. The results show that ABC variants give competitive results compared with the other algorithms at the literature. BABC algorithm also gives the best performance in comparison with the state-of-the-art algorithms for the problem III.

6 Conclusion

The main objective of this paper was to determine the best performing ABC variants in engineering optimization problems. For this purpose, nine ABC algorithms were compared on three test problems taken from CEC 2011 competition. The comparison test was done using default parameters and tuned parameters to determine real performances of the algorithms.

The test results showed that there is no best performing algorithm for all considered test problems. IABC algorithm has better results for the first problem but worst results for the third problem. Just the opposite, BABC algorithm has good results for the third problem but very poor results for first and second problem. Besides, RABC algorithm is the best for second problem. Therefore, these results show that there is not a best algorithm for all problems. As in discussed in [49], the differences in search equations, that effect exploitation/exploration behavior, and local search strategies are the main reason of these results. IABC

Table 3: Comparison results of ABC algorithms with default parameter values

		Problem I			Problem II			Problem III		
Algorithm	**FEs**	**Best**	**Median**	**Mean**	**Best**	**Median**	**Mean**	**Best**	**Median**	**Mean**
ABC	50000	1.58E-03	8.54E-02	8.98E-01	7.30E-01	8.76E-01	8.69E-01	9.249E+02	9.249E+02	9.249E+02
	100000	8.03E-04	1.66E-02	5.16E-01	6.51E-01	8.42E-01	8.38E-01	9.249E+02	9.249E+02	9.249E+02
	150000	8.03E-04	1.44E-02	3.14E-01	6.46E-01	8.14E-01	8.10E-01	9.249E+02	9.249E+02	9.249E+02
BABC	50000	8.20E-02	4.07E+00	3.63E+00	6.12E-01	8.29E-01	8.27E-01	**9.232E+02**	**9.232E+02**	**9.233E+02**
	100000	2.76E-02	2.97E+00	2.77E+00	5.92E-01	8.03E-01	7.76E-01	**9.231E+02**	**9.232E+02**	**9.233E+02**
	150000	1.04E-02	2.82E+00	2.20E+00	5.91E-01	7.76E-01	7.58E-01	**9.231E+02**	**9.232E+02**	**9.232E+02**
CABC	50000	1.48E-02	7.11E-01	9.26E-01	6.98E-01	8.81E-01	8.65E-01	9.249E+02	9.250E+02	9.250E+02
	100000	1.48E-02	3.85E-01	6.86E-01	6.98E-01	8.28E-01	8.27E-01	9.249E+02	9.250E+02	9.250E+02
	150000	1.48E-02	2.59E-01	3.24E-01	6.98E-01	8.00E-01	8.03E-01	9.249E+02	9.249E+02	9.249E+02
GbABC	50000	2.12E-03	2.18E-02	7.49E-01	6.73E-01	8.53E-01	8.55E-01	9.249E+02	9.249E+02	9.249E+02
	100000	2.00E-05	1.16E-02	2.67E-01	6.65E-01	8.11E-01	7.97E-01	9.249E+02	9.249E+02	9.249E+02
	150000	2.00E-05	3.57E-03	1.20E-01	6.65E-01	7.78E-01	7.71E-01	9.249E+02	9.249E+02	9.249E+02
GbdABC	50000	5.73E-04	1.16E-02	**5.89E-01**	6.84E-01	8.47E-01	8.36E-01	9.249E+02	9.249E+02	9.249E+02
	100000	1.48E-04	5.37E-03	**1.18E-01**	6.59E-01	7.83E-01	7.94E-01	9.249E+02	9.249E+02	9.249E+02
	150000	8.60E-05	1.12E-03	**1.14E-01**	6.59E-01	7.78E-01	7.80E-01	9.249E+02	9.249E+02	9.249E+02
IABC	50000	**6.70E-07**	4.07E+00	4.34E+00	**5.00E-01**	7.11E-01	7.12E-01	9.249E+02	9.249E+02	9.249E+02
	100000	**0.00E+00**	3.36E+00	3.37E+00	**5.00E-01**	7.08E-01	6.94E-01	9.249E+02	9.249E+02	9.249E+02
	150000	**0.00E+00**	2.91E+00	2.70E+00	**5.00E-01**	6.98E-01	6.81E-01	9.249E+02	9.249E+02	9.249E+02
IncABC	50000	1.21E-05	4.06E+00	3.23E+00	5.59E-01	9.07E-01	8.88E-01	9.249E+02	9.249E+02	9.249E+02
	100000	1.14E-05	4.06E+00	3.23E+00	5.58E-01	9.06E-01	8.85E-01	9.249E+02	9.249E+02	9.249E+02
	150000	1.09E-05	3.93E+00	3.22E+00	5.57E-01	9.05E-01	8.83E-01	9.249E+02	9.249E+02	9.249E+02
MABC	50000	3.34E-02	5.77E-01	9.38E-01	6.90E-01	9.21E-01	9.17E-01	9.249E+02	9.249E+02	9.249E+02
	100000	3.34E-02	3.26E-01	4.90E-01	6.90E-01	8.97E-01	8.90E-01	9.249E+02	9.249E+02	9.249E+02
	150000	3.34E-02	2.53E-01	3.84E-01	6.87E-01	8.77E-01	8.64E-01	9.249E+02	9.249E+02	9.249E+02
RABC	50000	2.78E-06	**1.15E-03**	9.93E-01	**5.00E-01**	**6.67E-01**	**6.76E-01**	9.249E+02	9.249E+02	9.249E+02
	100000	2.20E-06	**5.49E-04**	6.81E-01	**5.00E-01**	**6.06E-01**	**6.21E-01**	9.249E+02	9.249E+02	9.249E+02
	150000	1.79E-06	**3.71E-04**	6.74E-01	**5.00E-01**	**5.83E-01**	**5.94E-01**	9.249E+02	9.249E+02	9.249E+02

Table 5: Comparison of the best ABC variants and CEC 2011 Algorithms for the problem I

Algorithm	Best	Mean
BABC	9.13E-04	2.88E+00
RABC	3.70E-06	1.12E+00
IABC	0.00E+00	1.59E+00
GA-MPC [29]	0.00E+00	0.00E+00
SAMODE [30]	0.00E+00	1.21E+00
ENSML_DE [31]	0.00E+00	1.78E+00
EA-DE-MA [32]	1.17E-11	2.09E+00
Adap.DE171 [33]	0.00E+00	3.85E+00
ED-DE [34]	0.00E+00	0.00E+00
OXCoDE [35]	0.00E+00	4.40E+00
DE-RHC [36]	5.02E-20	8.91E+00
RGA [37]	1.00E-04	9.29E+00
CDASA [38]	3.28E-18	1.01E+00
mSBX-GA [39]	6.79E-05	4.20E+00
DE-ACr [40]	7.21E-15	8.77E-01
WI_DE [41]	0.00E+00	3.28E+00
Mod_DE_LS [42]	3.00E-06	2.60E-05

Table 6: Comparison of the best ABC variants and CEC 2011 Algorithms for the problem II

Algorithm	Best	Mean
BABC	6.58E-01	7.39E-01
RABC	5.00E-01	5.03E-01
IABC	5.00E-01	5.57E-01
GA-MPC [29]	5.00E-01	7.48E-01
SAMODE [30]	5.00E-01	8.17E-01
ENSML_DE [31]	1.28E+00	1.42E+00
EA-DE-MA [32]	5.00E-01	5.28E-01
Adap.DE171 [33]	5.00E-01	5.00E-01
ED-DE [34]	5.19E-01	1.19E+00
OXCoDE [35]	5.00E-01	6.84E-01
DE-RHC [36]	9.51E-01	1.15E+00
RGA [37]	6.77E-01	9.65E-01
CDASA [38]	6.76E-01	9.39E-01
mSBX-GA [39]	6.79E-01	9.84E-01
DE-ACr [40]	6.66E-01	8.85E-01
WI_DE [41]	5.00E-01	6.56E-01
Mod_DE_LS [42]	7.23E-01	8.33E-01

algorithm modifies more than one dimension in the search equation that leads to a performance improvement for low dimensional hard problems. For high dimensional functions, the performance of ABC decreases

Table 4: Comparison results of ABC algorithms with default parameter values

		Problem I			Problem II			Problem III		
Algorithm	**FEs**	**Best**	**Median**	**Mean**	**Best**	**Median**	**Mean**	**Best**	**Median**	**Mean**
ABC	50000	**2.72E-04**	**2.30E-02**	**6.89E-01**	**5.63E-01**	**8.31E-01**	**8.18E-01**	**9.249E+02**	9.249E+02	9.249E+02
	100000	**7.59E-05**	**9.41E-03**	6.11E-01	**5.62E-01**	**8.19E-01**	**8.00E-01**	**9.249E+02**	9.249E+02	9.249E+02
	150000	**7.59E-05**	**6.54E-03**	4.65E-01	**5.62E-01**	**7.82E-01**	**7.80E-01**	**9.249E+02**	9.249E+02	9.249E+02
BABC	50000	2.21E-01	4.28E+00	4.30E+00	6.96E-01	**7.90E-01**	**7.94E-01**	**9.232E+02**	**9.232E+02**	9.237E+02
	100000	**1.84E-02**	3.70E+00	3.21E+00	6.76E-01	**7.35E-01**	**7.60E-01**	9.232E+02	**9.232E+02**	9.237E+02
	150000	**9.13E-04**	3.58E+00	2.88E+00	6.58E-01	**7.19E-01**	**7.39E-01**	9.232E+02	9.232E+02	9.237E+02
CABC	50000	**5.59E-04**	**3.98E-02**	1.10E+00	7.87E-01	**8.77E-01**	8.77E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
	100000	**5.46E-04**	**3.42E-02**	**6.34E-01**	7.33E-01	8.52E-01	8.51E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
	150000	**1.69E-04**	**1.09E-02**	5.83E-01	7.33E-01	8.00E-01	8.39E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
GbABC	50000	**1.28E-04**	**1.09E-02**	**5.17E-01**	**6.46E-01**	8.67E-01	8.52E-01	9.249E+02	**9.249E+02**	**9.249E+02**
	100000	1.28E-04	**6.57E-03**	4.54E-01	**6.46E-01**	8.41E-01	8.31E-01	9.249E+02	**9.249E+02**	**9.249E+02**
	150000	1.28E-04	**3.28E-03**	4.51E-01	**6.42E-01**	8.36E-01	8.24E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
GbdABC	50000	**4.48E-05**	**4.86E-03**	**2.94E-01**	7.11E-01	8.49E-01	8.40E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
	100000	**3.52E-05**	**7.60E-04**	1.31E-01	**6.14E-01**	8.21E-01	7.97E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
	150000	**3.33E-05**	**6.71E-04**	1.16E-01	**6.14E-01**	7.98E-01	7.88E-01	**9.249E+02**	**9.249E+02**	**9.249E+02**
IABC	50000	**2.23E-13**	**3.57E+00**	**3.21E+00**	***5.00E-01***	**5.32E-01**	**5.89E-01**	9.249E+02	9.249E+02	9.249E+02
	100000	***0.00E+00***	**2.91E+00**	**2.37E+00**	***5.00E-01***	**5.01E-01**	**5.63E-01**	9.249E+02	9.249E+02	9.249E+02
	150000	***0.00E+00***	**2.74E+00**	**1.59E+00**	***5.00E-01***	**5.00E-01**	**5.57E-01**	9.249E+02	9.249E+02	9.249E+02
IncABC	50000	3.46E-05	**2.75E+00**	**1.63E+00**	6.18E-01	**8.13E-01**	**8.06E-01**	9.249E+02	9.249E+02	9.249E+02
	100000	3.42E-05	**2.75E+00**	**1.62E+00**	**5.48E-01**	**7.92E-01**	**7.64E-01**	**9.249E+02**	9.249E+02	9.249E+02
	150000	3.40E-05	**2.75E+00**	**1.62E+00**	**5.44E-01**	**7.84E-01**	**7.56E-01**	**9.249E+02**	9.249E+02	9.249E+02
MABC	50000	**3.66E-04**	**3.19E-03**	1.06E+00	**5.93E-01**	**7.22E-01**	**7.19E-01**	9.249E+02	9.249E+02	9.249E+02
	100000	**3.66E-04**	**1.16E-03**	9.26E-01	**5.00E-01**	**6.77E-01**	**6.58E-01**	**9.249E+02**	9.249E+02	9.249E+02
	150000	**2.63E-04**	**1.08E-03**	7.38E-01	**5.00E-01**	**6.17E-01**	**6.27E-01**	**9.249E+02**	**9.249E+02**	9.249E+02
RABC	50000	8.47E-05	**6.89E-04**	1.26E+00	***5.00E-01***	**5.00E-01**	**5.21E-01**	**9.249E+02**	9.249E+02	9.249E+02
	100000	6.81E-06	6.80E-04	1.13E+00	***5.00E-01***	**5.00E-01**	**5.03E-01**	**9.249E+02**	9.249E+02	9.249E+02
	150000	3.70E-06	5.99E-04	1.12E+00	***5.00E-01***	**5.00E-01**	**5.03E-01**	**9.249E+02**	9.249E+02	9.249E+02

Table 7: Comparison of the best ABC variants and other algorithms for the problem III

Algorithm	**Best**	**Mean**
BABC	9.232E+02	9.237E+02
RABC	9.249E+02	9.249E+02
IABC	9.249E+02	9.249E+02
MSG-HS [28]	9.256E+02	9.269E+02
GA [43]	9.960E+02	-
GA-APO [43]	9.960E+02	-
NSOA [43]	9.849E+02	-
DE [44]	9.630E+02	-
PSO [45]	9.258E+02	9.264E+02
EP [46]	9.555E+02	9.577E+02
IEP [8]	9.536E+02	9.565E+02
TS [47]	9.565E+02	9.585E+02
TS-SA [46]	9.596E+02	9.629E+02
ITS [46]	9.691E+02	9.771E+02
SADE_ALM [48]	9.440E+02	9.548E+02

dramatically when modifying too many dimensions in search equation. In BABC, all dimensions are replaced by a guidence of the same dimension. This approach is very effective for some problems such as third problem or basic test problems but in most cases, a decrease of the performance can be observed on several engineering problems and shifted-rotated test functions. For RABC case, local search strategy effect on search behavior can be observed. Local search used in RABC is effective for the second problem but is not good for other problems. This behavior can be seen in benchmark functions presented in [12].

References

[1] E. Bonabeau, M. Dorigo, Theraulaz, G., Swarm intelligence: from natural to artificial systems, Oxford University Press, Inc., (1999).

[2] J. Kennedy and R. Eberhart, Particle swarm optimization, Neural Networks, Proceedings., IEEE International Conference on **4**, 1942-1948, (1995).

[3] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. thesis, Politecnico di Milano, Italy, (1992).

[4] X.S. Yang and S. Deb, Engineering optimisation by cuckoo search, International Journal of Mathematical Modelling and Numerical Optimisation **1(4)**, 330-343, (2010).

[5] S.K. Saha, S.P. Ghoshal, R. Kar, D. Mandal, Cat swarm optimization algorithm for optimal linear phase FIR filter design, ISA Transactions **52(6)**,781-794, (2013).

[6] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization **39(3)**, 459-471, (2007).
[7] R. S. Parpinelli, F.R. Teodoro, H.S., A comparison of swarm intelligence algorithms for structural engineering optimization, International Journal for Numerical Methods in Engineering **91(6)**, 666-684, (2012).
[8] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures **87(13-14)**, 861-870, (2009).
[9] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing **9(2)**, 625-631, (2009).
[10] S. Dhouib, S. Dhouib, H. Chabchoub, Artificial Bee Colony metaheuristic to optimize Goal Programming Engineering Design Problems, International Conference on Control, Decision and Information Technologies (CoDIT), 355-358, (2013).
[11] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing **8(1)**, 687-697, (2008).
[12] T. Liao, D. Aydın, T. Sttzle, Artificial bee colonies for continuous optimization: Experimental analysis and improvements, Swarm Intelligence **7(4)**, 327-356, (2013).
[13] D. Aydın and S. Özyön, Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search, Applied Soft Computing **13(5)**, 2456-2466, (2013).
[14] P. Balaprakash, M. Birattari, T. Stutzle, Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement, Hybrid Metaheuristics **4771**, T. Bartz-Beielstein, M. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, (ed.). Springer Berlin Heidelberg, 108-122, (2007).
[15] M. Birattari, Z. Yuan, P. Balaprakash, T. Stutzle, F-Race and Iterated F-Race: An Overview, Experimental Methods for the Analysis of Optimization Algorithms, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, (ed.). Springer Berlin Heidelberg, 311-336, (2010).
[16] B. Akay and D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences **192(1)**, 120-142, (2012).
[17] D. Karaboga, B. Gorkemli, C. Ozturk and N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review **42(1)**, 21-57, (2012).
[18] G. Zhu and S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation **217(7)**, 3166-3173, (2010).
[19] K. Diwold, A. Aderhold, A. Scheidler, M. Middendorf, Performance evaluation of artificial bee colony optimization and new selection schemes, Memetic Comp. **3(3)**, 149-162, (2011).
[20] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in Artificial Bee Colony algorithm, Applied Soft Computing **11(2)**, 2888-2901, (2011).
[21] W. Gao and S. Liu, Improved artificial bee colony algorithm for global optimization, Information Processing Letters **111(17)**, 871-882, (2011).
[22] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications **37(8)**, 5682-5687, (2010).
[23] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Information Sciences **181(16)**, 3508-3531, (2011).
[24] D. Aydın, T. Liao, M. Montes de Oca, T. Stutzle, Improving Performance via Population Growth and Local Search: The Case of the Artificial Bee Colony Algorithm, Artificial Evolution **7401**, J.-K. Hao, P. Legrand, P. Collet, N. Monmarch, E. Lutton, and M. Schoenauer, (ed.). Springer Berlin Heidelberg, 85-96, (2012).
[25] M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, The Computer Journal **7(2)**, 155-162, (1964).
[26] L. Tseng and C. Chen, Multiple trajectory search for Large Scale Global Optimization, Evolutionary Computation (CEC), 2008 IEEE Congress on, 3052-3059, (2008).
[27] S. Das and P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Technical Report, Jadavpur University, India and Nanyang Technological University, Singapore, (2010).
[28] C. Yaşar and S. Özyön, A new hybrid approach for nonconvex economic dispatch problem with valve-point effect, Energy **36(10)**, 5838-5845, (2011).
[29] S.M. Elsayed, R.A. Sarker, D.L. Essam, GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1034-1040, (2011).
[30] S.M. Elsayed, R.A. Sarker, D.L. Essam, Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1041-1048, (2011).
[31] R. Mallipeddi and P.N. Suganthan, Ensemble differential evolution algorithm for CEC2011 problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1557-1564, (2011).
[32] H.K. Singh and T. Ray, Performance of a hybrid EA-DE-memetic algorithm on CEC 2011 real world optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1322-1326, (2011).
[33] M. Asafuddoula, T. Ray, R. Sarker, An adaptive differential evolution algorithm and its performance on real world optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1057-1062, (2011).
[34] Y. Wang, B. Li, K. Zhang, Estimation of distribution and differential evolution cooperation for real-world numerical optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1315-1321, (2011).
[35] X. Li and Y. Minghao, Enhancing the Exploration Ability of Composite Differential Evolution through Orthogonal Crossover, Evolutionary Computation (CEC), 2011 IEEE Congress on, (2011).
[36] A. LaTorre, S. Muelas, J.M. Pena, Benchmarking a hybrid DE-RHC algorithm on real world problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1027-1033, (2011).
[37] A. Saha and T. Ray, How does the good old Genetic Algorithm fare at real world optimization?, Evolutionary

Computation (CEC), 2011 IEEE Congress on, 1049-1056, (2011).
[38] P. Korosec and J. Silc, The Continuous Differential Ant-Stigmergy Algorithm applied to real-world optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1327-1334, (2011).
[39] S. Bandaru, R. Tulshyan, K. Deb, Modified SBX and adaptive mutation for real world single objective optimization, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1335-1342, (2011).
[40] G. Reynoso-Meza, J. Sanchis, X. Blasco, J.M. Herrero, Hybrid DE algorithm with adaptive crossover operator for solving real-world numerical optimization problems, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1551-1556, (2011).
[41] U. Haider, S. Das, D. Maity, A. Abraham, P. Dasgupta, Self adaptive cluster based and weed inspired differential evolution algorithm for real world optimization, Evolutionary Computation (CEC), 2011 IEEE Congress on, 750-756, (2011).
[42] A. Mandal, A.K. Das, P. Mukherjee, S. Das, P.N. Suganthan, Modified differential evolution with local search algorithm for real world optimization, Evolutionary Computation (CEC), 2011 IEEE Congress on, 1565-1572, (2011).
[43] T.N. Malik, A. ul Asar, M.F. Wyne, S. Akhtar, A new hybrid approach for the solution of nonconvex economic dispatch problem with valve-point effects, Electric Power Systems Research **80(9)**, 1128-1136, (2010).
[44] S. Özyön, C. Yaşar, H. Temurtaş, Differential evolution algorithm approach to nonconvex economic power dispatch problems with valve point effect, 6th International Advanced Technologies Symposium (IATS'11), 181-186, (2011).
[45] S. Özyön, C. Yaşar, H. Temurtaş, Particle swarm optimization algorithm for the solution of nonconvex economic dispatch problem with valve point effect, Electrical and Electronics Engineering (ELECO), 2011 7th International Conference on, 101-105, (2011).
[46] W. Ongsakul and T. Tantimaporn, Optimal Power Flow by Improved Evolutionary Programming, Electric Power Components and Systems **34(1)**, 79-95, (2006).
[47] M.A. Abido, Optimal Power Flow Using Tabu Search Algorithm, Electric Power Components and Systems **30(5)**, 469-483, (2002).
[48] C. Thitithamrongchai and B. Eua-arporn, Self-adaptive differential evolution based optimal power flow for units with non-smooth fuel cost functions, J. Electrical Systems **3(2)**, 88-99, (2007).
[49] D. Aydin, Composite artificial bee colony algorithms: From component-based analysis to high-performing algorithms, Applied Soft Computing **32**, 266-285, (2015).

Bekir Afşar is a Research Assistant in Department of Computer Engineering at Muğla Sıtkı Koçman University, Muğla. He received the PhD degree in Computer Engineering at Ege University, Izmir. His research interests are in the areas of metaheuristics, continuous optimization, self-adaptive approaches and model-driven software development. He has published conference papers in area of model-driven development and metaheuristics.

Doğan Aydın is Assistant Professor of Computer Engineering at Dumlupınar University, Kütahya and also visiting researcher in IRIDIA at Universite Libr de Bruxelles, Brussels. He received the PhD degree in Computer Engineering at Ege University, Izmir. He is guest editor of two international journals and referee in several high-impact scientific journals in the frame of artificial intelligence and energy. He has published more than 20 papers in journals and conferences. His main research interests are: metaheuristics, continuous optimization, swarm intelligence, automatic parameter configuration and image processing.