Performance of the improved artificial bee colony algorithm on standard engineering constrained problems

Ivona Brajevic, Milan Tuba, and Milos Subotic

***Abstract*—**Artificial bee colony (ABC) algorithm is successfully used for many hard, mostly continuous, optimization problems. There is a way to extend standard ABC algorithm to constrained problems. In this paper an improved version of the artificial bee colony algorithm adjusted for constrained optimization problems is presented. It uses Deb's rule. This modified algorithm has been implemented and tested on four standard engineering constrained benchmark problems which contain discrete and continuous variables. Our results were compared to the results obtained by simple constrained particle swarm optimization algorithm (SiC-PSO) which showed a very good performance when it was applied to the same problems. Our results are of the comparable quality with faster convergence.

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

I. Introduction

CONSTRAINED optimization problems have numerous applications. Engineering design is one of the scientific fields in which constrained optimization problems frequently arise [1]. These types of problems normally have mixed (continuous and discrete) design variables, nonlinear objective functions and nonlinear constrains. Constrains are very important in engineering design problems. They are usually imposed in the statement of the problems and sometimes are very hard to satisfy, which makes the search difficult and inefficient.

Different deterministic as well as stochastic algorithms have been developed for solving constrained optimization problems. Deterministic approaches such as sequential quadratic programming methods and generalized reduced gradient methods [2], [3], [4] are inflexible to adapt the solution algorithm to a given problem. Generally a given problem is modeled in such a way that a classical algorithm can handle it [5]. This often requires making several assumptions which might not be easy to justify in many situations. Therefore their applicability is limited. On the other hand, stochastic optimization algorithms such as genetic algorithms, simulated annealing algorithms, evolution strategies, evolutionary programming and particle swarm optimization (PSO) do not make such assumptions and they have been successfully applied for solving constrained optimization problems during the past few years [1], [6], [7], [8], [9], [10].

Karaboga has described an artificial bee colony (ABC) algorithm based on the foraging behavior of honey bees for numerical optimization problems [11]. Karaboga and Basturk have compared the performance of the ABC algorithm with the performance of other well-known modern heuristic algorithms such as genetic algorithm (GA), differential evolution (DE), particle swarm optimization on unconstrained and constrained problems [12], [13]. It has been shown that the ABC algorithm can be efficiently used for solving unconstrained and constrained optimization problems. In this work, our approach to the ABC algorithm for constrained optimization problems, called SC-ABC (Simple Constrained ABC), was applied to real engineering problems existing in the literature and its performance was compared with the performance of Simple Constrained Particle Swarm Optimizer (SiC-PSO) [1]. SiC-PSO algorithm showed a very good performance when it was applied to several engineering design optimization problems.

This paper is organized as follows. Section 2 describes the ABC algorithm for unconstrained and constrained problems. Section 3 presents our proposed approach. Section 4 describes four benchmark problem formulations. Section 5 presents the experimental setup adopted and provides an analysis of the results obtained from our empirical study. Conclusions and some plans for future research are provided in Section 6.

II. Artificial Bee Colony Algorithm

A. The ABC Algorithm Used for Unconstrained Optimization Problems

In ABC algorithm [11], [12], [13], [14], [15] for the unconstrained optimization, the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. One half of the colony consists of the employed artificial bees and the other half includes the onlookers. All

Manuscript received January 30, 2011.

The research was supported by the Ministry of Science, Republic of Serbia, Project No. III 44006

M. Tuba is with the Faculty of Computer Science, Megatrend University, Belgrade, Serbia, e-mail: tuba@ieee.org

I. Brajovic is with the Faculty of Mathematics, University of Belgrade, Serbia, e-mail: ivona.brajevic@googlemail.com

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

bees that are currently exploiting a food source are known as employed and for every food source, there is only one employed bee. The employed bees exploit the food source and they carry the information about food source back to the hive and share this information with onlooker bees. Onlooker bees are waiting in the hive for the information to be shared by the employed bees about their discovered food sources and scouts bees will always be searching for new food sources near the hive. Employed bees share the information about food sources by dancing in the dance area inside the hive. The dance is dependent on the nectar content of food source just exploited by the dancing bee. Onlooker bees watch the dance and choose a food source according to the probability proportional to the quality of that food source. Therefore, good food sources attract more onlooker bees compared to bad ones. The employed bee whose food source has been abandoned by the bees becomes a scout. Scout bees can be visualized as performing the job of exploration, whereas employed and onlooker bees can be visualized as performing the job of exploitation.

Each food source is a possible solution for the problem and the nectar amount of a food source represents the quality of the solution represented by the fitness value. At the first step, the ABC generates a randomly distributed initial population of *SN* solutions, where *SN* denotes the number of food source positions which is equal to number of employed bees. Each solution X_i *(i = 1, 2, ..., SN)* is a *D*-dimensional vector and *D* is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles of the search processes of the employed bees, the onlooker bees and the scout bees. Maximum cycle number *MCN* is one of the four control parameters in the ABC algorithm.

In each iteration, every employed bee determines a food source in the neighborhood of its current food source and evaluates its nectar amount (fitness). The *i-th* food source position is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$. $F(X_i)$ refers to the nectar amount of the food source located at X_i. After watching the dancing of employed bees, an onlooker bee goes to the region of food source at X_i with the probability:

$$p_i = \frac{F(X_i)}{\sum_{k=1}^{SN} F(X_k)} \tag{1}$$

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression:

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

where $k \in \{1,2,...,SN\}$ and $j \in \{1,2,...,D\}$ are randomly chosen indexes where *k* has to be different from *i* and φ_{ij} is a random number between [-1, 1].

After each candidate source position v_{ij} is produced and then evaluated by the artificial bee, its performance is compared with that of its old one and a greedy selection mechanism is employed as the selection operation between the old and the new candidate. Otherwise, if the new food source has an equal or better nectar than the old source, it is replaced with the old one in the memory.

In ABC algorithm, providing that a position cannot be improved further through a predetermined number of cycles, the related food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called "limit for abandonment". Assuming that the abandoned source is $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, the scout discovers a new food source to replace X_i. This operation can be defined as:

$$x_{ij} = l_i + \delta(u_i - l_i) \tag{3}$$

where $j \in \{1,2,...,D\}$, l_i and u_i are the lower and upper bound of the parameter x_{ij} and δ is a random number in the range [0, 1). It can be concluded from the above explanation that there are four control parameters used in the ABC: the number of food sources which is equal to the number of employed or onlooker bees (*SN*), the value of limit, the maximum cycle number (*MCN*).

The pseudo code of the ABC algorithm is:

1. Initialize the population solutions x_{ij}, i=1,2,…,SN, j=1,2,…,D by Eq. (3)
2. Evaluate fitness value of the population
3. cycle = 1
4. **repeat**
5. Produce new solutions v_{ij} for the employed bees by using Eq. (2) and evaluate them
6. Apply the greedy selection process
7. Calculate the probability values P_{ij} for the solutions x_{ij} by Eq. (1)
8. Produce the new solutions v_{ij} for the onlookers from the solutions x_{ij} selected depending on P_{ij} and evaluate their fitness value
9. Apply the greedy selection process
10. Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_{ij} by Eq. (3)
11. Memorize the best solution achieved so far
12. cycle = cycle + 1
13. **until** cycle = MCN

B. The ABC Algorithm for Constrained Optimization Problems

General constrained optimization (CO) problem is to find *x* so as to

$$\text{minimize } f(x), \quad x = (x_1,...,x_n) \in R^n$$

where $x \in F \subseteq S$. The objective function f is defined on the search space $S \subseteq R^n$ and the set $F \subseteq S$ defines the feasible region. The search space S is defined as an *n*-dimensional rectangle in R^n. The variable domains are limited by their lower and upper bounds:

$$l_i \le x_i \le u_i \ , \ 1 \le i \le n$$

whereas the feasible region $F \subseteq S$ is defined by a set of *m* additional constraints ($m \ge 0$):

$$g_k(x) \le 0, \text{ for } k = 1,...,q$$

$$h_j(x) \le 0, \text{ for } j = q+1,..m$$

At any point $x \in F$, the constraints g_k that satisfy $g_k(x)=0$ are called the active constraints at *x*. By extension, equality constraints h_i are also called active at all points of S [16]. Two methods are often used to handle constraints. One is a variable restriction method which restricts the solution space to the solutions which conform to the constraints. The second one is a penalty function method which allows solutions which violate the constraints at the expense of a suitably defined penalty function. However, determining appropriate penalty coefficients is not an easy task, it must be estimated based on the relative scaling of the distance metrics of multiple constraints, the difficulty of satisfying a constraint, and the seriousness of a constraint violation, or be determined experimentally. Although the adaptive penalty strategies proved to be effective in some cases, they are still quite problem-dependent. It was also noticed that if the values of the objective function were very large, then small differences between objective values were not easily identified, and this was undesirable, a leading cause of premature convergence in evolutionary computation [17].

In order to handle the constraints of this problem, the ABC algorithm employs Deb's rules [18], which are used instead of the greedy selection employed between v_i and x_i in the version of the ABC proposed for unconstrained optimization problems. The method uses a tournament selection operator, where two solutions are compared at a time by applying the following criteria:

- Any feasible solution satisfying all constraints is preferred to any infeasible solution violating any of the constraints
- Among two feasible solutions, the one having better fitness value is preferred
- Among two infeasible solutions, the one having the smaller constraint violation is preferred

Because initialization with feasible solutions is very time consuming process and in some cases it is impossible to produce a feasible solution randomly, the ABC algorithm does not consider the initial population to be feasible. Structure of the algorithm already directs the solutions to feasible region in running process due to the Deb's rules employed instead of greedy selection. Scout phase of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population. Beside of Deb's rules, the second change in ABC for CO problems is in order to produce a candidate food position from the old one in memory. The adapted ABC algorithm uses the following expression:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}), & if \quad R_j \le MR \\ x_{ij} , & otherwise \end{cases} \tag{4}$$

where $k \in \{1,2,...,SN\}$ and $j \in \{1,2,...,D\}$ are randomly chosen indexes where *k* has to be different from *i* and φ_{ij} is a random number between [-1, 1]. *Rj*, $j \in \{1,2,...,D\}$, is a randomly chosen real number in the range [0,1]. *MR*, the modification rate, is a control parameter that controls whether the parameter x_{ij} will be modified or not. In adapted ABC algorithm, artificial scouts are produced at a predetermined period of cycles for discovering new food sources randomly. This period is another control parameter called scout production period (*SPP*) of the algorithm. At every *SPP* cycle, it is checked if there is an abandoned food source or not. If there is, a scout production process is carried out.

III. Proposed Algorithm: SC-ABC

In our proposed approach (called Simple Constrained Artificial Bee Colony, or SC-ABC) as in the ABC for constrained problems, algorithm uses Deb's rules instead of the greedy selection in order to decide what solution will be kept. The expressions for evaluating probability Eq. (1), for producing a candidate food position Eq. (2) and for initialization new food sources Eq. (3) stayed the same as in the version of the ABC proposed for unconstrained optimization problems.

SC-ABC algorithm has changed the initialization phase and the scout phase compared to the ABC. In the initialization phase only the first initialization of food sources is completely random. In other initialization phases the first new food source is the food source from the previous run of the algorithm which has the best fitness value. In other words, the runs of the SC-ABC algorithm are not completely independent. Therefore, exploitation of the good sources was increased. In order to increase the exploration the scout bee's phase was changed. In the scout phase the algorithm checks every possible solution. If the solution is not feasible, that food source is replaced with a new randomly produced solution.

The pseudo code of the SC-ABC algorithm is:

1. Initialize the population solutions x_{ij}, $i = 1,2,...,SN$, $j = 1,2,...,D$ by Eq. (3) for the first run. For every other run, if exists, x_{1j}, $j = 1,2,...,D$ is the best solution from previous run and x_{1j}, $i = 2,...,SN$, $j = 1,...,D$ are randomly

produced solutions by Eq. (3)
2. Evaluate fitness value of the population
3. cycle = 1
4. **repeat**
5. Produce new solutions v_{ij} for the employed bees by using Eq. (2) and evaluate them
6. Apply selection process based on Deb's method
7. Calculate the probability values P_{ij} for the solutions x_{ij} by Eq. (1)
8. Produce the new solutions v_{ij} for the onlookers from the solutions x_{ij} selected depending on P_{ij} and evaluate their fitness value
9. Apply selection process based on Deb's method
10. Determine the abandoned feasible solution for the scout, if exists, and replace it with a new randomly produced solution x_{ij} by Eq. (3)
11. Every infeasible solution replace with randomly produced solution x_{ij} by Eq. (3)
12. Memorize the best solution achieved so far
13. cycle = cycle + 1
14. **until** cycle = MCN

The original ABC can be applied only to the continuous problems. However, the method can also be expanded to the discrete problems using discrete numbers. The state variables were treated in the SC-ABC as follows: for continuous variables, initial values were generated randomly between upper and lower bounds of the specification values. The value was also modified in the employed and onlooker bee's phases between the bounds. For discrete variables, they could be handled in Equations (2) and (3) with a small modification, i.e., as though they were continuous with nearest available discrete values then being chosen. In that way, both continuous and discrete numbers can be handled by the algorithm with no inconsistency.

IV. Benchmark Problems

Proposed approach to Artificial Bee Colony Algorithm for constrained optimization problems (SC-ABC) was applied to four numerical examples: welded beam design optimization problem, pressure vessel design optimization problem, tension/compression spring design optimization problem and speed reducer design optimization problem [1]. These non-linear engineering design problems have discrete and continuous variables. Discrete variables are used in many ways such as the representation of the set of standard sized components, the decision on the number of identical parts or the choice between different design options. For example, the number of teeth on pinion of a speed reducer are integer variables, the spherical head thickness pitch and the shell thickness of a pressure vessel are discrete variables. The benchmark problems represent optimization situations involving discrete and continuous variables that are similar to those encountered in everyday mechanical engineering design tasks.

A. Welded beam design optimization problem

The problem consists in dimensioning a welded steel beam and the welding length so as to minimize its cost subject to constraints on shear stress, τ, bending stress in the beam, σ, buckling load on the bar, Pc, end deflection of the beam, δ, and side constraints [1], [19]. The beam has a length of 14 in. and 6000 lb. force is applied at the end of the beam. There are four continuous variables: x_1, x_2, x_3, x_4, which in structural engineering are commonly symbolized by the letters shown in Fig. 1 (h,l,t,b). The design variables are thickness of the weld h, length of the weld l, width of the beam t, and thickness of the beam b.

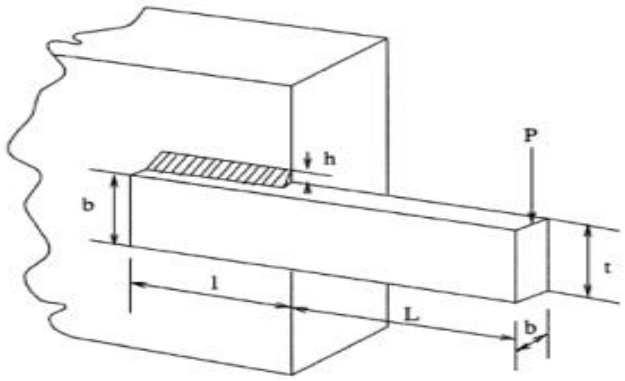

Fig.1: The Welded Beam design structure

The mathematical model of the problem is:

Minimize

$$f(X) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \quad (5)$$

subject to:

$$c_1(x) = t(x) - t_{\max} \le 0$$

$$c_2(x) = \sigma(x) - \sigma_{\max} \le 0$$

$$c_3(x) = x_1 - x_4 \le 0$$

$$c_4(x) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0$$

$$c_5(x) = 0.125 - x_1 \le 0$$

$$c_6(x) = \delta(x) - \delta_{\max} \le 0$$

$$c_7(x) = P - Pc(x) \le 0$$

The important stress conditions which were used in mathematical model is Weld Stress - $t(x)$. The weld stress has two components. They are t_1 - primary stress and t_2 - secondary stress. M is the moment which is created by Force (F). J is polar inertia moment. This parameters are defined as follows:

$$t(x) = \sqrt{{t_1}^2 + \frac{2t_1t_2x_2}{2R} + {t_2}^2}$$

$$t_1 = \frac{P}{\sqrt{2}x_1x_2}$$

$$t_2 = \frac{MR}{J}$$

$$M = P(L + \frac{x_2}{2})$$

$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{{x_2}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

Bar bending stress - $\sigma(x)$ is given as follows:

$$\sigma(x) = \frac{6PL}{x_4{x_3}^2}$$

Bar deflection - $\delta(x)$ is given as follows:

$$\delta(x) = \frac{4PL^3}{Ex_4{x_3}^3}$$

Bar buckling load is given as follows:

$$Pc(x) = \frac{4.013\sqrt{EG{x_3}^2{x_4}^6/36}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$$

The material properties and constraint values like shearing modulus, young's modulus, etc. used above are given as follow: P=6000, L=14, $\delta_{\max} = 0.25$, $E = 30 \cdot 10^6$, $G = 12 \cdot 10^6$, t_{max}=13600, $\delta_{\max} = 30000$.

The bounds are: $0.1 \le x_1 \le 2$, $0.1 \le x_2 \le 10$, $0.1 \le x3 \le 10$ and $0.1 \le x_1 \le 2$.

Best solution:
$x^* = (0.205733.470489 9.036624 0.20573)$
where $f(x^*) = 1.724852$.

B. Pressure Vessel design optimization problem

This example is to design a compressed air storage tank with a working pressure of 3000 psi and a minimum volume of 750 ft^3 [1], [17]. The schematic of a pressure vessel is shown in Fig.2. A cylindrical vessel is capped at both ends by hemispherical heads. Using rolled steel plate, the shell is made in two halves that are joined by two longitudinal welds to form a cylinder. Each head is forged and then welded to the shell. Let the design variables be denoted by the vector $X=[x_1,x_2,x_3,x_4]^T$, where x_1 is the spherical head thickness, x_2 is the shell thickness, x_3 and x_4 are the radius and length of the shell, respectively. The objective is to minimize the manufacturing cost of the pressure vessel. The manufacturing cost of pressure vessel is a combination of material cost, welding cost and forming cost. The design variables x_1 and x_2 have to be integer multiples of 0.0625 inch which are the available thickness of rolled steel plates. The radius x_3 and the length x_4 are continuous variables.

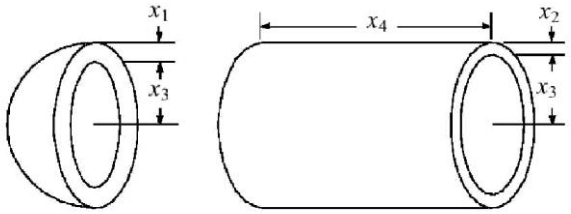

Fig.2: Pressure Vessel design

The mathematical model of the problem is:

Minimize

$$f(X) = 0.6224x_1x_3x_4 + 19.84{x_1}^2x_3 + 1.7781x_2{x_3}^3 + 3.1661{x_1}^2x_4 \quad (6)$$

subject to:

$$c_1(x) = -x_1 + 0.0193x_3 \le 0$$

$$c_2(x) = -x_2 + 0.0954x_3 \le 0$$

$$c_3(x) = -\pi{x_3}^2x_4 - \frac{4}{3}\pi{x_3}^3 + 1296000 \le 0$$

$$c_4(x) = x_4 - 240 \le 0$$

where the bounds are: $1 \times 0.0625 \le x_1 \le 99 \times 0.0625$, $1 \times 0.0625 \le x_2 \le 99 \times 0.0625$ and $10 \le x_3, x_4 \le 200$

Best solution:
$x^* = (0.8125 0.4375 42.098446 176.636596)$
where $f(x^*) = 6059.714335$.

C. Tension/compression spring design optimization problem

This problem minimizes the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables [1], [10], [20]. There are three continuous variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. The schematic

of a tension/compression spring is shown in Fig.3.

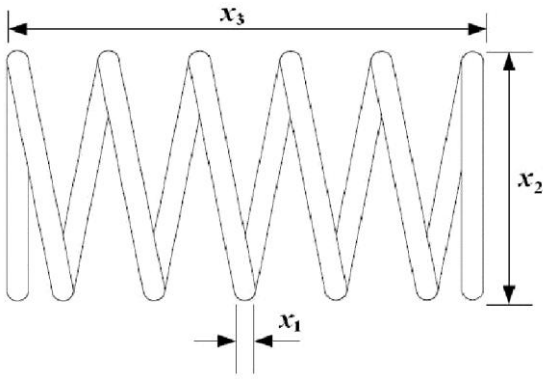

Fig.3: Tension/Compression Spring

The mathematical model of the problem is:

Minimize

$$f(X) = (x_3 + 2)x_1^2 x_2 \tag{7}$$

subject to:

$$c_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0$$

$$c_2(x) = \frac{4x_2^2 - x_1 x_2}{12566 x_1^3 x_2 - x_1^4} + \frac{1}{5108 x_1} - 1 \le 0$$

$$c_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$c_4(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0$$

where the bounds are: $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$ and $2.0 \le x_3 \le 15.0$

Best solution:

$x^* = (0.051690, 0.356750, 11.287126)$

where $f(x^*) = 0.012665$

D. Speed Reducer design optimization problem

This problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed. The design of the speed reducer shown in Fig.4, is considered with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7.

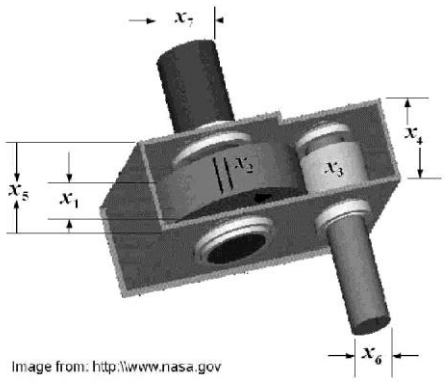

Fig.4: Speed Reducer

This is an example of a mixed integer programming problem. All variables are continuous except x_3 that is integer. The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft.

The mathematical model of the problem is:

Minimize

$$\begin{aligned} f(X) &= 0.7854 x_1 x_2^2 (3.3333 x_3^2 \\ &+ 14.9334 x_3 - 43.0934) \\ &- 1.508 x_1 (x_6^2 + x_7^2) + 7.4777 (x_6^3 + x_7^3) \\ &+ 0.78054 (x_4 x_6^2 + x_5 x_7^2) \end{aligned} \tag{8}$$

subject to:

$$c_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0$$

$$c_2(x) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0$$

$$c_3(x) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0$$

$$c_4(x) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0$$

$$c_5(x) = \frac{1.0}{110 x_6^3} \sqrt{\left(\frac{750.0 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0$$

$$c_6(x) = \frac{1.0}{85 x_7^3} \sqrt{\left(\frac{750.0 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0$$

$$c_7(x) = \frac{x_2 x_3}{40} - 1 \le 0$$

$$c_8(x) = \frac{5x_2}{x_1} - 1 \le 0$$

$$c_9(x) = \frac{x_1}{12x_2} - 1 \le 0$$

$$c_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$c_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

where the bounds are: $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.8 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$, and $5.0 \le x_7 \le 5.5$

Best solution:

$x^* = (3.5, 0.7, 17, 7.3, 7.8, 3.350214, 5.286683)$
where $f(x^*) = 2996.348165$.

V. Parameter Settings, Results and Discussion

A. Figures and Tables

Control parameters of the ABC algorithm are: swarm size, limit, number of employed bees, number of onlookers, number of scouts and maximum number of cycles [11]. In these experiments, the colony size was taken 40 and the maximum number of cycles was taken 4000. So, the total objective function evaluation number is 240000. Each experiment was repeated 60 runs. The percentages of onlooker bees and employed bees were 50% of the colony and the number of scout bees was changeable, as it was described in previous section. The value of "limit" is equal SN(2D+1) where *SN* is the number of possible solutions and *D* is the dimension of the problem. The performance of the algorithm was considered in terms of the best and average optimum values, and the best solutions were recorded. Our approach to ABC algorithm has been implemented in Java programming language and run on a Pentium Core2Duo, 1.40-GHz personal computer with 2 GB RAM memory.

Parameters adopted for SC-ABC algorithm are given in Table 1.

TABLE I
CONTROL PARAMETERS ADOPTED FOR SC-ABC ALGORITHM

Control parameters for SC-ABC algorithm	
swarm size	40
limit	*SN*(2D+1)*
number of onlookers	50% of the swarm
number of onlookers	50% of the swarm
number of scouts	changeable

Tables 2, 3, 4 and 5 show the solution vectors of the best solution reached by our approach to ABC algorithm and the values of the constrains for each of the problems tested. From these tables can be concluded that the SC-ABC reached for the first three tested problems almost the best known values, and for the fourth tested problem the best known value. It is important to mention that for the first two tested problems the program in the most of executions found solution at value between 1.73 and 1.74 and between 6060 and 6061, respectively. For the problem Tension/compression spring the program in the most of executions find the solution at value 0.01269 and for the problem Speed reducer, SC-ABC reached the best known value in every run of the program execution.

TABLE II
ABC SOLUTION VECTOR FOR WELDED BEAM DESIGN OPTIMIZATION PROBLEM

	Best solution
x_1	0.205563
x_2	3.471719
x_3	9.042758
x_4	0.205836
$c_1(x)$	-0.042486
$c_2(x)$	-56.120983
$c_3(x)$	-2.721E-4
$c_4(x)$	-3.431014
$c_5(x)$	-0.080563
$c_6(x)$	-0.235577
$c_7(x)$	-11.964330
f(x)	1.726625

TABLE III
ABC SOLUTION VECTOR FOR PRESSURE VESSEL DESIGN OPTIMIZATION PROBLEM

	Best solution
x_1	0.812500
x_2	0.437500
x_3	42.098187
x_4	176.640750
$c_1(x)$	-4.988451
$c_2(x)$	-0.035883
$c_3(x)$	-5.297613
$c_4(x)$	-63.359250
f(x)	6059.768058

TABLE IV
ABC SOLUTION VECTOR FOR TENSION / COMPRESSION SPRING DESIGN OPTIMIZATION PROBLEM

	Best solution
x_1	0.051871
x_2	0.361108
x_3	11.036860
$c_1(x)$	-1.634E-7
$c_2(x)$	-4.383E-5

$c_3(x)$	-4.062131
$c_4(x)$	-0.724680
f(x)	0.012667

TABLE V
ABC SOLUTION VECTOR FOR SPEED REDUCER DESIGN OPTIMIZATION PROBLEM

	Best solution
x_1	3.500000
x_2	0.700000
x_3	17
x_4	7.300000
x_5	7.800000
x_6	3.350215
x_7	5.286683
$c_1(x)$	-0.073915
$c_2(x)$	-0.197996
$c_3(x)$	-0.499172
$c_4(x)$	-0.90147
$c_5(x)$	-2.220E-16
$c_6(x)$	-3.331E-16
$c_7(x)$	-0.702500
$c_8(x)$	0.000000
$c_9(x)$	-0.583333
$c_{10}(x)$	-0.051326
$c_{11}(x)$	-0.010852
f(x)	2996.348165

Our results were compared to the results reached by Simple Constrained Particle Swarm optimization algorithm (SiC-PSO). Tables 6 and 7 show best, average fitness values and standard deviation for each of the problems tested.

TABLE VI
BEST RESULTS OBTAINED BY SC-ABC AND SIC-PSO

Prob.	**Optimal**	**SC-ABC**	**SiC-PSO**
Ex. 1	1.724852	1.726625	1.724852
Ex. 2	6059.714335	6059.768058	6059.714335
Ex. 3	0.012665	0.012667	0.012665
Ex. 4	NA	2996.348165	2996.348165

TABLE VII
AVERAGE AND STANDARD DEVIATIONS FOR THE RESULTS OBTAINED

	Average		St. Dev.	
Prob.	SC-ABC	SiC-PSO	SC-ABC	SiC-PSO
Ex. 1	1.7413	2.0574	2.29E-4	0.2154
Ex. 2	6060.2097	6092.0498	0.0069	12.1725
Ex. 3	0.0127	0.0131	2.4 E-07	4.1 E-04
Ex. 4	2996.3482	2996.3482	0.0000	0.0000

The results from Table 6 and Table 7 show that the average values reached by SC-ABC, for each problem tested, are better than the average values reached by SiC-PSO. But the SiC-PSO reached the best known values for each problem tested. It can be seen that the SC-ABC can converge very quickly towards the global optimum, except for the problem Welded beam. To have better results the SC-ABC algorithm needs to be modified in some way to avoid the algorithm to trap at some local attractors.

VI. CONCLUSION

In this paper, we present an improved ABC algorithm for constrained problems (SC-ABC). The SC-ABC was tested on three constrained optimization problems which contain discrete and continuous variables. The algorithm showed a good performance. We compared our results to the results reached by Simple Constrained Particle Swarm optimization algorithm (SiC-PSO) which showed a very good performance when it was applied to the same problems. Although our algorithm did not obtain the optimal values for each tested problem, the average values reached by SC-ABC are better. We can conclude that the SC-ABC can quickly search toward the global optimum and can be a promising alternative for solving this sort of problems due to its simplicity and reliability. As part of our future work, we are interested to perform a more detailed statistical analysis of the performance of our proposed approach and to improve the new algorithm's ability to escape the local attractors.

REFERENCES

[1] L. C. Cagnina, S. C. Esquive: Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer, Informatica, No.32, 2008, pp. 319-326.

[2] O. Yeniay: A comparative study on optimization methods for the constrained nonlinear programming problems, Mathematical Problems in Engineering Hindawi Publishing Corporation, 2005, pp. 165-173.

[3] M. Ettaouil, C. Loqman: Constraint satisfaction problems solved by semidefinite relaxations, WSEAS Transactions on Computers, Vol.7, 2008, pp. 951-961.

[4] T. Y. Chen,Y. L. Cheng: Global optimization using hybrid approach, WSEAS Transactions on Mathematics, Vol.7 ,2008 , pp. 254-262.

[5] A. Baykasoglu, L. Özbakır, P. Tapkan: Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem, Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria.

[6] J. A. Joines, C. R. Houck: On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with gas, In: Proc. IEEE Int. Conf. Evol. Comp., 1994, pp. 579-585.

[7] L. Ozdamar: A dual sequence simulated annealing algorithm for constrained optimization, Proceedings of the 10th WSEAS International Conference on applied mathematics, Dallas, Texas, 2006, pp. 557-564.

[8] 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.

[9] L. D. Li, J. Zhou, X. Yu, and X. Li: Constrained Power Plants Unit Loading Optimization using Particle Swarm Optimization Algorithm, WSEAS Transactions on Information Science and Applications, Vol.4, 2007, pp. 296-302.

[10] M. Mahdavi, M. Fesanghary, E. Damangir: An improved harmony search algorithm for solving optimization problems, Elsevier, 2006, pp. 1567-1579.

[11] D. Karaboga: An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[12] D. Karaboga, B. Basturk: Artificial Bee Colony Optimization (ABC) Algorithm for Solving Constrained Optimization Problems, IFSA 2007, LNAI 4529, Springer-Verlag, Berlin, Heidelberg, pp. 789-798
[13] D. Karaboga, B. Basturk: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization, Vol. 39, 2007, pp. 459-471.
[14] 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, Prague, Czech Republic, 2009, pp. 84-89.
[15] 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 table of contents, Moscow, Russia, 2009, pp. 87-92.
[16] Z. Michalewicz, M. Schoenauer: Evolutionary Algorithms for Constrained Parameter Optimization Problems, Evolutionary Computation, 1995, pp. 1–32.
[17] C. X. Guo, J. S. Hu, B. Ye, Y. J. Cao: Swarm intelligence for mixed-variable design optimization, Journal of Zhejiang University Science, 2004, pp. 249-260.
[18] D.E. Goldberg, K. Deb: A comparison of selection schemes used in genetic algorithms Foundations of Genetic Algorithms, edited by G. J. E. Rawlins, 1991, pp. 69-93.
[19] A. R. Yildiz, Hybrid Taguchi: Harmony Search Algorithm for Solving Engineering Optimization Problems, International Journal of Industrial Engineering, 2008, pp. 286-293.
[20] J. Arora: Introduction to optimum design, McGraw-Hill, 1989.

Milan Tuba received B.S. in mathematics, M.S. in mathematics, M.S. in computer Science, M.Ph. in computer science, Ph.D. in computer science from University of Belgrade and New York University.

From 1983 to 1994 he was in the U.S.A. first as a graduate student and teaching and research assistant at Vanderbilt University in Nashville and Courant Institute of Mathematical Sciences, New York University and later as an assistant professor of electrical engineering at Cooper Union Graduate School of Engineering, New York. During that time he was the founder and director of Microprocessor Lab and VLSI Lab, leader of scientific projects and supervisor of many theses. From 1994 he was associate professor of computer science and Director of Computer Center at University of Belgrade, Faculty of Mathematics, and from 2004 also a Professor of Computer Science and Dean of the College of Computer Science, Megatrend University Belgrade. He was teaching more than 20 graduate and undergraduate courses, from VLSI design and Computer architecture to Computer networks, Operating systems, Image processing, Calculus and Queuing theory. His research interest includes mathematical, queuing theory and heuristic optimizations applied to computer networks, image processing and combinatorial problems. He is the author of more than 100 scientific papers and a monograph. He is coeditor or member of the editorial board or scientific committee of number of scientific journals and conferences.

Prof. Tuba is member of the ACM since 1983, IEEE 1984, New York Academy of Sciences 1987, AMS 1995, SIAM 2009. He participated in many WSEAS Conferences with plenary lectures and articles in Proceedings and Transactions.

Ivona Brajevic received B.S. in mathematics in 2006 and M.S. in mathematics in 2008 from University of Belgrade, Faculty of Mathematics.

She is currently Ph.D. student at Faculty of Mathematics, Computer science department, University of Belgrade and works as teaching assistant at College of Business, Economy and Entrepreneurship in Belgrade. She is the coauthor of two papers. Her current research interest includes nature inspired metaheuristics.

Ms. Brajevic participated in WSEAS conferences.

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 coauthor of two papers. His current research interest includes nature inspired metaheuristics.

Mr. Subotic participated in WSEAS conferences.