Artificial bee colony algorithm for constrained optimization problems modified with multiple onlookers

Milos Subotic

***Abstract*—** Artificial bee colony (ABC) algorithm has been active research area recently and great number of modifications were suggested, both for unconstrained and constrained optimization problems. Our modification that is based on idea that in nature more than one onlooker bee goes to the promising food source is presented in this paper. In our approach the candidate solution in onlooker bee phase is formed using three solutions, while in the original ABC only one solution is used. Our modified algorithm is tested on the full set of 24 well known benchmark functions known as *g*–functions and proved to obtain better results than the pure ABC algorithm in majority of the test cases. The results are better both in the terms of quality and performance.

***Keywords*—**Artificial bee colony, Constrained optimization, Swarm intelligence, Metaheuristic optimization

I. Introduction

OVER the years many different technics for solving optimization problems were developed. Besides many traditional methods, heuristic methods become very prominent. Special place among heuristic methods belongs to the technics based on social behavior of certain animals and insects. These methods are known as swarm intelligence algorithms. Formally, a swarm can be defined as a group of (generally mobile) agents which communicate with each other (either directly or indirectly), by acting on their local environment. Within these groups, individuals are not aware of the global behavior of the group, nor do they have any information on the global environment. Bees' warming about their hive is an example of swarm intelligence [1]. Swarm intelligence is a heuristic method that models the population of entities that are able to self – organize and interact among them. Swarm intelligence refers to the problem-solving behavior that emerges from the interaction of such agents, and computational swarm intelligence refers to algorithmic models of such behavior [2]. Particle swarm optimization (PSO) and ant colony optimization (ACO) are two of the most representative swarm intelligence heuristics.

Particle swarm optimization (PSO) algorithm models social behavior of flock of birds or school of fish. It was introduced by Eberhart and Kennedy in 1995 [3]. Particle swarm optimization (PSO) is a global optimization algorithm for dealing with problems in which a best solution can be represented as a point or surface in an n-dimensional space [4]. Each particle moves through the search space influenced by their personal experience (it maintains a memory of the best solution found so far) and the experience of its neighbors.

Manuscript received December 25, 2011.

This research is supported by Ministry of Science, Republic of Serbia, Project No. III-44006

M. Subotic is with the Faculty of Computer Science, Megatrend University, Belgrade, Serbia, e-mail: milos.subotic@gmail.com

II. Artificial bee colony algorithm

Artificial bee colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee swarm.

In ABC system, artificial bees fly around in a multidimensional search space and some (employed and onlooker bees) choose food sources depending on their own experience and also their nest mates' experience and adjust their positions [5]. There are number of various optimization technics that simulate social life of real bees. Although there are several models based on honeybees [6], our modification is based on the artificial bee colony (ABC) algorithm. This model was initially proposed by Karaboga [7] and then lately formally introduced by Karaboga and Basturk [8]. ABC belongs to the group of algorithm which simulate foraging behavior.

The process of searching for nectar in flowers by honeybees can be observed as an optimization process. A colony of honey bees can fly in multiple directions simultaneously to exploit a large number of food sources [9]. The key elements in biological model of gathering food by honeybees are: food sources, employed collectors and unemployed collectors. The quality of a food source depends on many factors, such as the proximity to the hive, the concentration of food and how easy it is to extract it. In order to simplify representation of the profitability of a food source, it is possible to assign it a numerical value that is called fitness.

Employed collectors are associated with a particular food source which is exploited by them. Employed bees share information such as location and profitability of food source with the rest of the colony. Unemployed collectors are constantly looking for a food source to exploit. We can divide them into two groups: scout bees and onlooker bees. Scout bees are searching in the vicinity of the hive for new food sources. Onlooker bees are waiting in the hive and choosing a

food source based on the information shared by employed bees. Information about food sources is shared by the employed bees in a form of dance called waggle dance. Since dances of the most profitable sources have a longer duration, they are more likely to be observed by unemployed bees, increasing the probability of a collector bee choosing that food source.

When a food source is depleted, the bee or bees employed on it become unemployed and they have to decide between either becoming a scout bee and find another food source to exploit randomly or returning to the hive as onlooker bees and waiting for information about other food sources currently exploited.

An important difference between the ABC and other swarm intelligence algorithms is that in the ABC algorithm the possible solutions represent food sources (flowers), not individuals (honeybees). In other algorithms, like PSO, each possible solution represents an individual of the swarm. In the ABC algorithm the quality of solution is represented as fitness of a food source. Fitness is calculated by using objective function of the problem.

The whole colony of artificial bees is divided into 3 groups: employed bees, onlooker bees and scout bees. The number of employed bees is equal to the number of food sources and an employed bee is assigned to one of the sources. The exploitation process is performed by employer bees. Employed bee will generate a new solution (mutant solution) by using nearby food source and then retain the best solution (in a greedy selection). The number of onlooker bees is also the same as the number of employed bees and they are allocated to a food source based on their profitability. Like the employed bees, they calculate a new solution from its food source. They also carry out exploitation process. After certain number of cycles, if food source cannot be further improved, it is abandoned and replaced by randomly generated food source. This is called exploration process and it is performed by the third group of bees in the colony – scout bees. The solutions in the ABC algorithm are represented as food sources. The food sources are *D*-dimensional vectors (where *D* is the number of variables of the problem). Each one of the variables in the solution is associated with a range ($L_i \le x_i \le U_i$) , which must be considered when we randomly generate, with a uniform distribution, the initial solutions (food sources). L_i and U_i represent lower and upper limit of parameter x_i.

At the first step, a randomly distributed initial population is generated. The number of solutions equals *SN*, and the colony size is *2*SN*. Each solution is represented by a *D*-dimensional vector, where *D* is the number of optimization parameters. After initialization, the population is modified *MCN* times, where *MCN* is total number of iterations. The modifications are performed by employer bees, onlooker bees and scout bees. An employed bee modifies the solution in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new food source (new solution). Employer bee keeps in memory the solution with better fitness value. After search process is completed by all employer bees they share the information about food sources and nectar amount with the onlooker bees on the dance area throughout a waggle dance. An onlooker bee evaluates the nectar information collected from all employed bees and chooses a food source with a probability related to its nectar amount. Onlooker bee produces a mutant solution in the similar way as employed bee. She also keeps the solution with better fitness value. An onlooker bee chooses a food source depending on its probability value. Probability is calculated by using formula Eq. 1.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{1}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees. Equation 2 is used for production of mutant solution by employer and onlooker bees.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{2}$$

where $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes. Although *k* is determined randomly, it has to be different from *i*. $\varphi_{i,j}$ is a random number between [−1, 1]. *SN* is the number of solutions, and *D* is number of parameters of evaluation function. If a parameter value produced in this process exceeds its predetermined boundary value, it is set to boundary value.

$$x_i = \begin{cases} L_i, & if\ x_i < L_i \\ U_i, & if\ x_i > U_i \\ x_i, & otherwise \end{cases} \tag{3}$$

Where L_i and U_i are lower and upper limit of parameter x_i When food source is depleted, bees abandon it. In the ABC this happens after solution was not improved after certain number of cycles. This predetermined number of cycles is called limit for abandonment or just limit. Then the new food source is found. In ABC exploration process is carried out by scout bees. Scout bee produces a new solution randomly.

In ABC there are only three parameters to be modified: number of solutions, total number of iterations (cycles) and abandonment limit. Number of solutions (*SN*) represents the total number of solutions as well as the number of employer bees and number of onlooker bees. The colony size is 2*SN. Total number of iterations (*MCN*) represents max number of cycles.

Short pseudo – code of the ABC algorithm is:

Initialize the population of solutions
Evaluate the population
Produce new solutions for the employed bees
Apply the greedy selection process
Calculate the probability values
Produce the new solutions for the onlookers
Apply the greedy selection process

Send scout bees
Memorize the best solution achieved so far

III. Constrained optimization problems

The term optimization can be defined as:

- To make as perfect, effective, or functional as possible
- To make optimal; to get the most out of; to use best

In mathematics, to optimize means finding the best solution to a problem, where best is considered an acceptable (or satisfactory) solution, which must be absolutely better than a set of candidate solutions, or all candidate solutions. In applications, optimization is used in engineering and economics.

Constrained optimization is the minimization of an objective function subject to constraints on the possible values of the independent variable. Constraints can be either:

- equality constraints
- inequality constraints

Many real-world optimization problems require besides maximization (minimization) of objective function that certain constrains are satisfied. We can define the general constrained problem, without loss of generality, as:

Minimize*: f(x)*, subject to

$g_m(x) \leq 0,\ m = 1, \ldots, n_g$

$h_m(x) = 0,\ m = n_g + 1, \ldots, n_g + n_h$

$x_i \in dom(x_i)$

where n_g and n_h are the number of inequality and equality constraints respectively, and $dom(x_i)$ is the domain of the variable x_i. These constraints often limit feasible solution space to a small subset [10].

Given a point x in the feasible region, a constraint $g_i(x) \geq 0$ is called active at x if $g_i(x) = 0$ and inactive at x if $g_i(x) < 0$. Equality constraints are always active. The active constraints are particularly important in optimization theory as they determine which constraints will influence the final result of optimization problem.

For unconstrained optimization problems the greedy selection is used for keeping better solution after employed bee and onlooker phase. Our multiple onlooker modification of ABC algorithm, as well as original ABC algorithm, uses Deb's rule [11] to deal with constrained optimization problems. Deb's method consists of three very simple heuristic rules. It uses a tournament selection operator, where two solutions are compared at a time using the following criteria:

1. Any feasible solution is preferred to any infeasible solution
2. Among two feasible solutions, the one having better objective function value is preferred
3. Among two infeasible solutions, the one having smaller constraint violation is preferred

It is very time consuming process to create feasible initial population, and for some optimization problems it is not even possible to initialize feasible solutions using random numbers. Hence, nor ABC algorithm, nor our proposed modification, consider the initial population to be feasible.

IV. Multiple onlooker modification of the ABC algorithm

It is observed that real onlooker bees in nature go to the food source that is marked as promising by several employed bees. In the original ABC algorithm mutant solution is produced using a solution from one employed bee. Hence if that solution is very far from optimum, the new mutated solution will be probably very far from optimum solution too, even if it is better than solution from employed bee. But if there are a several employed bees, the influence of one solution is smaller. Original ABC algorithm uses the Eq. 2 to produce candidate solution in onlooker phase.

In real life onlookers are going to areas where more than one employed bee has found promising food source. Our modification uses three employed bees to create mutant solution in onlooker bee phase. We conducted the experiments with different number of solutions that are used in forming a mutant solution, but for 20 food sources (colony size 40) that we have used in this paper best results are obtained with three solutions. Results obtained from these experiments for various colony sizes are showed on next graph. Best solution is presented as 100%; other solutions are presented as percentage of best solution. The experiments are performed for 1 to 5 solutions that are forming candidate solution, for 20, 40 and 100 solutions in total and for all test functions. Y axis shows the percentage of best solution, while X axis shows the number of solution that participates in forming a mutant solution. For totals of 20 and 40 solutions best results are obtained when mutant solution is formed from three solutions, while best results for total of 100 solutions are obtained when four solutions forms new solution.

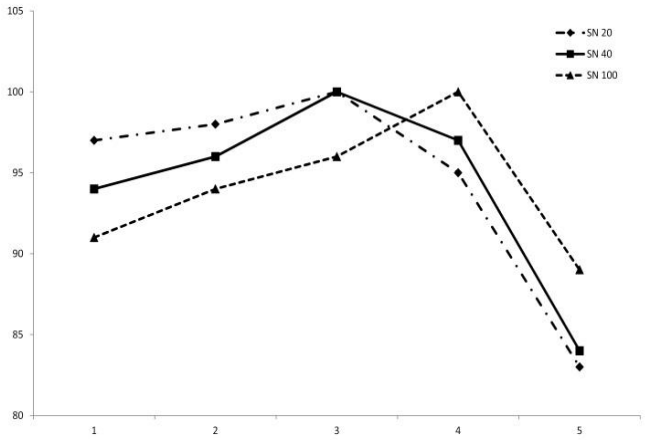

Fig. 1. comparison of results obtained for various number of solutions that forms mutant solution

When colony size is increased, the number of solution that are participating in forming a mutate solution used for

obtaining best results increase too.

In multiple onlooker approach the solutions have to be sorted by probability before onlooker bee phase. The bees in nature are coming from same area, so when solutions are sorted, the mutant solution is formed from solutions that are have the smallest distance between themselves in all set of solutions. Our modification of ABC algorithm uses next expression to calculate the parameter for candidate solution:

$$v_{ij} = x_{ij} + a_1\varphi_{ij}(x_{ij} - x_{k-1j}) + \\ + a_2\varphi_{ij}(x_{ij} - x_{kj}) + a_3\varphi_{ij}(x_{ij} - x_{k+1j}) \quad (4)$$

where v_{ij} is a parameter of mutant solution, x_{ij} is a parameter of the current solution and $x_{k-1,j}$, x_{kj}, $x_{k+1,j}$ are parameters of three neighbor solutions sorted by probability. It is obvious that i must be different from k, $k - 1$, and $k + 1$; a_1, a_2 and a_3 are quotients that show how much influence on the new solution one particular solution has. In multiple onlooker approach three neighbor solutions do not participate equally in forming a mutant solution. Our empirical experiments shows that best results are obtained when $a_1 = 0.3$, $a_2 = 0.4$ and $a3 = 0.3$. Strongest influence has the middle neighbor solution, thus it has the largest factor of 0.4. Other two solutions have equal factors of 0.3.

Here we present the steps of MO-ABC algorithm. The first step is initialization of populations by using Eq. 5 and evaluation of solutions.

$$X_{ij} = l_j + rand\,(0,1) * (u_j - l_j) \quad (5)$$

After evaluation, next steps are repeated MCN times where MCN is maximum number of cycles. In every iteration of this loop new solutions for employed bees are being produced by using Eq. 2 and evaluated. Better solution is chosen between current and mutant by using Deb's rule. Probability for each food source is calculated by using Eq. 1. After calculation of probabilities for every solution, sorting the solutions by probability is executed. Then for each onlooker bee, production a new solution vi is performed by using Eq. 6.

$$v_{ij} = \begin{cases} x_{ij} + a_1\varphi_{ij}(x_{ij} - x_{k-1j}) + a_2\varphi_{ij}(x_{ij} - x_{kj}) \\ \quad +a_3\varphi_{ij}(x_{ij} - x_{k+1j}),\ if\ Rj < MR \\ x_{ij},\ otherwise \end{cases} \quad (6)$$

where v_{ij} is a parameter of mutant solution, x_{ij} is a parameter of the current solution and $x_{k-1,j}$, x_{kj}, $x_{k+1,j}$ are parameters of three neighbor solutions sorted by probability. R_j is randomly chosen real number in the range [0,1] and $j \in \{1, 2,...,D\}$, a_1, a_2 and a_3 are quotients that are showing how much influence on the new solution one particular solution has. MR, modification rate, is a control parameter that controls whether the parameter x_{ij} will be modified or not. This is a new parameter that Karaboga and Basturk added to the ABC algorithm for constrained optimizations [12]. Our experiments showed that using of this parameter helped in achieving better results compared with a version without this parameter. Therefore, as in ABC for constrained optimization, this parameter was included in our modification of ABC algorithm. Then new solutions are evaluated. Again, better solution is chosen by using Deb's rule. In scout bee phase abandoned solutions are found by using "limit" parameter and then replaced with a new randomly produced solution by using Eq. 4. Best solution achieved so far is memorized and the algorithm goes at the first step of the loop. Short pseudo-code of the MO-ABC is given below:

```
Evaluate the population
  cycle = 1
  repeat
    Produce new solutions for employers by using
      Eq. 2 and evaluate them
    Choose better solution
    Calculate probabilities by using Eq. 1.
    Sort solutions by probability
    Produce new solutions for onlookers by using
      Eq. 5 and evaluate them
    Choose better solution
    Send scout bees
    Memorize the best solution achieved so far
    cycle = cycle + 1
until cycle = MCN
```

V. Experimental Results and Discussion

Multiple onlooker ABC algorithm (MO-ABC) is compared to the original ABC algorithm. Settings for MO-ABC are given in Table 1.

TABLE I
CONTROL PARAMETERS FOR MO – ABC

Parameter	Symbol	Value
Solutions Number	SN	20
Total number of cycles	MCN	6000
Limit	*limit*	MCN / (2 * SN) = 150
Modification Rate	MR	0.8
Weight quotient for 1st neighbor solution	a_1	0.3
Weight quotient for 2nd neighbor solution	a_2	0.4
Weight quotient for 3rd neighbor solution	a_3	0.3

Table 2 contains a summary of the main characteristics of the test problems. Table 3 shows comparison of the first 13 benchmark functions published in [12] and Table 4 shows comparison for functions g_{14}–g_{24}. Experiments were repeated 30 times each using random population with different seeds. Better results in Tables 3 and 4 are bolded. To evaluate the performance of the proposed algorithm and to make a comparison with original ABC algorithm, we used the benchmark constrained optimization functions proposed in [13]. This set of benchmark function illustrates well different types of numerical optimization problems.

TABLE II
CHARACTERISTICS OF TEST FUNCTIONS

f	n	Type of f.	ρ (%)	li	ni	le	ne	a
g01	13	quad.	0.0003	9	0	0	0	6
g02	20	nonl.	99.9973	2	0	0	0	1
g03	10	nonl.	0.0026	0	0	0	1	1
g04	5	quad.	27.0079	4	2	0	0	2
g05	4	nonl.	0.0000	2	0	0	3	3
g06	2	nonl.	0.0057	0	2	0	0	2
g07	10	quad.	0.0000	3	5	0	0	6
g08	2	nonl.	0.8581	0	2	0	0	0
g09	7	nonl.	0.5199	0	4	0	0	2
g10	8	linear	0.0020	6	0	0	0	6
g11	2	quad.	0.0973	0	0	0	1	1
g12	3	quad.	4.7697	0	1	0	0	0
g13	5	nonl.	0.0000	0	0	1	2	3
g14	10	nonl.	0.0000	0	0	3	0	3
g15	3	quad.	0.0000	0	0	1	1	2
g16	5	nonl.	0.0204	4	34	0	0	4
g17	6	nonl.	0.0000	0	0	0	4	4
g18	9	quad.	0.0000	0	13	0	0	6
g19	15	nonl.	33.4761	0	5	0	0	0
g20	24	linear	0.0000	0	6	2	12	16
g21	7	linear	0.0000	0	1	0	5	6
g22	22	linear	0.0000	0	1	8	11	19
g23	9	linear	0.0000	0	2	3	1	6
g24	2	linear	79.6556	0	2	0	0	2

Parameter *n* denotes the number of parameters. The function can be linear, nonlinear (nonl.) or quadratic (quad.), *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 inequality constraints, *a* is the number of active restrictions and ρ is a percentage of the feasible area. A percentage of feasible area is:

$$\rho = |F| / |S| \quad (7)$$

where |F| is the number of feasible solutions and |S| is the total number of solutions randomly generated. Michalewicz and Schoenauer [13] suggested a total number of 1,000,000 solutions for |S|.

It is shown in Table 3, that results obtained using MO-ABC are better than results obtained by original ABC algorithm for constrained optimization problems. The *g02* function illustrates that due to greater exploration power of MO-ABC algorithm, better best results are reached, but also the worst result is slightly worse than result from original ABC. The standard deviation for *g02* function is somewhat inferior for the same reason. MO-ABC reaches much better results for *g13* function then the original ABC algorithm.

TABLE III
COMPARISON OF RESULTS OBTAINED BY ORIGINAL ABC AND MO-ABC FOR G01 – G13 FUNCTIONS

Function Optimum		Method	
		ABC	MO-ABC
g1 -15,000	Best	-15.000	-15.000
	Mean	-15.000	-15.000
	Worst	-15.000	-15.000
	Std. Dev.	0.000	0.000
g2 -0,803619	Best	-0.803598	**-0.803605**
	Mean	-0.792412	**-0.793506**
	Worst	**-0.749797**	-0.744311
	Std. Dev.	**0.012**	0.014
g3 -1,0005001	Best	-1	-1
	Mean	-1	-1
	Worst	-1	-1
	Std. Dev.	0.000	0.000
g4 -30665,538672	Best	-30665.539	-30665.539
	Mean	-30665.539	-30665.539
	Worst	-30665.539	-30665.539
	Std. Dev.	0.000	0.000
g5 5126,496714	Best	**5126.484**	5126.582
	Mean	5185.714	**5162.496**
	Worst	5438.387	**5229.134**
	Std. Dev.	75.358	**4,78E01**
g6 -6961,813875	Best	-6961.814	-6961.814
	Mean	-6961.814	-6961.814
	Worst	-6961.805	**-6961.814**
	Std. Dev.	0.002	**0.000**
g7 24,306209	Best	24.330	**24.329**
	Mean	24.473	**24.444**
	Worst	25.190	**24.940**
	Std. Dev.	0.186	**0.137**
g8 -0,095825	Best	0.095825	0.095825
	Mean	0.095825	0.095825
	Worst	0.095825	0.095825
	Std. Dev.	0.000	0.000
g9 680,6300573	Best	680.634	**680.630**
	Mean	680.640	**680.632**
	Worst	680.653	**680.638**
	Std. Dev.	0.004	**0.002**
g10 7049,2480205	Best	7053.904	**7053.404**
	Mean	7224.407	**7167.873**
	Worst	7604.132	**7418.313**
	Std. Dev.	133.870	**83.002**
g11 0,7499	Best	0.750	0.750
	Mean	0.750	0.750
	Worst	0.750	0.750
	Std. Dev.	0.000	0.000
g12 -1	Best	-1.000	-1.000
	Mean	-1.000	-1.000
	Worst	-1.000	-1.000
	Std. Dev.	0.000	0.000
g13 0,0539415	Best	0.760	**0.445**
	Mean	0.968	**0.465**
	Worst	1.000	**0.490**
	Std. Dev.	0.055	**0.023**

TABLE IV
COMPARISON OF RESULTS OBTAINED BY OUR IMPLEMENTATION OF ORIGINAL ABC AND MO-ABC FOR G14 – G24 FUNCTIONS

Function Optimum		Method	
		ABC	MO-ABC
g14 -47,764	Best	No feasible solutions found	**-46.450858**
	Mean		**-45.998005**
	Worst		**-45.316806**
	Std. Dev.		**0.265**
g15 961,715	Best	961.715	961.715
	Mean	961.899	961.876
	Worst	964.508	964.345
	Std. Dev.	0.557	0.543
g16 -1,905155	Best	-1.905155	-1.905155
	Mean	-1.905155	-1.905155
	Worst	-1.905155	-1.905155
	Std. Dev.	0.000	0.000
g17 8853,539	Best	**8906.443**	8939.010
	Mean	9036.270	**8946.173**
	Worst	9225.281	**8956.243**
	Std. Dev.	124.023	**9.548**
g18 -0,866025	Best	-0.865809	**-0.865958**
	Mean	-0.762018	**-0.767066**
	Worst	-0.663042	**-0.670531**
	Std. Dev.	0.094	0.094
g19 32,655	Best	34.196	**33.778**
	Mean	35.863	**35.315**
	Worst	37.613	**37.373**
	Std. Dev.	**0.686**	0.690
g20 No feasible solution	Best	No feasible solutions found	No feasible solutions found
	Mean		
	Worst		
	Std. Dev.		
g21 193,724	Best	**287.253415**	329.120
	Mean	473.151297	**329.438**
	Worst	987.385078	**329.757**
	Std. Dev.	250.094567	**0.451**
g22 236,430	Best	No feasible solutions found	No feasible solutions found
	Mean		
	Worst		
	Std. Dev.		
g23 -400,055	Best	No feasible solutions found	No feasible solutions found
	Mean		
	Worst		
	Std. Dev.		
g24 -5,508013	Best	-5.508013	-5.508013
	Mean	-5.508013	-5.508013
	Worst	-5.508013	-5.508013
	Std. Dev.	6,64E-10	**0.000**

Exploration capabilities of MO-ABC are even more obvious by looking at *g14* function. While original ABC cannot find any feasible solution, MO-ABC reaches feasible solution near optimum solution. MO-ABC is also incapable to find solutions for *g20*, *g22* and *g23* functions. It is due to their very small feasible area. The results for *g01*, *g03*, *g04*, *g06*, *g07*, *g08*, *g09*, *g11*, *g12*, *g15*, *g16*, *g18* and *g24* are very close to their optimum solutions, hence there is no much space for improvements. MO-ABC obtains better best results for 8 functions, while original ABC obtains better results for 3 functions. MO-ABC reaches 11 better mean results, while ABC never achieves better mean result than MO-ABC. Also MO-ABC obtains better worst results for 11 test functions, while original ABC finds better worst results for only 1 benchmark function. Standard deviation (Std. Dev.) obtained by MO-ABC is better for 10 benchmark functions, while standard deviation obtained by ABC is better for 2 functions. Table 5 shows a quick summary of results from tables 3 and 4.

TABLE V
QUICK SUMMARY OF RESULTS

Category	Method	
	ABC	MO-ABC
Best	3	8
Mean	0	11
Worst	1	11
Std. Dev.	2	10

VI. PERFORMANCE MEASURES

After the quality of results has been compared in previous section, in this section we are illustrating performance of our modified ABC algorithm, MO-ABC. First some terms will be denoted and explained for easier understanding of performance measuring.

Evaluation: Represents calculation of value of the objective function and the values of the constraints for one solution. The total number of evaluations of an algorithm is an important measure of computational cost.

Feasible solution: Represents a solution that satisfies all the constraints of the optimization function.

Feasible run: Represents a run of the algorithm where at least one solution is feasible.

Successful solution: Represents a solution that it is equal or better to the best known value for that test function.

Successful run: Represents a run of the problem where at least one solution is a successful solution.

ANESS: Represents the average number of evaluations needed to found a successful solution. It is calculated with the following formula:

$$\frac{\sum evaluations\ needed\ to\ found\ a\ successful\ solution}{number\ of\ successful\ runs} \quad (8)$$

In this performance measure low values are desired.

PSR: Represents the percentage of successful runs, calculated by the next formula:

$$\frac{number\ of\ successful\ runs}{total\ number\ of\ runs} \quad (9)$$

In this performance measure high values are desired.

PFR: It is the percentage of feasible runs, calculated by the formula:

$$\frac{number\ of\ feasible\ runs}{total\ number\ of\ runs} \quad (10)$$

In this performance measure high values are desired.

EVALS: Proposed by Lampinen in [14], EVALS represents the number of evaluations needed to found the first feasible

solution in every run of the algorithm. In this performance measure the best, worst, mean and standard deviation are reported. In this performance measure low values are desired.

Comparing is made between our implementation of ABC algorithm and MO-ABC, since there is no performance analysis of original ABC for constrained optimization problems published.

The results of ANESS, PSR and PFR are divided into two tables, hence it makes comparing easier to follow. These results are shown in tables 6 and 7.

TABLE VI
COMPARING ANESS, PSR AND PFR FOR FUNCTION G01-G15

Function		Method	
		ABC	MO-ABC
g1	ANESS	**12398.468**	15293.625
	PSR	1	1
	PFR	1	1
g2	ANESS	-	**80324.500**
	PSR	0	**0.800**
	PFR	1	1
g3	ANESS	**143567.345**	153238.325
	PSR	0.033	**1**
	PFR	0.9	**1**
g4	ANESS	42567.667	**41266.765**
	PSR	1	1
	PFR	1	1
g5	ANESS	-	-
	PSR	0	0
	PFR	0.7	**1**
g6	ANESS	143638.333	**98756.234**
	PSR	0.633	**1**
	PFR	1	1
g7	ANESS	-	-
	PSR	0	0
	PFR	0.9	**1**
g8	ANESS	1321.543	**1211.333**
	PSR	1	1
	PFR	1	1
g9	ANESS	-	**123727.366**
	PSR	0	**0.666**
	PFR	1	1
g10	ANESS	-	-
	PSR	0	0
	PFR	1	1
g11	ANESS	179654.756	**165734.763**
	PSR	0.033	**1**
	PFR	1	1
g12	ANESS	**1193.645**	1225.723
	PSR	1	1
	PFR	1	1
g13	ANESS	-	-
	PSR	0	0
	PFR	0.800	**1**
g14	ANESS	-	-
	PSR	0	0
	PFR	0	**0.667**
g15	ANESS	234811.456	**189766.345**
	PSR	0.7	**1**
	PFR	0.800	**1**

Table 6 shows results for functions g01-g15, while Table 7 presents results for benchmark functions g16-g24. Multiple onlooker ABC obtains smaller average number of function evaluation required to find successful solution for benchmark functions g02, g04, g06, g08, g09, g11, g15, g18 and g24, while original ABC has lower ANESS values for g01, g03, g12 and g16. This indicates higher convergence speed of MO-ABC algorithm compared to original ABC algorithm. The percentage of successful runs is higher for functions g02, g03, g06, g09, g11, g15 and g18 when MO-ABC is used. Original ABC algorithm is always inferior or equal in terms of percentage of successful run compared to our MO-ABC algorithm.

This result illustrates the consistency of algorithm. MO-ABC shows greater consistency compared to ABC algorithm. PFR shows the ability of algorithm to find feasible solutions. MO-ABC has higher PFR for g03, g05, g07, g13, g14, g15, g17, g19 and g21 test functions, while ABC never reaches higher PFR than MO-ABC.

TABLE VII
COMPARING ANESS, PSR AND PFR FOR FUNCTION G16-G24

Function		Method	
		ABC	MO-ABC
g16	ANESS	**32557.711**	33876.543
	PSR	1	1
	PFR	1	1
g17	ANESS	-	-
	PSR	0	0
	PFR	0.1	**1**
g18	ANESS	213635.333	**87455.111**
	PSR	0.033	**0.333**
	PFR	1	1
g19	ANESS	-	-
	PSR	0	0
	PFR	0.9	**1**
g20	ANESS	-	-
	PSR	0	0
	PFR	0	0
g21	ANESS	-	-
	PSR	0	0
	PFR	0.033	**0.166**
g22	ANESS	-	-
	PSR	0	0
	PFR	0	0
g23	ANESS	-	-
	PSR	0	0
	PFR	0	0
g24	ANESS	6877.333	**6656.446**
	PSR	1	1
	PFR	1	1

Tables 8 and 9 show the result of measuring EVALS performance parameter. The results are divided into two tables for easier analysis. Table 8 presents EVALS results of g01 – g12 functions while table 9 shows results for g13-g24 functions.

Our MO-ABC reached better best values of EVALS then original ABC algorithm for test problems g02, g03, g06, g07,

g09, g10, g11, g12, g14, g15, g17, g18, g19, g21 and g24, while original ABC reached better results for g01, g04, g05, g08, g13 and g16 benchmark problems. This indicates greater ability of MO-ABC to find feasible solutions. MO-ABC has better mean results for functions g02, g03, g05, g07, g10, g11, g12, g14, g15, g16, g17, g18, g19, g21 and g24. ABC has better mean results for g01, g04, g06, g08 and g09 test functions.

TABLE VIII
COMPARING EVALS PARAMETER FOR FUNCTION G01-G12

Function		Method	
		ABC	MO-ABC
g1	Best	**79**	245
	Mean	**360.333**	654.422
	Worst	**649**	956
	Std. Dev.	154.887	**149.554**
g2	Best	24	**19**
	Mean	26.765	**19**
	Worst	31	**19**
	Std. Dev.	2.657	**0**
g3	Best	188563	**187456**
	Mean	189664.321	**189544.870**
	Worst	198332	**196688.007**
	Std. Dev.	2112.537	**2041.440**
g4	Best	**22**	23
	Mean	**23.366**	24.580
	Worst	35	**34**
	Std. Dev.	2.906	**2.607**
g5	Best	**181446**	185997
	Mean	200324.450	**199003.008**
	Worst	213887	**205876**
	Std. Dev.	12007.555	**8790.345**
g6	Best	145	**142**
	Mean	**534.876**	550.785
	Worst	**987**	1203
	Std. Dev.	**197.378**	234.667
g7	Best	345	**334**
	Mean	699.087	**678.876**
	Worst	1002	**992**
	Std. Dev.	201.320	**198.345**
g8	Best	**27**	29
	Mean	**98.800**	102.333
	Worst	**201**	205
	Std. Dev.	**41.256**	45.341
g9	Best	24	**23**
	Mean	**126.876**	139.504
	Worst	**271**	287
	Std. Dev.	**56.870**	61.008
g10	Best	608	**345**
	Mean	1298.058	**1034.876**
	Worst	1997	**1455**
	Std. Dev.	503.854	**432.850**
g11	Best	192668	**187234**
	Mean	193908.554	**191991.775**
	Worst	194998	**193398**
	Std. Dev.	130.618	**127.665**
g12	Best	24	**22**
	Mean	41.922	**34**
	Worst	65	**57**
	Std. Dev.	13.089	**8.903**

Standard deviation of EVALS parameter is better for g01, g02, g03, g04, g05, g07, g10, g11, g12, g14, g15, g16, g17, g18, g21 and g24 when MO-ABC is used. ABC reaches better standard deviation values for g06, g08, g19 test functions. Better mean and standard deviation values indicate a better consistency of MO-ABC algorithm compared to original ABC algorithm.

TABLE IX
COMPARING EVALS PARAMETER FOR FUNCTION G13-G24

Function		Method	
		ABC	MO-ABC
g13	Best	**178976**	182938
	Mean	**181654.197**	183118
	Worst	**183002**	183854
	Std. Dev.	3612.916	**34.619**
g14	Best	-	**218034**
	Mean	-	**223902.934**
	Worst	-	**225987**
	Std. Dev.	-	**1836.093**
g15	Best	189444	**171982**
	Mean	194921.610	**178095.054**
	Worst	201823	**182931**
	Std. Dev.	1929.056	**432.098**
g16	Best	**110**	139
	Mean	715.765	**705.191**
	Worst	**1269**	1298
	Std. Dev.	303.831	**299.865**
g17	Best	194065	**168928**
	Mean	204987.519	**178597.454**
	Worst	206456	**183186**
	Std. Dev.	9294.086	**8187.416**
g18	Best	939	**898**
	Mean	1543.865	**1478.875**
	Worst	**1876**	1902
	Std. Dev.	187.867	**185.826**
g19	Best	22	**21**
	Mean	24.961	**24.591**
	Worst	**35**	36
	Std. Dev.	**3.710**	3.960
g20	Best	-	-
	Mean	-	-
	Worst	-	-
	Std. Dev.	-	-
g21	Best	189007	**186527**
	Mean	198408.717	**195587.046**
	Worst	203981	**202581**
	Std. Dev.	8012.851	**7988.804**
g22	Best	-	-
	Mean	-	-
	Worst	-	-
	Std. Dev.	-	-
g23	Best	-	-
	Mean	-	-
	Worst	-	-
	Std. Dev.	-	-
g24	Best	29	**27**
	Mean	34.800	**33.916**
	Worst	39	**38**
	Std. Dev.	3.523	**2.918**

MO-ABC reaches better worst results for functions g01, g02, g03, g04, g05, g07, g10, g11, g12, g13, g14, g15, g16, g17, g18, g21 and g24 test functions, while original ABC obtains better worst results for g06, g08, g09 and g19 test functions. This is due to ability of MO-ABC to focus on promising areas of search space.

VII. Conclusion

This paper presents multiple onlookers modification of the artificial bee colony algorithm for constrained optimization problems. The performance of MO-ABC algorithm is tested on 24 well-known constrained optimization benchmark functions and compared with results obtained by the original ABC algorithm. It is shown that our modification of ABC algorithm for constrained optimization problems can handle tested functions very well. This indicates a potential practical usage since many real life problems are constrained problems. Also from these results it can be seen that MO-ABC has more exploration power compared to the original ABC. In 8 out of 24 test functions, better results are obtained. Experiments performed in this paper show that modified ABC (MO-ABC) has greater ability to find feasible solutions, due to its focus on promising search space. Multiple onlooker approach showed greater consistency compared to original ABC algorithm.

References

[1] L. Jiann-Horng, H. Li-Ren: Chaotic bee swarm optimization algorithm for path planning of mobile robots, Proceedings of the 10th WSEAS international conference on evolutionary computing, 2009, pp. 84-89.

[2] A. P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, Wiley and Sons, 2005, pp. 615

[3] Kennedy, J., Eberhart, R.C.: Particle swarm optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks, 1995, vol. 4, pp. 1942–1948,

[4] N. Gomez, L. F. Mingo, J. Bobadilla, F. Serradilla, J. A. C. Manzano: Particle Swarm Optimization models applied to Neural Networks using the R language, WSEAS Transactions on Systems, Volume 9, Issue 2, 2010, pp. 192-202

[5] L. Jiann-Horng, L. Meei-Ru, H. Li-Ren: A novel bee swarm optimization algorithm with chaotic sequence and psychology model of emotion, Proceedings of the 9th WSEAS International Conference on Systems Theory and Scientific Computation 2009, pp. 87-92.

[6] A. Baykasoğlu, L. Özbakır, P. Tapkan, Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem, Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, ISBN: 978-3-902613-09-7, 2007, pages 532, pp. 113-144

[7] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey, 2005, http://mf.erciyes.edu.tr/abc/publ.htm

[8] B. Basturk, D. Karaboga. An artificial bee colony (ABC) algorithm for numeric function optimization, Applied Soft Computing, Vol. 8, Issue 1, 2008, pp. 687-697

[9] R. Mohamad Idris, A. Khairuddin and M.W. Mustafa, Optimal Allocation of FACTS Devices in Deregulated Electricity Market Using Bees Algorithm, WSEAS Transactions on Power Systems, Vol. 5, Issue 2, Apr 2010, pp. 108-119.

[10] R. M. Gamot, A. Mesa: Particle swarm optimization: Tabu search approach to constrained engineering optimization problems, WSEAS Transactions on Mathematics, Vol.7, 2008, pp. 666-675.

[11] K. Deb, An Efficient Constraint Handling Method for Genetic Algorithms, Computer Methods in Applied Mechanics and Engineering, Vol. 186, No. 2-4, 2000, pp. 311–338.

[12] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, 2007, pp. 789–798

[13] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation, 1996, Vol. 4, No. 1, pp. 1–32

[14] J . Lampinen, A constraint handling approach for the differential evolution algorithm, Proceedings of the Congress on Evolutionary Computation 2002, volume 2, pp. 1468–1473

Milos Subotic received B.S. in computer science in 2010 from Advanced School of Electrical and Computer Engineering, Belgrade, Serbia and also B.S. in economics in 2006 from Megatrend University of Belgrade.

He is currently Ph.D. student at Faculty of Mathematics, Computer science department, University of Belgrade and works as teaching assistant at Faculty of Computer Science, Megatrend University of Belgrade. He is the author or coauthor of five papers. His current research interest includes nature inspired metaheuristics.

Mr. Subotic participated in WSEAS conferences.