Modified artificial bee colony algorithm for constrained problems optimization

Nadezda Stanarevic, Milan Tuba, and Nebojsa Bacanin

***Abstract*—**Original Karaboga's artificial bee colony (ABC) algorithm was applicable to unconstrained problems only and modifications for constrained problems were introduced later. In this article we propose an improved artificial bee colony algorithm for constrained problems. Since the ABC algorithm for constrained problems does not consider the initial population to be feasible we introduced a modification, besides penalty function and Deb's rule, in a form of "smart bee" (SB) which uses its historical memories for the location and quality of food sources. This modified SB-ABC algorithm was tested on standard benchmark functions for constrained optimization problems and proved to be better.

***Keywords*—**Artificial bee colony, Constrained optimization, Nature inspired metaheuristic algorithms, Swarm intelligence.

I. Introduction

ALGORITHM that is working with a set of solutions and trying to improve them is called population based. Population based algorithms can be classified by the nature of phenomenon simulated by the algorithm into two groups: evolutionary algorithms (EA) and swarm intelligence based algorithms [1], [2]. The most popular among EA is genetic algorithm (GA). GA attempts to simulate the phenomenon of natural evolution. A branch of nature inspired algorithms which are called swarm intelligence is focused on collective behavior of some self-organized systems in order to develop some metaheuristics which can mimic such system's problem solution abilities [3], [4]. Interaction between individuals locally with one another and with their environment contributes to the collective intelligence of the social colonies [5]. Even though there is no centralized component that controls the behavior of individuals, local interactions between all individuals often lead to the emergence of global behavior. These characteristics of swarms inspired huge number of researchers to implement such behavior in computer software for optimization problems [6]. Flocking of birds and schooling of fish are examples of swarm systems. The classical example of a swarm is bees swarming around their hive but the metaphor can easily be extended to other systems with a similar architecture such as ants [7].

Nature inspired algorithms based on the social behavior of certain animals and insects can solve many complex problems such as the traveling salesman problem (TSP), vehicle routing, scheduling, networks design and many more [2]. Generally, such algorithms are applied to problems classified as NP-hard or NP-complete. Many practical problems in industry and business are in the class of intractable combinatorial (discrete) or numerical (continuous or mixed) optimization problems. Many traditional methods were developed for solving continuous optimization problems, while on the other hand, discrete problems are being solved using heuristics [8]. Complete search algorithms search all possible assignments of values to variables in the search space, but they have the disadvantage of being time consuming. In the past few years, the usage of metaheuristic algorithms has increased in popularity. A meta-heuristic is a general algorithmic framework which can be used in different optimization problems with few modifications to adapt it to a specific problem. Several modern metaheuristic algorithms (typically high-level strategies which guide an underlying subordinate heuristic to efficiently produce high quality solutions and increase their performance) that apply to both domains have been developed for solving such problems [1]. They include population based, iterative based, stochastic, deterministic and other approaches.

For example, ant colony optimization (ACO) is a quite successful technique proposed by Dorigo in 1992 for solving hard combinatorial optimization problems. The inspiring source of ACO was the foraging behavior of real ants which enables them to find shortest paths between food sources and their nests. In nature, researchers have found that during the foraging behavior of ants, they have no direct communication between each other, but they use an indirect method of communication through the segregation of a chemical called pheromone. Paths that contain more pheromone concentrations are chosen with higher probability by ants than those that contain lower pheromone concentrations. This method of communication has been defined as stigmergy. Stigmergy means any form of indirect communication among a set of possibly concurrent and distributed agents which happens through acts of local modification of the environment and local sensing of the outcomes of these modifications.

Manuscript received February 10, 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

N. Stanarevic is with the Faculty of Faculty of Computer Science, Megatrend University, Belgrade, Serbia, e-mail: srna@stanarevic.com

N. Bacanin is with the Faculty of Computer Science, Megatrend University, Belgrade, Serbia, e-mail: nbacanin@megatrend.edu.rs

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart in 1995 to emulate the behavior of a flock of birds or a school of fish. It is considered a swarm intelligence algorithm because it tries to emulate the behavior of social organisms. In the case of birds, they choose a leader to guide the search for food so that other individuals will follow the leader during this search, however, each individual will maintain a personal search of food and try to inform the group if they find it. In nature, when there is an individual with better attributes than the current leader, the leader is replaced. Each individual must be able to get a general idea of their surroundings to avoid collisions with other individuals and to avoid losing sight of the leader's direction. PSO is stochastic optimization technique which is well adapted to the optimization of nonlinear functions in multidimensional space and it has been applied to several real-world problems [9].

One of the recently used examples of interactive behavior and swarm intelligence is the waggle dance of bees during the food procuring. By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates. So this is a successful mechanism by which foragers can recruit other bees in their colony to productive locations to collect various resources. Bee colony can quickly and precisely adjust its searching pattern in time and space according to changing nectar sources. The information exchange among individual insects is the most important part of the collective knowledge. Communication among bees about the quality of food sources is being achieved in the dancing area by performing waggle dance.

Several metaheuristics have been proposed to model the specific intelligent behavior of honey bee swarms [10], [11], [12], [13]. The bee swarm intelligence was used in the development of artificial systems aimed at solving complex problems [14], [15]. An algorithm called bee colony optimization metaheuristic (BCO) is used for problems in traffic and transportation [16] and for solving deterministic combinatorial problems, as well as combinatorial problems characterized by uncertainty.

Drias introduced a novel intelligent approach called bees swarm optimization (BSO), which is also inspired by the behavior of real bees. BSO is adapted for solving maximum weighted satisfiability (max-sat) problem.

Karaboga introduced the artificial bee colony (ABC) algorithm. In the ABC algorithm the first half of the colony consists of the employed bees and the second half includes the onlookers. For every food source, there is only one employed bee. Another issue that is considered in the algorithm is that the employed bee whose food source has been exhausted by the bees becomes a scout. In other words, if a solution representing a food source is not improved by a predetermined number of trials, then the food source is abandoned by its employed bee and the employed bee is converted to a scout.

In this paper, we present enhancements of the artificial bee colony algorithm for constrained problems proposed by Karaboga and Bastuk [11]. We also measure performance of this enhanced algorithm against Karaboga`s original work.

II. ABC Algorithm

Several approaches have been proposed to model the specific intelligent behaviors of honey bee swarms. Real-world problems usually have many design parameters that should be considered in the design process. Algorithms that are not robust to large-scale problems cannot preserve their effectiveness against high dimensionality. Artificial bee colony is a relatively new member of swarm intelligence algorithms.

In the ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. Short pseudo-code of the ABC algorithm is given below [10]:

- 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
- Determine the abandoned solution for the scout, and replace it with a new randomly produced solution
- Memorize the best solution achieved so far

For every food source, there is only one employed bee. Every bee colony has scouts that are the colony's explorers. The scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. Each solution x_i $(i = 1, 2, ..., SN)$ is a D-dimensional vector, where SN denotes the size of population.

An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution). Provided that the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one [11]. Otherwise she keeps the position of the previous one in her memory. The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. The employed bee of an abandoned food source becomes a scout.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, p_i, calculated by the following expression

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

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.

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

$$\upsilon_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) \tag{2}$$

where $k \in \{1, 2,.., SN\}$ and $j \in \{1, 2,...,D\}$ are randomly chosen indexes. A greedy selection mechanism is employed as the selection operation between the old and the candidate one. Providing that a position cannot be improved further through a predetermined number of cycles, the 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"* [11]. In the ABC, the parameter *limit* is calculated using the formula *SN*D*, where *SN* is the number of solutions and *D* is the number of variables of the problem.

There are three main control parameters that are 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. In the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. An important difference between ABC and other swarm intelligence algorithms is that in the ABC algorithm the possible solutions represent food sources (flowers), not individuals (honeybees) [12]. In other algorithms, like PSO, each possible solution represents an individual of the swarm. In the ABC algorithm the fitness of a food source is given by the value of the objective function of the problem.

III. Constrained optimization

Evolutionary algorithms have been widely used to solve several types of unconstrained optimization problems. Many real-world search and optimization problems involve inequality and equality constraints and are thus posed as constrained optimization problems. Real world problems often have unique characteristics that make them difficult (and sometimes impossible) to solve by applying traditional methods. Moreover, there are problems where these methods can be applied, however the results or the time required to obtain a solution are not what is expected by the problem solver. Michalewicz and Fogel [17] describe the following characteristics that make it difficult to solve an optimization problem in the real world:

1. The number of possible solutions (search space) is too large.
2. The problem is so complicated that, with the aim of obtaining a solution, simplified models of the same problem must be used. Thus, the solution is not useful.
3. The evaluation function that describes the quality of each solution in the search space varies over time or it has noise.
4. Possible solutions are highly restricted, making it difficult even generating at least one feasible solution (i.e., satisfy the constraints of the problem).

Constrained optimization (CO) problems are encountered in numerous applications such as: structural optimization, engineering design, VLSI design, economics and more. The considered problem is reformulated so as to take the form of optimizing two functions, the objective function and the constraint violation function. This has motivated the development of a considerable number of approaches to incorporate constraints into the fitness function of an evolutionary algorithm. The CO problem can be represented as the following nonlinear programming problem [18]:

$$\text{minimize } f(x),\ x=(x_1,\ ...,\ x_n) \in R^n \tag{3}$$

where $x \in F \in S$. The objective function f is defined on the search space $S \in R^n$ and the set $F \in S$ defines the feasible region. Usually, the search space S is defined as an n-dimensional rectangle in R^n (domains of variables defined by their lower and upper bounds):

$$lb_i \le x_i \le ub_i, \quad 1 \le i \le n \tag{4}$$

the feasible region $F \in S$ is defined by a set of m additional constraints:

$$g_j(x) \le 0, \text{ for } j = 1, \ldots, q$$
$$h_j(x) = 0, \text{ for } j = q + 1, \ldots, m. \tag{5}$$

At any point $x \in F$, the constraints g_k that satisfy $g_k(x) = 0$ are called the active constraints at x.

Constraint handling methods used in classical optimization algorithms can be classified into two groups: generic methods that do not exploit the mathematical structure (whether linear or nonlinear) of the constraint, and specific methods that are only applicable to a special type of constraints. The constrained optimization problems can be addressed using either deterministic or stochastic methods. Deterministic approaches such as feasible direction and generalized gradient descent make strong assumptions on the continuity and differentiability of the objective function [18]. On the other hand, stochastic optimization algorithms such as genetic algorithms, evolution strategies, volutionary programming and particle swarm optimization do not make such assumptions and they have been successfully applied for tackling constrained optimization problems during the past few years.

The goal of an optimization method is to assign values, within the allowed domain, to the variables, so the objective function is optimized and the restrictions are satisfied. Therefore, the optimization algorithm seeks a solution in the search space S of candidate solutions.

In case of problems with constraints, a desired solution must be located in the feasible space F, $F \in S$, where feasibility means that the solution satisfies all the constraints. Most of the methods to solve constrained problems start with solutions that are outside of the feasible area and it is expected that, after some computational time, these solutions reach the feasible area. A basic graphical example in an artificial two-dimensional problem can be seen in Fig. 1.

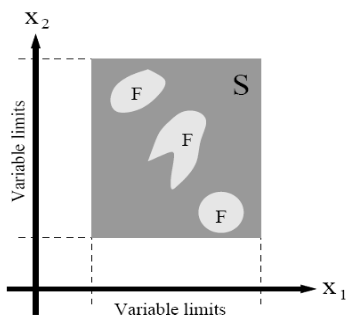


Fig. 1: Example of Search (S) and Feasible (F) areas in a two-dimensional problem

A. *Equality constraints*

Equality constraints are difficult to satisfy because they define a very small search space. Therefore it is common practice to rewrite the equality constraints as inequality constraints as seen in Equation (5):

$$|h(x)| - \varepsilon \leq 0 \tag{6}$$

where ε is the tolerance given to the equality constraint, slightly extending the search space of the problem. The disadvantage of this method is that the solutions found can be slightly infeasible due to this tolerance. All equality constraints are converted into inequality constraints ε=0.001. We can see a graphical representation of this method in Fig. 2.

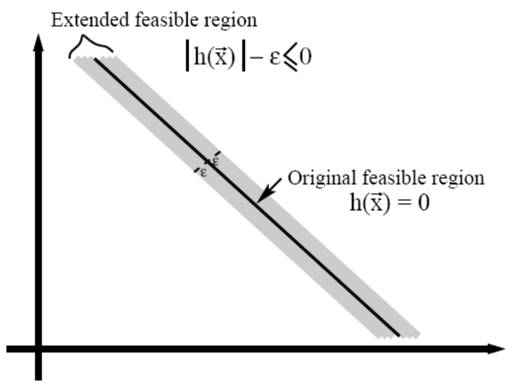


Fig. 2: Graphical example of an equality constraint converted to an inequality constraint

IV. ABC ALGORITHM MODIFICATIONS FOR CONSTRAINED OPTIMIZATION PROBLEMS

The ABC algorithm has been firstly proposed for unconstrained optimization problems and showed that it has superior performance on these kinds of problems [10]. The search space in constrained optimization problems consists of two kinds of points: feasible and unfeasible. Feasible points satisfy all the constraints, while unfeasible points violate at least one of them. For solving constrained optimization problems the ABC algorithm has been modified. The most common approach adopted to deal with constrained search spaces is the use of penalty functions. When using a penalty function, the amount of constraint violation is used to punish or "penalize" an infeasible solution so that feasible solutions are favored by the selection process. However, since the penalty function approach is generic and applicable to any type of constraint (linear or nonlinear), their performance is not always satisfactory. Thus, researchers have developed sophisticated penalty functions specific to the problem at hand and the search algorithm used for optimization. However, the most difficult aspect of the penalty function approach is to find appropriate penalty parameters needed to guide the search towards the constrained optimum. Penalty functions require a careful fine tuning of the penalty factors that accurately estimates the degree of penalization to be applied so that we can approach efficiently the feasible region.

The first proposal, to extend the ABC algorithm [19] to constrained spaces, used a constraint handling technique originally proposed for a genetic algorithm by Deb [20], [21]. Deb has developed a constraint handling method based on the penalty function approach which does not require any penalty parameter. In order to adapt the ABC algorithm Karaboga has adopted Deb's constrained handling method instead of the selection process (greedy selection) of the ABC algorithm.

Deb's method uses a tournament selection operator, where two solutions are compared at a time, and the following criteria are always enforced:

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.

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 [19]. 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 production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population.

In the ABC for constrained optimization, in order to produce a candidate food position (by an employed or an onlooker bee) the following is used:

$$\upsilon_{i,j} = \begin{cases} x_{i,j} + \phi * (x_{i,j} - x_{k,j}), & R_j < MR \\ x_{i,j} & otherwise \end{cases} \tag{7}$$

where $k \in \{1, 2, \ldots, SN\}$ is randomly chosen index., $x_{i,j}$ is the variable j of the current food source, x_k is a randomly selected solution (different from $x_{i,j}$), Rj is a randomly chosen real number in the range [0,1], $j \in \{1, 2, \ldots, D\}$, D is the number of variables of the problem. *MR*, modification rate, is a new parameter that Karaboga and Basturk added to the ABC algorithm. It is a control parameter that controls whether the parameter x_{ij} will be modified or not.

In the ABC algorithm, if a solution constructed by an employed bee or an onlooker bee exceeds the boundaries of the variable, the variable takes the value of the trespassed bound. In our algorithm we have used a different mechanism from original ABC algorithm, based on the Kukkonen and Lampinen work [22]:

$$\upsilon_{i,j} = \begin{cases} 2*lb_j - \upsilon_{ij}, & if \;\; \upsilon_{ij} < lb_j \\ 2*ub_j - \upsilon_{ij}, & if \;\; \upsilon_{ij} > ub_j \\ \upsilon_{i,j} & otherwise \end{cases} \tag{8}$$

where υ_{ij} is the variable j of the candidate solution i, lb_j is the lower bound of the variable j and ub_j is the upper bound of variable j.

Since the ABC algorithm does not consider the initial population to be feasible we have decided to add a smart bee. This type of bee uses its historical memories for the location and quality of food sources. Smart bee can memorize the position of the best food source and its quality which was found at previous times [23]. The position of the best food source replaces the position of the random new food source in two cases: if the new food source is unfeasible solution, or if the new food source is feasible solution but it doesn't have better fitness.

1. Initialize the population of solutions
2. Evaluate the population
3. cycle=1
4. repeat
5. Produce new solutions for the employed bees by using (7) and evaluate them
6. If cycle≠1 use smart bee
7. Apply selection process based on Deb's method
8. Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ using fitness of the solutions and the constraint violations (CV) by

$$pi = \begin{cases} 0.5 + \left(\dfrac{fitness_i}{\sum\limits_{i=1}^{SN} fitness_i} \right) * 0.5 \;\; if \;\; solution \;\; is \;\; feasible \\ \left(1 - \dfrac{CV}{\sum\limits_{i=1}^{SN} CV} \right) * 0.5 \; if \;\; solution \;\; is \;\; infeasible \end{cases}$$

where CV is defined by

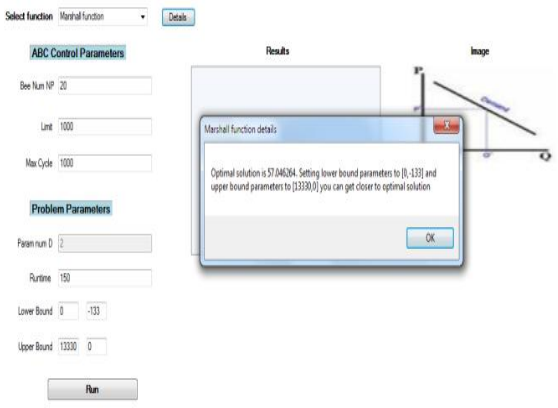
$$CV = \sum_{1}^{q} g_j(x) \; + \; \sum_{q+1}^{m} h_j(x)$$

9. For each onlooker bee, produce a new solution υ_i by Equation (7) in the neighborhood of the solution selected depending on p_i and evaluate it
10. Apply selection process between υ_i and x_i based on Deb's method
11. If Scout Production Period (SPP) is completed, determine the abandoned solutions by using "limit" parameter for the scout, if it exists, replace it with a new randomly produced solution by

$$x_i^j = x_{\min}^j + rand(0,1) * (x_{\max}^j - x_{\min}^j)$$

12. Memorize the best solution achieved so far
13. cycle = cycle+1
14. until cycle = MCN

As we can see from the Fig.3, user can adjust multiple parameters for ABC algorithm.


Select function: Marshall function
Details

ABC Control Parameters
Bee Num NP: 20
Limit: 1000
Max Cycle: 1000

Problem Parameters
Param num D: 2
Runtime: 150
Lower Bound: 0 -133
Upper Bound: 13330 0

Run

Marshall function details
Optimal solution is 57.046264. Setting lower bound parameters to [0,-133] and upper bound parameters to [13330,0] you can get closer to optimal solution
OK

Fig. 3: Additional information about selected function

The proposed SB-ABC algorithm is coded in C# and run on a Pentium Core2Duo, 3-GHz computer with 4 GB RAM memory.

Control parameters are:

1. Bee Num NP is number of bees in the colony (employed bees plus onlooker bees).
2. Limit controls the number of trials to improve certain food source. If a food source could not be improved within defined number of trial, it is abandoned by its employed bee.
3. Max Cycle defines the number of cycles for foraging. This is a stopping criterion

Problem specific parameters are:

1. Param Num D is the number of parameters of the problem to be optimized
2. Runtime defines the number of times to run the algorithm.
3. Lower bound is lower bound of problem parameters.
4. Upper bound is upper bound of problem parameters.
5. Constrained number is the number of constraints

V. Settings of the Algorithms

To evaluate the performance of the proposed Karaboga ABC algorithm and ABC algorithm with smart bee (*SB*) we used the set of benchmark constrained optimization functions proposed in [24], [25]. The performance of the SB-ABC algorithm is compared with ABC algorithm, particle swarm optimization (PSO) algorithms, self-adaptive penalty function genetic algorithm (SAPF-GA) proposed by Tessema and Yen, and hybrid constrained optimization evolutionary algorithm (HCOEA) proposed by Wang.

The test cases include objective functions of various types with different types of constraints. Basic function information are listed in Table 1. This set includes various forms of objective function such as linear, nonlinear cubic and quadratic.

TABLE I
Set of constrained optimization test function

Fun.	Dim.	Type	Optimal
G1	13	Quadratic	-15
G4	5	Quadratic	-30665.5386
G6	2	Cubic	-6961.814
G8	2	Nonlinear	-0.0958
G13	5	Nonlinear	0.0539

Benchmark constrained optimization functions are defined:

G1: Minimize:

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Subject to:

$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$
$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$
$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0$
$g_4(x) = -8x_1 + x_{10} \le 0$
$g_5(x) = -8x_2 + x_{11} \le 0$
$g_6(x) = -8x_3 + x_{12} \le 0$
$g_7(x) = -2x_4 - x_5 + x_{10} \le 0$
$g_8(x) = -2x_6 - x_7 + x_{11} \le 0$
$g_9(x) = -2x_8 - x_9 + x_{12} \le 0$

Where the bounds are $0 \le x_i \le 1$ $(i = 1, \dots, 9)$, $0 \le x_i \le 100$ $(i = 10, 11, 12)$ and $0 \le x_{13} \le 1$.
The global minimum is at
$x = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ and $f(x) = -15$.

G4: Minimize:

$5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$

Subject to:

$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$
$g_2(x) = 85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$
$g_4(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$
$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$
$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$

Where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, $27 \le x_i \le 45$ $(i = 3, 4, 5)$.
The optimum solution is $f(x) = -30665.538672$.

G6: Minimize:

$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to:

$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$
$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$

where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is located at $x = (14.09500000000000064, 0.8429607892154795668)$, $f(x) = -6961.813875580$.

G8: Minimize:

$$f(x) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to:

$g_1(x) = x_1^2 - x_2 + 1 \le 0$
$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$

Where $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$. The optimum solution is located at $x = (1.22797135260752599, 4.24537336612274885)$ where
$f(x) = -0.0958250414180359$.

G13: Minimize:

$$f(x) = e^{x_1x_2x_3x_4x_5}$$

Subject to:

$g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
$g_2(x) = x_2x_3 - 5x_4x_5 = 0$
$g_3(x) = x_1^3 + x_2^3 + 1 = 0$

Where $-2.3 \le xi \le 2.3$ $(i = 1, 2)$ and $-3.2 \le xi \le 3.2$ $(i = 3, 4, 5)$. The optimum solution is $x = (-1.717142240, 1.595721240494, 1.827250240, -0.76365988191, -0.7636598673)$ where $f(x) = 0.0539415140418$.

In ABC, the value of modification rate (*MR*) is 0.8, colony size (*2 *SN*) is 40 and the maximum cycle number (*MCN*) is 6000. So, the total objective function evaluation number is 240,000. The value of limit is equal to *SN x D* where *D* is the dimension of the problem and *SPP* is also *SNxD*. Experiments were repeated 30 times each starting from a random population with different seeds [19].

Control parameters of the ABC algorithm are: colony size, solution number, limit, maximum number of cycles and modification rate. Our algorithm was implemented using the parameters' values described in the Table 2.

TABLE II
PARAMETERS FOR THE MODIFIED ALGORITHM

Parameter	Symbol	Value
Colony size	NP	40
Solutions Number	SN	20
Maximum Cycle Number	maxCycle	6000
Limit	limit	SN*D
Modification Rate	MR	0.8

Each of the experiments was repeated 30 times with different random seeds and the best, worst and average function values were recorded. The results are in the Table 3.

TABLE III
STATISTICAL RESULTS OBTAINED BY SB-ABC, ABC, PSO, SAPF-GA AND HCOEA ALGORITHMS ON TEST FUNCTIONS

Function	Algor.	Worst	Best	Average
G1:	SB-ABC	-15.00000	-15.00000	-15.00000
-15.000	ABC	-15.00000	-15.00000	-15.00000
	PSO	-13.00000	-15.00000	-14.71000
	SAPF-GA	-13.097	-15.00000	-14.55200
	HCOEA	-14.999	-15.00000	-15.00000
G4:	SB-ABC	-30665.539	-30665.539	-30665.539
-30665.539	ABC	-30665.539	-30665.539	-30665.539
	PSO	-30665.539	-30665.539	-30665.539
	SAPF-GA	-30656.471	-30665.401	-30659.221
	HCOEA	-30665.539	-30665.539	-30665.539
G6:	SB-ABC	-6961.813	-6961.814	-6961.814
-6961.814	ABC	-6961.808	-6961.814	-6961,813
	PSO	-6961.814	-6961.814	-6961.814
	SAPF-GA	-6953.061	-6961.046	-6943.304
	HCOEA	-6961.813	-6961.813	-6961.813
G8:	SB-ABC	-0.095825	-0.095825	-0.095825
-0.095825	ABC	-0.095825	-0.095825	-0.095825
	PSO	-0.095825	-0.095825	-0.095825
	SAPF-GA	-0.092697	-0.095825	-0.095635
	HCOEA	-0.095825	-0.095825	-0.095825
G13:	SB-ABC	0.183	0.054	0.105
0.053950	ABC	1.000	0.760	0.968
	PSO	1.793	0.085	0.569
	SAPF-GA	0.885276	0.053941	0.28627
	HCOEA	0.0539499	0.0539498	0.0539498

An indirect comparison was performed between the published results of the ABC algorithm [19], PSO algorithm [25], SAPF-GA algorithm [26], HCOEA [27] and our proposed algorithm. In all test problems, the two variants of ABC algorithms exhibited similar results. Table 3, gives the summary of the comparative results of the best, mean and worst solutions of the investigated algorithms.

Our proposed algorithm had equal or better results on G1, G4, G6 and G8 than ABC, PSO, SAPF-GA and HCOEA algorithms. In comparison with SAPF-GA and HCOEA our algorithm had better or equal results in 5 of 6 problems while HCOEA presented a greater quality of the results than our proposed algorithm for G13 .

From the best, worst and mean results presented in Table 3, it can be concluded that the SB-ABC algorithm performs better than ABC and PSO algorithms. Our proposed algorithm, SB-ABC presented a greater quality of the results and more consistency in the results than the ABC algorithm. Based on the data we have acquired we can conclude that the better results have been made thanks to the implemented modifications, especially by introducing the smart bee.

VI. CONCLUSION

The capability of the ABC algorithm for constrained optimization problems was investigated through the performance of several experiments on well-known test problems. In this paper, we present an improved ABC algorithm for constrained problems. The SB-ABC was tested on five constrained optimization problems: quadratic, cubic, linear and nonlinear. The results obtained by the modified ABC algorithms for constrained optimization problems are quite satisfactory.

Future work will include investigation of the SB-ABC performance in other benchmark and real life problems. The main steps in further modifications of ABC algorithm for constrained problems are directed towards finding better feasible solutions that will guide the swarm towards the optimum solution. Also, the fine tuning of the parameters may result in better solutions. It has been concluded that the ABC algorithm can be efficiently used for solving constrained optimization problems. The performance of the SB-ABC algorithm can be also tested for real engineering problems existing in the literature and compared with other algorithms.

REFERENCES

[1] Xin-She Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008

[2] Johann Dréo, Patrick Siarry, Alain Pétrowski and Eric Taillard, Metaheuristics for Hard Optimization, Springer Berlin Heidelberg, pp. 1-19, 2006.

[3] Muddassar Farooq, Bee-Inspired Protocol Engineering: From Nature to Networks, Springer, 2008

[4] Xin-She Yang, Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Volume 3562/2005, No. 10.1007/b137296, 2005, pp. 317-323.

[5] Saif Mahmood Saab , Nidhal Kamel Taha El-Omari, Hussein H. Owaied, Developing optimization algorithm using artificial bee colony system, UbiCC Journal - Volume 4, No 5, 2009, pp. 391-396.

[6] Tricia Rambharose and Alexander Nikov, Computational intelligence-based personalization of interactive web systems, WSEAS Transactions on Information Science and Applications, Vol. 7, Issue 4, Apr 2010, pp. 484-497.

[7] N. Buniyamin, N. Sariff, W. A. J. Wan Ngah, Z. Mohamad, Robot Global Path Planning Overview and a Variation of Ant Colony System Algorithm, International Journal of Mathematics and Computers in Simulation, Vol. 5, Issue 1, 2011, pp. 9-16.

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

[9] Milan Rapaic, Zeljko Kanovic, Zoran Jelicic, A theoretical and empirical analysis of convergence related particle swarm optimization, WSEAS Transactions on Systems and Control, Vol. 4, Issue 11, Nov 2009, pp. 541-550.

[10] Karaboga D, Basturk B, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. of Global Optimization, Volume 39, No. 0925-5001, 2007, pp. 459-471.

[11] Karaboga D, Basturk B, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing, Volume 8, No. 1568-4946, 2007, pp. 687-697.

[12] Karaboga D, Akay B, Ozturk C, Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks, Modeling Decisions for Artificial Intelligence, Volume 4617/2007, No. 0302-9743, 2007, pp. 318-329.

[13] Baykasoglu A, Ozbakir L, Tapkan P. Artificial bee colony algorithm and its application to generalized assignment problem, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, No. 978-3-902613-09-7, 2007, pp. 113–144.

[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] 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.

[16] Teodorovic, D., Dell'Orco M., Bee colony optimization—a cooperative learning approach to complex transportation problems, Proceedings of the 10th EWGT Meeting, Poznan, 13-16, September 2005.

[17] Michalewicz Z, Fogel D B, How to Solve It: Modern Heuristics. Springer, 2004.

[18] Parsopoulos K, Vrahatis M, Particle Swarm Optimization Method for Constrained Optimization Problems, Intelligent technologies: theory and applications, Volume 76, No. 978-1-58603-256-2, 2002, pp. 214-220

[19] Karaboga D, Basturk B, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, Volume 4529/2007, No. 0302-9743, 2007, pp.789–798.

[20] Deb K, An Efficient Constraint-handling Method for Genetic Algorithms, Computer Methods in Applied Mechanics and Engineering, Volume 186, No. 0045-7825, 2000, pp. 311-338.

[21] Deb K. Optimization for Engineering Design, Algorithms and Examples 4th. Edition, chapter 1 Introduction, pages 1–30, Prentice-Hall India, 2000.

[22] Kukkonen S, Lampinen J, Constrained real-parameter optimization with generalized differential evolution, In Proceedings of IEEE Congress on Evolutionary Computation, No. 9723462, pp. 207-214, 2006

[23] Baykasoglu A, Ozbakır L, Tapkan P, 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, 2007

[24] Michalewicz Z, Schoenauer M, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation, 1996.

[25] Zavala A, Aguirre A, Diharce E, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO'05), No. 1-59593-010-8, 2005, pp. 209–216.

[26] Tessema B, Yen G. G. A self adaptive penalty function based algorithm for constrained optimization, In Proceedings of IEEE Congress on Evolutionary Computation, No. 10.1109/CEC.2006.1688315, 2006, pp. 246 – 253

[27] Wang Y, Zixing C, Guanqi G, Zhou Y, Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B, Volume 37, 2007, No. 10.1109/TSMCB.2006.886164 , pp. 560 – 575

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.

Nadezda Stanarevic 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. Stanarevic participated in WSEAS conferences.

Nebojsa Bacanin received B.S. and M.S. in economics and computer science in 2006 and 2008 from Megatrend University of Belgrade and also M.S. in computer science in 2008 from 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. Bacanin participated in WSEAS conferences.