A fast artificial bee colony algorithm variant for continuous global optimization problems

George Anescu[1]

Since its creation in 2005 by D. Karaboga the ABC algorithm proved to be very effective in approaching a wide variety of research optimization problems. However, some drawbacks were also experienced related mainly to a poor exploitation capability (which makes the algorithm relatively slow) and poor success rates when highly non-linear optimization problems with unstructured modes are approached. In order to improve the performance of the ABC algorithm, in both efficiency and success rate, the paper presents a set of proposed enhancements to the original ABC algorithm. The novel proposed ABC variant, Fast ABC (F-ABC), was tested against two known variants of ABC, the original algorithm proposed by D. Karaboga ([1]), and an improved variant, Gbest-guided Artificial Bee Colony (GABC) ([2]). The testing was conducted by employing an original testing methodology over a set of 11 scalable, multimodal, continuous optimization functions (10 unconstrained and 1 constrained) with known global solutions. The novel F-ABC algorithm clearly outperformed the older variants in both efficiency and success rate over the test functions which present unstructured modes, while for the remaining test functions the results were mixed.

Keywords: Optimization, Continuous Global Optimization Problem (CGOP), Swarm Intelligence (SI), Artificial Bee Colony Algorithm (ABC), Gbest-guided Artificial Bee Colony Algorithm (GABC), Fast Artificial Bee Colony Algorithm (F-ABC), Keane's Bump Function

MSC2010: 68T 20, 68W 10, 90C 26, 90C 56, 90C 59.

1. Introduction

Due to the difficulty of solving some real world optimization problems from a variety of scientific and engineering fields, in the last decades some modern optimization algorithms inspired from nature were developed. A special class of such nature inspired optimization algorithms is represented by the Swarm Intelligence (*SI*) algorithms. *SI* can be briefly defined as the collective intelligent behavior of decentralized and self-organized swarms of individuals (bird flocks, fish schools and colonies of social insects such as termites, ants and bees). Several algorithms have been developed inspired from different intelligent behaviors of honey bee swarms, among which Artificial

[1]PhD, Department of Power Plant Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: george.anescu@gmail.com

Bee Colony (ABC), originally published by D. Karaboga in 2007 ([1]), is the one which has been most widely studied on and applied to solve some real world optimization problems. The ABC algorithm presents many advantages compared to the traditional optimization methods and modern meta-heuristic methods: does not assume continuity and differentiability of the objective function (derivative free), needs fewer control parameters (parameter free), has a simple design which it is easy to implement, can be easily modified and hybridized with other meta-heuristic algorithms. From the application perspective ABC has been tailored successfully to solve a wide variety of discrete, continuous (constrained and unconstrained) and combinatorial optimization problems in a wide variety o fields: decision making, engineering design, pattern recognition, image processing, scheduling, protein structure prediction, etc. Two comprehensive surveys concerning the state of the art in the ABC algorithm research and its applications are presented in papers [3] and [4].

From the numerical performance perspective the ABC algorithm was compared to many other meta-heuristic population-based algorithms and the numerical results showed that the algorithm is competitive, although there was room for enhancements. The main two problems (which will also be emphasized in the experimental results section of the present paper) are related to a poor exploitation capability (which makes the algorithm relatively slow) and poor success rates when optimization problems with a highly non-regular structure of the modes are approached. The main goal of the present paper is to propose a variant of the ABC algorithm which is able to overcome the mentioned problems.

The rest of this paper is organized as follows: Section 2 shortly presents the Continuous Global Optimization Problem ($CGOP$); Section 3 presents the Deb's Rules for Constraints Handling as they were adapted for the F-ABC method; Section 4 presents the original ABC algorithm and the modifications implemented in the $GABC$ variant; Section 5 presents the enhancements introduced in the design of the new F-ABC variant; Section 6 presents the set of test optimization problems used in the testing experiments and the obtained comparative results for the novel F-ABC method, the original ABC algorithm and the $GABC$ variant; and finally, Section 7 summarizes and draws some conclusions.

2. Continuous Global Optimization Problem ($CGOP$)

The Continuous Global Optimization Problem ($CGOP$) is generally formulated as ([5]):

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in D \end{aligned} \tag{1}$$

with

$$D = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u};\ and\ \ g_i(\mathbf{x}) \leq 0,\ \ i = 1, \ldots, G; \\ and\ \ h_j(\mathbf{x}) = 0,\ \ j = 1, \ldots, H\} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a real n-dimensional vector of decision variables ($\mathbf{x} = (x_1, x_2, \ldots, x_n)$), $f : \mathbb{R}^n \to \mathbb{R}$ is the continuous objective function, $D \subset \mathbb{R}^n$ is the non-empty set of feasible decisions (a proper subset of $\mathbb{R}^n$), $\mathbf{l}$ and $\mathbf{u}$ are explicit, finite (component-wise) lower and upper bounds on $\mathbf{x}$, $g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, G$ is a finite collection of continuous inequality constraint functions, and $h_j : \mathbb{R}^n \to \mathbb{R},\ j = 1, \ldots, H$ is a finite collection of continuous equality constraint functions. No other additional supposition is made on the *CGOP* problem and it is assumed that no additional knowledge about the collections of real continuous functions can be obtained, in this way treating the *CGOP* problem as a black box, i.e. for any point $\mathbf{x}$ in the boxed domain $\{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$ it is assumed the ability to calculate the values of the functions $f(\mathbf{x})$, $g_i(\mathbf{x}),\ \ i = 1, \ldots, G$, $h_j(\mathbf{x}),\ \ j = 1, \ldots, H$, but nothing more.

3. Deb's Rules for Constraints Handling

The Deb's rules ([6]) offer a methodology to efficiently handle the constraints in constrained optimization problems. The presentation in this section is adapted from [7] with the notations from equations (1) and (2). The inequality constraints that satisfy $g_i(\mathbf{x}) = 0,\ \ i = 1, \ldots, G$ at the global optimum solution are called active constraints. All equality constraints are active constraints. The equality constraints can be transformed into the inequality form and can be combined with other inequality constraints as the auxiliary functions $\tilde{g}_i(\mathbf{x})$:

$$\tilde{g}_i(\mathbf{x}) = \begin{cases} \max[0, g_i(\mathbf{x})],\ \ i = 1, \ldots, G \\ \max[0, |h_{i-G}(\mathbf{x})| - \delta], i = G+1, \ldots, G+H. \end{cases} \tag{3}$$

where δ is a tolerance parameter for the equality constraints. Therefore, the objective is to minimize the objective function $f(\mathbf{x})$ such that the obtained optimal solution satisfies all the inequality constraints $\tilde{g}_i(\mathbf{x}) \leq 0$ as active constraints. The overall constraint violation for an infeasible solution is a weighted mean of all the constraints expressed as:

$$v(\mathbf{x}) = \frac{\sum_{i=1}^{G+H} w_i \tilde{g}_i(\mathbf{x})}{\sum_{i=1}^{G+H} w_i} \tag{4}$$

where $w_i = 1/\tilde{g}_{max,i}$ is a weight parameter and $\tilde{g}_{max,i}$ is the maximum violation of constraint $\tilde{g}_i(\mathbf{x})$ obtained so far (up to the current computing

iteration). $\tilde{g}_{max,i}$ varies during the optimization process in order to balance the contribution of every constraint in the problem irrespective of their differing numerical ranges. There are a number of constraint handling techniques based on constraint violation, the one used here being the Superiority of Feasible Solutions (SF) technique. In SF the Deb's rules ([6]) are applied when comparing two solutions $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$. According to Deb's rules $\mathbf{x}_{i_1}$ is regarded superior to $\mathbf{x}_{i_2}$ when:

- $\mathbf{x}_{i_1}$ is feasible and $\mathbf{x}_{i_2}$ is not feasible.
- $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are both feasible and $\mathbf{x}_{i_1}$ has a smaller objective value than $\mathbf{x}_{i_2}$.
- $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are both infeasible, but $\mathbf{x}_{i_1}$ has a smaller overall constraint violation than $\mathbf{x}_{i_2}$.

Therefore, in SF the feasible solutions are always considered better than the infeasible ones. Two infeasible solutions are compared based on their overall constraint violations only, while two feasible solutions are compared based on their objective function values only. The comparison of infeasible solutions based on the overall constraint violation aims to push the infeasible solutions toward the feasible regions, while the comparison of two feasible solutions based on the objective function value improves the overall solution.

In order to be able to correctly compare the infeasible solutions which are near feasible regions the following modification to the constraint violation was proposed:

$$v(\mathbf{x}) = \frac{\sum_{i=1}^{G+H} w_i \tilde{g}_i(\mathbf{x})}{\sum_{i=1}^{G+H} w_i} + G_{ns} + H_{ns} \tag{5}$$

where G_{ns} ($G_{ns} \leq G$) is the number of not satisfied inequality constraints, and H_{ns} ($H_{ns} \leq H$) is the number of not satisfied equality constraints.

Another important proposed improvement for the handling of the equality constraints is to make the δ tolerance parameter dependent on the current iteration index k:

$$\delta = k(\delta_2 - \delta_1)/iter_{max} + \delta_1 \tag{6}$$

where δ_1 is the initial tolerance parameter (at $k = 0$), δ_2 is the final tolerance parameter (at $k = iter_{max}$) with $\delta_1 \gg \delta_2$, and $iter_{max}$ is the maximum iteration count.

4. Artificial Bee Colony (*ABC*) Optimization

The *ABC* algorithm is a *SI* meta-heuristic algorithm based on the model proposed by D. Karaboga in 2007 ([1]) for the foraging behavior of honey bee colonies. In the *ABC* model the colony of artificial bees contains three groups (types) of bees: employed bees, onlooker bees and scouts. A bee searching around the food source visited by itself previously (its position at the previous iteration) is called an employed bee, a bee waiting in the dance area for making the decision to choose a food source is called an onlooker bee (the bees' dance is assumed as the method of communication), and a bee carrying out random search is called a scout bee. The main steps of the algorithm are given below:

Step 1: Initialization;
while (true)
    Step 2: Check termination conditions, break the
        loop if any applies;
    Step 3: Employed Bees Phase;
    Step 4: Onlooker Bees Phase;
    Step 5: Scout Bees Phase;
end while

The method's parameters are: N the number of employed bees and onlooker bees, ϵ the required precision, *limit* the stagnation count and $iter_{max}$ the maximum number of iterations. The N employed bees and N onlooker bees are represented as vector positions $\mathbf{x}_i^e$ and respectively $\mathbf{x}_i^o$, $i = 1, \ldots, N$ in the the limiting box (hyper-rectangle) defined by the lower limits l_j, $j = 1, \ldots, n$ and higher limits h_j, $j = 1, \ldots, n$ ($l_j < h_j$).

Each time an employed bee moves to a better position it sets a food source and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. Food sources around which the searching takes place are considered only the positions of the employed bees, while possible solutions are considered all the bee positions (employed bees, onlooker bees, scout bees).

At initialization the employed bees and the onlooker bees take random values in the limiting box:

$$x_{i,j}^e = l_j + rnd_{i,j}^e \times (u_j - l_j), \;\; i = 1, \ldots, N, \;\; j = 1, \ldots, n \tag{7}$$

$$x_{i,j}^o = l_j + rnd_{i,j}^o \times (u_j - l_j), \;\; i = 1, \ldots, N, \;\; j = 1, \ldots, n \tag{8}$$

where $rnd_{i,j}^e$ and $rnd_{i,j}^o$ are uniformly generated pseudo-random numbers in the interval $[0, 1)$. The iteration index is initialized to $k = 0$. The objective function $f(\mathbf{x})$ is evaluated in the current bees positions $\mathbf{x}_i^e(0), \mathbf{x}_i^o(0)$, $i = 1, \ldots, N$: $f_i^e(0) = f(\mathbf{x}_i^e(0)), f_i^o(0) = f(\mathbf{x}_i^o(0)), \;\; i = 1, \ldots, N$.

Each iteration of the optimization method consists of three phases: sending the employed bees onto the food sources and then measuring their nectar amounts (*Employed Bees Phase*); selecting the food sources by the onlooker bees after sharing the information of the employed bees and measuring the nectar amount of the food sources (*Onlooker Bees Phase*); determining the scout bees and then sending them onto possible food sources (*Scout Bees Phase*).

During the *Employed Bees Phase* each employed bee goes to the food source area visited at the previous iteration (since that food source exists in its memory), and chooses a new candidate food source by means of visual information in the neighborhood of the current one. The visual information is based on the comparison of food source positions. For each employed bee with index i another employed bee with index $m \in \{1, \ldots, N\}$, $m \neq i$, is selected in a discrete uniform pseudo-random manner. In order to produce a candidate food source position, the ABC algorithm uses the following equation:

$$x'^{e}_{i,j}(k+1) = x^{e}_{i,j}(k) + r^{e}_{i,j} \times (x^{e}_{m,j}(k) - x^{e}_{i,j}(k)) \tag{9}$$

where $j \in \{1, \ldots, n\}$, is an uniform pseudo-randomly selected index and $r^{e}_{i,j}$ are pseudo-random numbers uniformly sampled from the $[-1, 1)$ real interval. Equation (9) controls the generation of a neighbor food source position around $\mathbf{x}^{e}_{i}(k)$ and the modification represents the comparison of the neighbor food positions visually by the bee. Note that only one component of $\mathbf{x}'^{e}_{i}$, the one with index j, is different from the components of $\mathbf{x}^{e}_{i}$. The new candidate food source position is taken as the current food source position only if it is better than the old food source position: if $f(\mathbf{x}'^{e}_{i}(k+1)) < f(\mathbf{x}^{e}_{i}(k))$ then $\mathbf{x}^{e}_{i}(k+1) := \mathbf{x}'^{e}_{i}(k+1)$. Equation (9) shows that as the difference between the position components $x^{e}_{i,j}(k)$ and $x^{e}_{m,j}(k)$ decreases, the perturbation on the position $\mathbf{x}^{e}_{i,j}(k)$ decreases too. Thus, as the search approaches the optimal solution in the search space, the perturbation is adaptively reduced.

During the *Onlooker Bees Phase* first the employed bees go into the hive and share the nectar information of the food sources with the onlooker bees waiting in the dance area within the hive. After sharing the nectar information the food sources are given by the new positions of the employed bees calculated at the current iteration. In order to choose a food source an onlooker bee needs a selection mechanism. An onlooker bee prefers a food source area depending on the nectar information distributed by the employed bees in the dance area. As the nectar amount of a food source increases, the probability with which that food source is chosen by an onlooker bee increases, too. Hence, the dance of employed bees carrying higher nectar recruits the onlooker bees for the food source areas with higher nectar amount. The selection mechanism proposed originally was the roulette wheel selection (widely applied in Genetic Algorithms, see [8]), but in the present implementation a ternary tournament selection (also widely applied in Genetic Algorithms, see [8]) was preferred

considering that Deb's Rules are used for constraints handling and that they provide only the possibility to compare solutions, but they do not provide the possibility to precisely determine the selection probabilities needed for roulette wheel selection. Let's denote by l the index of the food source selected based on the adopted selection mechanism. After arriving at the selected area l the onlooker bee chooses a candidate food source in the neighborhood of the selected one depending on the visual information. In order to produce a candidate onlooker position the ABC algorithm uses the following expression:

$$x'^{o}_{i,j'}(k+1) = x^{e}_{l,j'}(k+1) + r^{o}_{i,j'} \times (x^{e}_{m',j'}(k+1) - x^{e}_{l,j'}(k+1)) \qquad (10)$$

where $m' \in \{1, \ldots, N\}$, $m' \neq l$, and $j' \in \{1, \ldots, n\}$, are discretely uniformly pseudo-randomly sampled indexes and $r^{o}_{i,j'}$ are pseudo-random numbers uniformly sampled from the $[-1, 1)$ real interval. Note that only one component of $\mathbf{x}'^{o}_{i}$, the one with index j', is different from the components of $\mathbf{x}^{e}_{l}$. The new candidate onlooker position is taken as the current onlooker position only if it is better than the old onlooker position: if $f(\mathbf{x}'^{o}_{i}(k+1)) < f(\mathbf{x}^{o}_{i}(k))$ then $\mathbf{x}^{o}_{i}(k+1) := \mathbf{x}'^{o}_{i}(k+1)$. Equation (10) shows that as the difference between the component positions $x^{e}_{l,j'}(k)$ and $x^{e}_{m',j'}(k)$ decreases, the perturbation on the position $\mathbf{x}^{e}_{l}(k+1)$ decreases too. Thus, as the search approaches the optimum solution in the search space, the perturbation is adaptively reduced.

In the ABC algorithm, if a food source cannot be improved further over a predetermined number of *limit* iterations (stagnation count), then that food source is abandoned. Originally the stagnation count was suggested $limit = n \times N$. The food source whose nectar is abandoned is replaced with a new food source by the scouts (note that only the employed bees can become scouts). This is simulated by randomly generating a position in the search space and replacing the abandoned one with it. At each iteration during the *Scout Bees Phase* at most one scout goes outside for searching a new food source, the one with the highest stagnation count, but only if it is higher than *limit*.

A first termination condition is defined when the current iteration index k attains the maximum number of iterations $iter_{max}$. A second termination condition is defined when the diameter of the current onlooker bees swarm becomes less than the required precision ϵ:

$$d(k) = \left(\sum_{j=1}^{n} (d_j(k))^2 \right)^{\frac{1}{2}} < \epsilon \qquad (11)$$

where the overall population diameter $d(k)$ is calculated according to the Euclidian distance, and the diameters on each dimension are calculated as the maximum absolute difference between two position values on that dimension over all the onlooker bees in the population:

$$d_j(k) = \max_{1 \le i_1, i_2 \le N, i_1 \ne i_2} \{|x^o_{i_1,j}(k) - x^o_{i_2,j}(k)|\}, \quad j = 1, 2, \ldots, n \tag{12}$$

A still further termination condition is defined when a flat region is detected. It can appear when the objective function $f(\mathbf{x})$ depends only on a subset of its decision variables, and it can be easily checked as:

$$f^o_{max}(k) - f^o_{min}(k) < \epsilon_f \tag{13}$$

with

$$f^o_{max}(k) = \max_{1 \le i_1 \le N} \{f^o_{i_1}(k)\} \tag{14}$$

$$f^o_{min}(k) = \min_{1 \le i_2 \le N} \{f^o_{i_2}(k)\} \tag{15}$$

where $f^e_i(k) = f(\mathbf{x}^e_i(k))$, $f^o_i(k) = f(\mathbf{x}^o_i(k))$, $i = 1, \ldots, N$ and ϵ_f is a very small value. If any of the termination conditions is satisfied then the iterative process is stopped (the loop is broken) and the onlooker bee position which gives $f^o_{min}(k)$ (positioned in $\mathbf{x}^o_{min}(k)$) is taken as the solution of the global optimization problem. Otherwise the iteration index is incremented to $k + 1$ and the computation continues to the next iteration.

Inspired by *PSO* ([9]) the *Gbest-guided ABC* (*GABC*) method ([2]) improves the original *ABC* method by taking advantage of the information of the global best (*gbest*) solution to guide the search of candidate solutions in order to improve the exploitation. The equations (9) and (10) are modified as follows:

$$\begin{aligned} x'^e_{i,j}(k+1) = x^e_{i,j}(k) + r^e_{i,j} \times (x^e_{m,j}(k) - x^e_{i,j}(k)) + \\ + C \times r1^e_{i,j} \times (x^{gbest}_j(k) - x^e_{i,j}(k)) \end{aligned} \tag{16}$$

$$\begin{aligned} x'^o_{i,j'}(k+1) = x^e_{l,j'}(k+1) + r^o_{i,j'} \times (x^e_{m',j'}(k+1) - x^e_{l,j'}(k+1)) + \\ + C \times r1^o_{i,j'} \times (x^{gbest}_{j'}(k) - x^e_{l,j'}(k+1)) \end{aligned} \tag{17}$$

where $\mathbf{x}^{gbest}(k)$ is the current global best food source position (as determined at iteration k), $r1^e_{i,j}$ and $r1^o_{i,j'}$ are pseudo-random numbers uniformly sampled from the $[0, 1)$ real interval and $C > 0$ is a real positive constant. The global best food source is updated at each iteration. Experiments conducted in ([2]) showed that the best results are obtained when taking $C = 1.5$.

5. The *Fast ABC* Algorithm

In order to improve the performance of the *ABC* optimization algorithm, for both speed and success rate, a set of enhancements were designed, implemented and tested, the new resulting optimization algorithm being named *Fast ABC* (*F-ABC*). The *F-ABC* algorithm is identical, in respect to the general algorithmic steps, to the original *ABC* algorithm, but differs from it in some detail aspects:

- In both the *Employed Bees Phase* and *Onlooker Bees Phase* in equations (9) and (10) the pseudo-random numbers $r^e_{i,j}$ and $r^o_{i,j'}$ are now uniformly sampled from the $[-0.5, 1.5)$ real interval;

- There are differences in the determination of indices l and m' intervening in equation (10) in *Onlooker Bees Phase*. The index l of the selected food source is determined by using a binary tournament selection mechanism, while the index m' is selected in a discrete uniform pseudo-random manner from the set $s_l = \{m' : 1 \leq m' \leq N;\ m' \neq l;\ and\ f(\mathbf{x}^e_{m'}) < f(\mathbf{x}^e_l)\}$. In other words, the selected food source l, instead of interacting with a pseudo-randomly selected different food source, it interacts with a better pseudo-randomly selected different food source. This topology used in the selection of m' is called *random greedy topology* (see also [10]).

- In the original *ABC* algorithm in the *Employed Bees Phase* and *Onlooker Bees Phase* equations (9) and respectively (10) are applied only to one pseudo-randomly determined dimension, j respectively j'. In order to make the algorithm able to cope with highly non-linear objective functions, there is a need to give chances to change to all the dimensions. Therefore to any food source is associated a weight, w_i, which will be used as a limit probability and which is calculated for each food source using the formula:

$$w_i = \frac{f(\mathbf{x}^e_i) - f_{best}}{f_{worst} - f_{best}},\ \ i = 1, \ldots, N \tag{18}$$

where

$$f_{best} = \min_{1 \leq i \leq N} f(\mathbf{x}^e_i),\ \ f_{worst} = \max_{1 \leq i \leq N} f(\mathbf{x}^e_i) \tag{19}$$

Note that for constrained optimization problems, in the context of applying the Deb's Rules for handling the constraints (as presented in Section 3), the weights are defined as $w_i = i/N$, with the assumption that the food sources are sorted in the increasing order (from the best to the worst), and the sorting is done according to the Deb's Rules, in this way the index i also being the rank of the food source. As defined above, the weights are in the $[0, 1]$ real interval and therefore it is safe to use them as limit probabilities. In both the *Employed Bees Phase* and *Onlooker Bees Phase* the candidate food source position are determined by applying equations (9) and respectively (10) on one dimension uniformly pseudo-randomly determined between 1 and n (discrete

selection with equal probabilities) and on the remaining dimensions only after successfully passing a probabilistic test: $rnd() < w_i$. Through the defined probabilistic test there is possible to induce a more exploratory behavior at the food sources with large weights (poor fitness) and a more exploitative behavior at the food sources with small weights (good fitness values, close to the best value).

- In the *Scout Bees Phase* it was found that the originally suggested formula of the stagnation count parameter *limit* gives a too high value and experimentally were tried different values as multiples of the search space dimension n. The experiments showed that balanced results can be obtained, from the efficiency perspective over the set of test optimization problems, with $limit = 4 \times n$ (see also [10]), which was proposed as an appropriate value.

- The second modification in the *Scout Bees Phase* consisted in determining the reset position of a scout bee in a uniformly pseudo-random manner inside the current hyper-rectangle that is limiting the food sources (also recommended in [10]) with lower and upper limits $\mathbf{l}(k+1)$ and respectively $\mathbf{u}(k+1)$ determined on each dimension (the notations being inclusive for the initial box bounds $\mathbf{l}(0) = \mathbf{l}$ and $\mathbf{u}(0) = \mathbf{u}$):

$$l_j(k+1) = \min_{1 \le i \le N} \{x_{i,j}^e(k+1)\} \tag{20}$$

$$u_j(k+1) = \max_{1 \le i \le N} \{x_{i,j}^e(k+1)\} \tag{21}$$

This modification demonstrated an obvious further improvement in efficiency for some of the test functions.

6. Testing and Results

The purpose of the testing phase was to prove that the new proposed *F-ABC* algorithm is competitive when compared to other existing *ABC* algorithm variants. For comparison two *ABC* variants were chosen: the original *ABC* algorithm and an improved version of it, *GABC* (both presented in Section 4 of the present paper).

In order to conduct the tests, an appropriate testing methodology was devised (see also [10], [11]). When the quality of an optimization method is estimated two (often conflicting) characteristics are of interest: a small number of function evaluations (NFE) and a high success rate (SR). For test functions with known solutions the success can be simply defined as the achievement of an absolute or relative precision tolerance to the known solutions. By fixing the tolerance and choosing $iter_{max}$ high enough so that this in never attained before the tolerance is attained, it is easy to measure the SR and average NFE to success ($\mu(NFE)$). There are other testing methodologies frequently applied in practice, like for example based on fixing NFE and reporting the best, the worst and the median results obtained after a number o runs, but in author's opinion such methodologies are not recognizing the importance of

Table 1

***F-ABC* versus *ABC* and *GABC*, n=10, runs=100, tolerance=0.1%, N=100**

Fn	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	100%	61940	100%	93774	100%	50350
f_2	100%	24704	100%	60558	100%	33708
f_3	100%	40476	84%	187084	99%	208131
f_4	100%	38662	98%	90477	100%	64486
f_5	100%	26541	100%	63672	100%	26188
f_6	100%	11660	100%	34176	100%	14600
f_7	100%	110778	100%	258828	100%	103454
f_8	100%	54672	100%	98842	100%	61244
f_9	100%	27976	100%	96998	100%	48602
f_{10}	100%	220334	99%	645614	100%	566790
f_{11}^{**}	100%	53621	0%	N/A	89%	232420

Table 2

***F-ABC* versus *ABC* and *GABC*, n=20, runs=100, tolerance=0.1%, N=100**

Fn	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	99%	164930	100%	250486	100%	129496
f_2	100%	40668	100%	144244	100%	83410
f_3	84%	105366	1%	125600	2%	918816
f_4	100%	44094	100%	130602	100%	80360
f_5	100%	58314	100%	228518	100%	71392
f_6	100%	22754	100%	97352	100%	43862
f_7	100%	507039	18%	1232733	26%	872115
f_8	66%	360543	100%	446884	100%	280196
f_9	100%	45974	100%	215866	100%	111490
f_{10}	66%	897662	0%	N/A	0%	N/A
f_{10}^{*}	N/A	N/A	33%	1532945	78%	1232751
f_{11}^{**}	94%	254994	0%	N/A	0%	N/A

success rate and are concealing it from reporting. A very efficient method (with a fast convergence) but with a low success rate cannot be considered better than a less efficient method, but with a high success rate, because the former may need many repeated runs in order to obtain the correct result, while the later may get the correct result in less runs, which can entail a larger overall NFE (obtaining by summation) for the former compared to the later.

A test bed of 11 known scalable multimodal optimization functions (10 unconstrained and 1 constrained, see [12], [13], [14], [15]) was used for the tests run on the three compared optimization methods:

- *Rastrigin's Function* - highly multimodal with the locations of the minima regularly distributed, global minimum value of 0 at $(0, 0, \ldots, 0)$

Table 3

***F-ABC* versus *ABC* and *GABC*, n=30, runs=100, tolerance=0.1%, N=100**

Fn	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	91%	289129	100%	436558	100%	218932
f_2	100%	54936	100%	235984	100%	137964
f_3	29%	180861	0%	N/A	0%	N/A
f_4	100%	55412	100%	195762	100%	119122
f_5	100%	91539	100%	473928	100%	124644
f_6	100%	37790	100%	120422	100%	93148
f_7	94%	1153975	0%	N/A	0%	N/A
f_8	12%	1131014	100%	920902	100%	595538
f_9	100%	61606	100%	339068	100%	177614
f_{10}	20%	1866919	0%	N/A	0%	N/A
f_{10}^*	N/A	N/A	0%	N/A	6%	2579333
f_{11}^{**}	42%	403339	0%	N/A	0%	N/A

$$f_1(\mathbf{x}) = 10n + \sum_{j=1}^{n}[x_j^2 - 10\cos(2\pi x_j)], \\ -5.12 \leq x_j \leq 5.12, \;\; j = 1, \ldots, n \tag{22}$$

- *Alpine 1 Function* - highly multimodal, global minimum value of 0 at $(0, 0, \ldots, 0)$

$$f_2(\mathbf{x}) = \sum_{j=1}^{n}(|x_j \sin(x_j)| + 0.1|x_j|), \\ -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{23}$$

- *Alpine 2 Function* - highly multimodal, global maximum value of 2.808^n at $(7.917, 7.917, \ldots, 7.917)$

$$f_3(\mathbf{x}) = \prod_{j=1}^{n}\sqrt{x_j}\sin(x_j), \\ 0 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{24}$$

- *Griewangk's Function* - many widespread local minima regularly distributed with the global minimum of 0 at $(0, 0, \ldots, 0)$

$$f_4(\mathbf{x}) = \frac{1}{4000}\sum_{j=1}^{n}x_j^2 - \prod_{j=1}^{n}\cos\left(\frac{x_j}{\sqrt{j}}\right) + 1, \\ -100 \leq x_j \leq 100, \;\; j = 1, \ldots, n \tag{25}$$

- *Schwefel's Function* - many widespread local minima distributed at distance from the origin with the global minimum of -418.9829 at $(420.9687, 420.9687, \ldots, 420.9687)$

$$f_5(\mathbf{x}) = -\frac{1}{n}\sum_{j=1}^{n} x_j \sin\left(\sqrt{|x_j|}\right), \quad -500 \leq x_j \leq 500, \;\; j = 1, \ldots, n \tag{26}$$

- *Paviani's Function* - many local minima with the global minimum of -45.77847 at $(9.351, 9.351, \ldots, 9.351)$ for $n = 10$, -9549.89061 at $(9.9658, 9.9658, \ldots, 9.9658)$ for $n = 20$, and respectively -99786.45525 at $(9.9993, 9.9993, \ldots, 9.9993)$ for $n = 30$:

$$f_6(\mathbf{x}) = \sum_{j=1}^{n-1} \left[\log(x_j - 2)^2 + \log(10 - x_j)^2\right] - \left(\prod_{j=1}^{n-1} x_j\right)^{0.2}, \quad 2.0001 \leq x_j \leq 9.9999, \;\; j = 1, \ldots, n \tag{27}$$

- *Expanded Schaffer's Function* - many local minima with the global minimum of 0 at $(0, 0, \ldots, 0)$

$$f_7(\mathbf{x}) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_n, x_1), \quad -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{28}$$

where

$$g(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{1 + 0.001(x^2 + y^2)^2} \tag{29}$$

- *Michaelwitz's Function* - highly multimodal with global minimum of: -0.966015 for $n = 10$, -0.9818507 for $n = 20$, and respectively -0.9876481 for $n = 30$:

$$f_8(\mathbf{x}) = -\frac{1}{n}\sum_{j=1}^{n} \sin(x_j) \sin^{2m}\left(\frac{j x_j^2}{\pi}\right), \quad m = 10, \;\; 0 \leq x_j \leq \pi, \;\; j = 1, \ldots, n \tag{30}$$

- *Ackley's Function* - highly multimodal with global minimum of 0 at $(0, 0, \ldots, 0)$:

$$f_9(\mathbf{x}) = 20 + e - 20e^{-0.2\left(\frac{1}{n}\sum_{j=1}^{n} x_j^2\right)^{1/2}} - e^{\frac{1}{n}\sum_{j=1}^{n} cos(2\pi x_j)}, \\ -30 \leq x_j \leq 30, \;\; j = 1, \ldots, n \tag{31}$$

- *Non-Linear Function* - highly multimodal, many global minima of 0:

$$f_{10}(\mathbf{x}) = n - 1 + \sum_{j=1}^{n-1} \cos\left(\frac{|x_{j+1} - x_j|}{|x_j + x_{j+1}| + 10^{-10}}\right), \\ -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n \tag{32}$$

- *Keane's Bump Function* - highly multimodal open constrained problem, best known global minima for different search space dimensions were used during testing (-0.747310362 for $n = 10$, -0.803619104 for $n = 20$, and respectively -0.821878040697 for $n = 30$):

$$f_{11}(\mathbf{x}) = -\left|\left\{\sum_{j=1}^{n} \cos(x_j)^4 - 2\prod_{j=1}^{n} \cos(x_j)^2\right\} / \left(\sum_{j=1}^{n} j x_j^2\right)^{0.5}\right|, \\ g_1(\mathbf{x}) = 0.75 - \prod_{j=1}^{n} x_j \leq 0.0, \\ g_2(\mathbf{x}) = \sum_{j=1}^{n} x_j - 7.5n \leq 0.0, \\ 0.0 \leq x_j \leq 10.0, \;\; j = 1, \ldots, n \tag{33}$$

All the test functions with global optima in origin were shifted, considering that the origin is favored by *ABC* type methods and this peculiarity can interfere in the results. In the tables the star sign (*) in the first column signifies that the test was repeated with increased tolerance ($tolerance = 1\%$), while the double star sign (**) signifies that a different value of the N parameter was used (it applied only to f_{11} and it was $N = 200$ for $n = 10, n = 20$ and $N = 400$ for $n = 30$).

Table 1 presents the comparative testing results obtained for $n = 10$. From the success rate perspective, it can be observed that *F-ABC* obtained the maximum percentage for all the test functions, while *ABC* was not able to solve f_{11} and showed an inception of problems with f_3, f_4 and f_{10}, while *GABC* showed an inception of problems with f_3 and f_{11}. Nevertheless, both *ABC* and *GABC* provided excellent results for the test functions they were able to solve. From the efficiency perspective, *F-ABC* clearly surpassed *ABC*

for all the test functions, while compared to *GABC* the results were mixed: *F-ABC* was faster for f_2, f_3, f_4, f_6, f_8, f_9, f_{10} and f_{11}, but slower for f_1, f_5 and f_7.

Table 2 presents the comparative testing results obtained for $n = 20$. From the success rate perspective, *F-ABC* showed difficulties with f_1, f_3, f_8, f_{10} and f_{11}, but still remained the only method able to solve all the problems in the given test conditions. *ABC* was not able to solve f_{10} and f_{11}, while the success rates for f_3 and f_7 deteriorated substantially. *GABC* showed similar behavior for exactly the same functions. From the efficiency perspective, *F-ABC* again surpassed *ABC* for all the test functions, while compared to *GABC* the results were mixed: *F-ABC* was faster for f_2, f_3, f_4, f_5, f_6, f_7 and f_9 but slower for f_1 and f_8.

Table 3 presents the comparative testing results obtained for $n = 30$. From the success rate perspective, *F-ABC* showed a substantial deterioration for f_1, f_3, f_7, f_8, f_{10} and f_{11}, but again was the only method able to solve all the problems in the given testing conditions. *ABC* was not able to solve f_3, f_7, f_{10} and f_{11}, but provided the maximum percentage for the remaining functions it was able to solve. *GABC* showed similar behavior for exactly the same functions. From the efficiency perspective, *F-ABC* was surpassed by *ABC* for f_8, but it was faster for the rest of functions that *ABC* was able to solve, while compared to *GABC*, the results were again mixed: *F-ABC* was surpassed for f_1 and f_8, but it was faster for the rest of functions that *ABC* was able to solve.

7. **Conclusions**

The paper proposed a novel global optimization method, *F-ABC*, as a variant of the known *ABC* method, designed to eliminate some of the drawbacks experienced by *ABC* related mainly to a poor exploitation capability (which makes the algorithm relatively slow) and poor success rates when approaching optimization problems with a highly non-regular structure of the modes. The novel *F-ABC* algorithm was tested against two other *ABC* variants, the original one proposed by D. Karaboga ([1]), and an improved variant, *GABC*, proposed by G. Zhu ([2]). The testing was conducted by employing an original testing methodology which emphasizes the importance of both success rate and efficiency. There was also of interest the study of the performance degradation with the increase of the search space dimension. A set of 11 scalable, multimodal, continuous optimization functions (10 unconstrained and 1 constrained) with known global solutions was constructed for testing purposes. The novel *F-ABC* variant was the only one capable to solve all the test functions for all the search space dimensions in the given testing conditions, but with the increase of the search space dimension it was observed that the methods *ABC* and *GABC* provided better success rates for the test functions with a regular structure of the modes (the modes disposed

in a lattice structure parallel with the coordinate system), while the novel *F-ABC* variant clearly provided better success rates for test functions presenting unstructured modes.

REFERENCES

[1] *Karaboga D., Basturk B.*, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization, **39**(2007), 459-471.

[2] *Zhu G., Kwong S.*, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation, **217**(2010), 3166-3173

[3] *Karaboga D., Gorkemli B., Ozturk C., Karaboga N.*, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review, **42**(2014), 21-57.

[4] *Khader A.T., Al-Betar M.A., Awadallah M.A.*, Artificial bee colony algorithm, its variants and applications: a survey, Journal of Theoretical & Applied Information Technology, **47**(2013), 434-459.

[5] *Pintér J. D.*, Global Optimization: Software, Test Problems, and Applications, Ch. 15 in Handbook of Global Optimization, Volume 2 (Ed. P. M. Pardalos and H. F. Romeijn), 515-569, Kluwer Academic Publishers, Dordrecht, Boston, London, 2002.

[6] *Deb K.*, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, **186**(2000), 311-338.

[7] *Mallipeddi R., Suganthan P. N.*, Differential Evolution with Ensemble of Constraint Handling Techniques for solving CEC 2010 Benchmark Problems, 2010 IEEE Congress on Evolutionary Computation (CEC), 18-23 July 2010.

[8] *Whitley D.*, A Genetic Algorithm Tutorial, Statistics and Computing, **4**(1994), 65-85.

[9] *Kennedy J., Eberhart R.C.*, Particle Swarm Optimization, in Proceedings of IEEE International Conference on Neural Networks, **4**(1995), 1942-1948, Perth, Australia.

[10] *Anescu G., Prisecaru I.*, NSC-PSO, a novel PSO variant without speeds and coefficients, Proceedings of 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2015, Timisoara, Romania, 460-467, 21-24 September, 2015.

[11] *Anescu G.*, An Imperialistic Strategy Approach to Continuous Global Optimization Problem, Proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing SYNASC 2014, Timisoara, Romania, 549-556, 22-25 September,2014.

[12] *Z. Michaelwicz*, Genetic Algorithms + Data structures = Evolution Programs, Springer, Berlin, 1994.

[13] *Momin J., Xin-She Yang*, A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, **4**(2013), 150-194.

[14] *Molga M., Smutnicki C.*, Test functions for optimization needs, http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf (last time accessed in February, 2016), (2005), 1-43.

[15] *Keane, AJ*, Experiences with optimizers in structural design, Proceedings of the 1st Conf. on Adaptive Computing in Engineering Design and Control, University of Plymouth, UK, (1994), 14-27.