Enhanced Artificial Bee Colony Algorithm Performance

Nadezda STANAREVIC, Milan TUBA, Nebojsa BACANIN
Faculty of Computer Science
Megatrend University Belgrade
Bulevar umetnosti 29
SERBIA
srna@stanarevic.com, tubamilan@ptt.rs, nbacanin@megatrend.edu.rs

Abstract: - This paper presents a modified Artificial Bee Colony (ABC) algorithm for constrained problems. Original Karaboga’s ABC algorithm was for unconstrained problems only and modifications for constrained problems were introduced later. The performance of this modified algorithm was examined by Karaboga. We introduced additional modification since the ABC algorithm does not consider the initial population to be feasible. We added a “smart bee” (SB) which uses its historical memories for the location and quality of food sources. This modified SB algorithm was tested on standard benchmark functions for unconstrained optimization problems and proved to be better.

Key-Words: - Artificial Bee Colony, Constrained optimization, Metaheuristics, Swarm intelligence

1 Introduction
Nature inspired algorithms based on the social behavior of certain animals and insects can solve many complex problems such as the Travelling 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. In the past few years, the usage of meta-heuristic algorithms has increased in popularity. 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.

Classification can be made depending on the nature of phenomenon simulated by the algorithm. This type of classification mainly has two important groups of population based algorithms: evolutionary algorithms (EA) and swarm algorithms. The most popular among EA is genetic algorithm (GA). GA attempts to simulate the phenomenon of natural evolution. Term swarm is used in a general manner to refer to any restrained collection of interacting agents or individuals [3], [4]. Swarm intelligence is research branch that models the population of interacting agents. Swarm Intelligence systems are typically made up of a population of self-organized individuals interacting locally with one another and with their environment [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.

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, birds, fish, etc. Inspired by behavior of mentioned above real organisms, a lot of swarm intelligence algorithms have been developed. For example, Ant Colony Optimization (ACO) is a technique that is quite successful in solving many 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. While working from their nests to food source, ants deposit a substance called pheromone. Paths that contain more pheromone concentrations are chosen with higher probability by ants than those that contain lower pheromone concentrations.

Particle swarm optimization (PSO) algorithm simulates social behavior of bird flocking or fish
schooling. 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.

Several metaheuristics have been proposed to model the specific intelligent behavior of honey bee swarms [6], [7], [8]. The bee swarm intelligence was used in the development of artificial systems aimed at solving complex problems in traffic and transportation [9]. That algorithm is called Bee Colony Optimization Meta-heuristic (BCO), which is used for solving deterministic combinatorial problems, as well as combinatorial problems characterized by uncertainty [9]. Drias introduced a novel intelligent approach called Bees Swarm Optimization (BSO), which is inspired from the behavior of real bees. BSO is adapted for solving maximum weighted satisfiability (max-sat) problem.

In this paper, we present enhancements of the Artificial Bee Colony (ABC) algorithm proposed by Karaboga and Bastuk [7]. We also measure performance of this enhanced algorithm against Karaboga’s original work. ABC is one of algorithms that model bee’s interactions in nature.

2 ABC Algorithm
Several approaches have been proposed to model the specific intelligent behaviors of honey bee swarms. Artificial Bee Colony (ABC) is a relatively new member of swarm intelligence.

Karaboga has described the Artificial Bee Colony (ABC) algorithm based on the foraging behavior of honey bees for numerical optimization problems. Problems optimizations depend on a number of parameters and the choice of these parameters affects the performance.

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 [6]:

1. Initialize the population of solutions
2. Evaluate the population
3. Produce new solutions for the employed bees
4. Apply the greedy selection process
5. Calculate the probability values
6. Produce the new solutions for the onlookers
7. Apply the greedy selection process
8. Determine the abandoned solution for the scout, and replace it with a new randomly produced solution
9. 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 [7]. 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

$$ p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (1) $$

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

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

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 [7]. 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 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) [8]. 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 upper bounds): 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.

### 3 Constrained optimization

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. The CO problem can be represented as the following nonlinear programming problem [10]:

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

(3)

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

$$lb_i \leq x_i \leq ub_i, \quad 1 \leq i \leq n$$

(4)

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

$$g_j(x) \leq 0, \text{ for } j = 1, \ldots, q$$

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

(5)

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

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 [10]. On the other hand, stochastic optimization algorithms such as Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Particle Swarm Optimization (PSO) do not make such assumptions and they have been successfully applied for tackling constrained optimization problems during the past few years.

### 4 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 [6]. 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 first proposal, to extend the ABC algorithm [11] to constrained spaces, used a constraint handling technique originally proposed for a genetic algorithm by Deb [12]. 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 [11]. 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 [11].

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:
where \( k \in \{1, 2, \ldots, SN\} \) is randomly chosen index., \( x_{ij} \) is the variable \( j \) of the current food source, \( x_i \) is a randomly selected solution (different from \( x_{ij} \)), \( R_j \) 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 tresspassed bound. In our algorithm we have used a different mechanism from original ABC algorithm, based on the Kukkonen and Lampinen work [13]:

\[
U_{i,j} = \begin{cases} 
2^a lb_j - v_{ij}, & \text{if } v_{ij} < lb_j \\
2^a ub_j - v_{ij}, & \text{if } v_{ij} > ub_j \\
v_{i,j} & \text{otherwise}
\end{cases}
\]

where \( v_{ij} \) 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 [14]. 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.

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

### 5.1 Benchmark constrained optimization functions

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:

\[
g_i(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0
\]

\[
g_i(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0
\]

\[
g_i(x) = -8x_1 + x_{10} \leq 0
\]

\[
g_i(x) = -8x_2 + x_{11} \leq 0
\]

\[
g_i(x) = -8x_3 + x_{12} \leq 0
\]

\[
g_i(x) = -2x_1 - x_3 + x_{10} \leq 0
\]

\[
g_i(x) = -2x_2 + x_4 + x_{12} \leq 0
\]

\[
g_i(x) = -2x_2 - x_4 + x_{12} \leq 0
\]

Where the bounds are \( 0 \leq x_i \leq 1 \) \((i = 1, \ldots, 9)\), \( 0 \leq x_{i3} \leq 100 \) \((i = 10, 11, 12)\) and \( 0 \leq x_{i3} \leq 1 \). The global minimum is at \( x = (1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1, 3, 3, 3, 3, 3, 1) \) and \( f(x) = -15 \).

G4: Minimize:

\[
5.3578547x_1^2 + 0.8356891x_1x_3 + 37.293239x_1 - 40792.141
\]

Subject to:

\[
g_i(x) = 85.354407 + 0.0056858x_2x_3 + 0.0006262x_1x_3 - 0.00022053x_1x_3 - x_{13} \leq 0
\]

\[
g_i(x) = 85.354407 - 0.0056858x_3x_5 - 0.0006262x_1x_5 + 0.00022053x_1x_5 \leq 0
\]

\[
g_i(x) = 80.51249 + 0.0071317x_3x_5 + 0.00029955x_5 + 0.0021813x_5^2 - 110 \leq 0
\]

\[
g_i(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_3x_5 + 0.00019085x_5 + 0.00021813x_5^2 \leq 0
\]

\[
g_i(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_3x_5 - 0.00019085x_5 + 0.00021813x_5^2 + 20 \leq 0
\]

Where \( 78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45, 27 \leq x_3 \leq 45 \) \((i = 3, 4, 5)\). The optimum solution is \( f(x) = -50665.538672 \).

G6: Minimize:

\[
f(x) = (x_1 - 10)^3 + (x_2 - 20)^3
\]

Subject to:

\[
g_i(x) = -(x_i - 5)^2 - (x_i - 5)^2 + 100 \leq 0
\]

\[
g_i(x) = (x_1 - 6)^2 + (x_1 - 5)^2 - 82.81 \leq 0
\]

Where \( 13 \leq x_1 \leq 100 \) and \( 0 \leq x_2 \leq 100 \). The optimum solution is located at \( x = (14.095, 0.000, 0.000, 0.000, 0.000, 0.842), f(x) = -6961.813875580 \).
G13: Minimize:

\[ f(x) = e^{x_1x_2x_3x_4} \]

Subject to:

\[
g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 = 0 \\
g_2(x) = x_1x_3 + 5x_4x_5 = 0 \\
g_3(x) = x_1^2 + x_2^2 + 1 = 0
\]

Where \(-2.3 \leq x_i \leq 2.3 \) (\(i = 1, 2\)) and \(-3.2 \leq x_i \leq 3.2 \) (\(i = 3, 4, 5\)). The optimum solution is 
\[ x = (-1.71714224, 1.59572124, 1.82725024, -0.76365988, -0.76365986) \text{ where } f(x) = 0.0539415140418. \]

Control parameters of the ABC algorithm are: colony size, solution number, limit, maximum number of cycles and modification rate. In these experiments, the colony size was 40 and the maximum number of cycles was 6000. The value of "limit" is equal to \(SN*D\) where \(SN\) is the number of solutions and \(D\) is the dimension of the problem. Our algorithm was implemented using the parameters' values described in the Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colony size</td>
<td>NP</td>
<td>40</td>
</tr>
<tr>
<td>Solutions Number</td>
<td>SN</td>
<td>20</td>
</tr>
<tr>
<td>Maximum Cycle Number</td>
<td>maxCycle</td>
<td>6000</td>
</tr>
<tr>
<td>Limit</td>
<td>limit</td>
<td>(SN*D)</td>
</tr>
<tr>
<td>Modification Rate</td>
<td>MR</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 1. Parameters for the modified algorithm

A set of constrained numerical optimization problems was tested in the experiment. These test cases include objective functions of various types with different types of constraints. Basic function information are listed in Table 2. This set includes various forms of objective function such as linear, nonlinear cubic and quadratic.

<table>
<thead>
<tr>
<th>Fun.</th>
<th>Dim.</th>
<th>Type</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>13</td>
<td>Quadratic</td>
<td>-15</td>
</tr>
<tr>
<td>G4</td>
<td>5</td>
<td>Quadratic</td>
<td>-30665.5386</td>
</tr>
<tr>
<td>G6</td>
<td>2</td>
<td>Cubic</td>
<td>-6961.814</td>
</tr>
<tr>
<td>G13</td>
<td>5</td>
<td>Nonlinear</td>
<td>0.0539</td>
</tr>
</tbody>
</table>

Table 2. Set of constrained optimization test function

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>
<thead>
<tr>
<th>Pr.</th>
<th>Alg.</th>
<th>Worst</th>
<th>Best</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>ABC</td>
<td>-15</td>
<td>-15</td>
<td>-15</td>
</tr>
<tr>
<td></td>
<td>SB</td>
<td>-15</td>
<td>-15</td>
<td>-15</td>
</tr>
<tr>
<td>G4</td>
<td>ABC</td>
<td>-30665.539</td>
<td>-30665.539</td>
<td>-30665.539</td>
</tr>
<tr>
<td></td>
<td>SB</td>
<td>-30665.539</td>
<td>-30665.539</td>
<td>-30665.539</td>
</tr>
<tr>
<td>G6</td>
<td>ABC</td>
<td>-6961.805</td>
<td>-6961.814</td>
<td>-6961.813</td>
</tr>
<tr>
<td></td>
<td>SB</td>
<td>-6961.808</td>
<td>-6961.814</td>
<td>-6961.813</td>
</tr>
<tr>
<td>G13</td>
<td>ABC</td>
<td>1.0000</td>
<td>0.760</td>
<td>0.968</td>
</tr>
<tr>
<td></td>
<td>SB</td>
<td>0.183</td>
<td>0.054</td>
<td>0.105</td>
</tr>
</tbody>
</table>

Table 3. Statistical results obtained by ABC and ABC-SB algorithms on test functions

6 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 ABC-SB was tested on four constrained optimization problems: quadratic, cubic, linear and nonlinear. The results obtained by the modified ABC algorithms for constrained optimization problems are quite satisfactory. In all test problems, the two variants of ABC algorithms exhibited similar results. Table 3, gives the summary of the obtained results using Karaboga ABC algorithm for constrained functions and ABC algorithm with smart bee (ABC-SB) respectively. Our proposed algorithm had equal results in G1, G4, G6 and better result for G13. 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.

Future work will include investigation of the ABC-SB 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 ABC algorithm can be also tested for real engineering problems existing in the literature and compared with other algorithms.
References:


