Journal of Advances in Mathematics and Computer Science

25(5): 1-16, 2017; Article no.JAMCS.38065

Previously known as British Journal of Mathematics & Computer Science

ISSN: 2231-0851

1P-ABC, a Simplified ABC Variant for Continuous Optimization Problems

George Anescu[1*]

[1] *Power Plant Engineering Faculty, Polytechnic University of Bucharest, Romania.*

Author's contribution

The sole author designed, analyzed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/JAMCS/2017/38065

Editor(s):

(1) Junjie Chen, Professor, Department of Electrical Engineering, University of Texas at Arlington, USA.

(2) Huchang Liao, Professor, Business School, Sichuan University, P. R. China.

(3) Kai-Long Hsiao, Associate Professor, Taiwan Shoufu University, Taiwan.

Reviewers:

(1) S. K. Srivatsa, Retired, Anna University, India.

(2) P. Amudha, Avinashilingam Institute for Home Science and Higher Education for Women, India.

(3) Osman Ozkaraca, Mugla Stk Kocman University, Turkey.

(4) Anand Nayyar, KCL Institute of Management and Technology, India.

Complete Peer review History: http://www.sciencedomain.org/review-history/22263

Original Research Article

Received: 9th November 2017

Accepted: 11th December 2017

Published: 12th December 2017

Abstract

In this paper a novel simplified and fast variant of the ABC algorithm is proposed, 1 Population ABC (1P-ABC), with the aim to increase the efficiency of the ABC algorithm by using only one population of bees, the employed bees, while maintaining a good effectiveness of the algorithm in solving difficult nonlinear optimization problems. The novel 1P-ABC algorithm was tested, both regarding the efficiency and the success rate, against three known variants of ABC, the original ABC algorithm, an improved variant, Gbest-guided Artificial Bee Colony (GABC), and another improved variant, Fast ABC (F-ABC). 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) most of them with known global solutions. The novel proposed 1P-ABC algorithm outperformed the other ABC variants in efficiency, while for the success rate the results were mixed.

**Corresponding author: E-mail: george.anescu@gmail.com;*

*Keywords: Optimization; Continuous Global Optimization Problem (CGOP);
Swarm Intelligence (SI); Artificial Bee Colony Algorithm (ABC);
Gbest-guided Artificial Bee Colony Algorithm (GABC); Keane's Bump Function
Fast Artificial Bee Colony Algorithm (F-ABC); 1 Population ABC (1P-ABC).*

2010 Mathematics Subject Classification: 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 nature inspired algorithms 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 (populations) of agents (individuals), for example 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 Bee Colony (*ABC*), originally proposed in [1] and published in [2], 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 (it is derivative free), needs fewer control parameters (it is parameter free), has a simple design which is easy to implement and 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 of fields: decision making, engineering design, pattern recognition, image processing, machine learning, scheduling, protein structure prediction, etc.

Since its first introduction in 2005, many variants and hybridization variants of *ABC* were advanced in order to improve the efficiency and effectiveness of the algorithm. Two comprehensive surveys concerning the state of the art in the *ABC* algorithm research and its applications are presented in the 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 it 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 reported when optimization problems with a highly non-regular arrangement 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 general formulation of the Continuous Global Optimization Problem (*CGOP*); Section 3 presents the Deb's Rules for Constraints Handling as they were adapted and implemented for the *1P-ABC* method; Section 4 presents the original *ABC* algorithm and the modifications implemented in the *GABC* variant; Section 5 presents the modifications introduced in the implementation and design of the new *1P-ABC* variant; Section 6 presents the set of test optimization problems used in the testing experiments and the statistical results obtained by comparing the novel *1P-ABC* variant with the other three *ABC* variants: the original *ABC* algorithm [1], the *GABC* variant [5] and the *F-ABC* variant [6]; and finally, Section 7 summarizes the paper and draws some conclusions.

2 Continuous Global Optimization Problem

The Continuous Global Optimization Problem ($CGOP$) is a general model for representing the optimization problems formulated as [7], [8]:

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

with

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

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 suppositions are 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 [9] offer a methodology to efficiently handle the constraints in constrained optimization problems. The presentation in this section is adapted from [10] and [8] 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. According to his definition all equality constraints are active constraints. The equality constraints can be transformed into an inequality formulation 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 becomes 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 the summation of all the constraints expressed as:

$$v(\mathbf{x}) = \sum_{i=1}^{G+H} \tilde{g}_i(\mathbf{x})^2. \tag{4}$$

There are a number of constraint handling techniques based on constraint violation, the one used here being the Superiority of Feasible Solutions (SF) technique. SF applies the Deb's rules 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, but $\mathbf{x}_{i_1}$ 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 superior to 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 of the optimization problem.

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

$$v(\mathbf{x}) = \sum_{i=1}^{G+H} \tilde{g}_i(\mathbf{x})^2 + G_{ns} + H_{ns}, \tag{5}$$

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. In this way δ tends to final δ_2 with the increase of the iteration count k.

4 Artificial Bee Colony Optimization

The detailed description of the ABC algorithm given in this section is based on [6] and [11], but it is respecting the general principles proposed in [2] 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 step) 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 (a total population of $2\times N$ 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 the respective vector positions $\mathbf{x}_i^e$ and $\mathbf{x}_i^o$, $i = 1, \ldots, N$ in the the limiting box (hyper-rectangle) defined by the lower limits l_j, $j = 1, \ldots, n$ and upper limits u_j, $j = 1, \ldots, n$ ($l_j < u_j$).

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 $[0, 1)$ real interval. The iteration index is initialized to $k = 0$. The objective function $f(\mathbf{x})$ is evaluated for the current bees positions $\mathbf{x}_i^e(0), \mathbf{x}_i^o(0)$: $f_i^e(0) = f(\mathbf{x}_i^e(0)), f_i^o(0) = f(\mathbf{x}_i^o(0)), \;\; i = 1, \ldots, N$.

Each iteration of the optimization method consists of three phases: sending the employed bees to 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 to 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:

$$x_{i,j}^{\prime e}(k+1) = x_{i,j}^{e}(k) + r_{i,j}^{e} \times (x_{m,j}^{e}(k) - x_{i,j}^{e}(k)), \tag{9}$$

where $j \in \{1, \ldots, n\}$, is an uniform pseudo-randomly selected index and $r_{i,j}^{e}$ are pseudo-random numbers uniformly sampled from the $[-1, 1)$ real interval. Equation (9) controls the generation of a neighbor food source position around $\mathbf{x}_i^e(k)$ and the modification represents the visual comparison of the neighbor food positions visually by the bee. Note that only one component of $\mathbf{x}'^e_i$, namely the one with index j, is different from the corresponding component of $\mathbf{x}_i^e$. The new candidate food source position is taken as the current food source position only if it is better than the old one: i.e. if $f(\mathbf{x}'^e_i(k+1)) < f(\mathbf{x}_i^e(k))$ then $\mathbf{x}_i^e(k+1) := \mathbf{x}'^e_i(k+1)$. Equation (9) shows that as the difference between the position components $x_{i,j}^e(k)$ and $x_{m,j}^e(k)$ decreases, the perturbation on the position $\mathbf{x}_{i,j}^e(k)$ 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 [12]), but in the present implementation a ternary tournament selection (also widely applied in Genetic Algorithms, see [12])

was preferred, based on considerations 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:

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

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 $\mathbf{x}'^{o}_{i}$, namely the one with index j', is different from the corresponding component 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 proposed as $limit = n \times N$. 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:

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

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:

$$\begin{aligned} d_j(k) = & \max_{1 \leq i_1, i_2 \leq N, i_1 \neq i_2} \{|x^{o}_{i_1,j}(k) - x^{o}_{i_2,j}(k)|\}, \\ & j = 1, 2, \ldots, n. \end{aligned} \tag{12}$$

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

$$f^{o}_{max}(k) = \max_{1 \leq i_1 \leq N} \{f^{o}_{i_1}(k)\}, \tag{14}$$

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

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 into consideration 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* [13] the *Gbest-guided ABC* (*GABC*) method [5] improves the original *ABC* method by taking advantage of the global best (*gbest*) solution's information 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 [5] showed that the best results are obtained when taking $C = 1.5$.

The third *ABC* variant used in the testing section, Fast ABC (*F-ABC*) was published in [6], and its design improvements present similarities with the design improvements introduced in *1P-ABC* variant, although *F-ABC* is still a two populations *ABC* variant.

5 1 Population ABC Optimization

In order to improve the performance of the *ABC* optimization algorithm in both speed and success rate, a set of modifications were designed, implemented and tested, the new resulting optimization algorithm being named *1 Population ABC* (*1P-ABC*), based on the main and foremost modification, namely the use of only one population of bees, the employed bees. The scout bees are still used, but in reality they cannot be considered as a separate population from the employed bees, as an employed bee can only temporarily become a scout bee when its position is reset.

The main steps of the *1P-ABC* become:

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

The initialization step is similar to the one of the original *ABC* method, but it takes place for only one population (for simplification, the notation with e superscript was eliminated):

$$x_{i,j} = l_j + rnd_{i,j} \times (u_j - l_j), \;\; i = 1, \ldots, N, \;\; j = 1, \ldots, n. \tag{18}$$

The termination conditions are similar to the ones of the original ABC method presented in the previous section, but this time they are applied to the population of employed bees.

In the Employed Bees Phase, the current bee with index i is interacting with another bee with index m now determined by applying a binary tournament selection mechanism, *i.e.* first two different employed bees with indices m_1 and m_2 ($m_1 \neq m_2 \neq i$) are randomly selected, and then the best one of the two is selected as the candidate position: if $f(\mathbf{x}_{m_1}) < f(\mathbf{x}_{m_1})$ then $m := m_1$, else $m := m_2$. The new candidate position equation becomes:

$$x'_{i,j}(k+1) = x_{i,j}(k) + r_{i,j} \times (x_{m,j}(k) - x_{i,j}(k)), \tag{19}$$

with the numbers $r_{i,j}$ now uniformly pseudo-randomly sampled from the $[-0.5, 1.5)$ real interval. Also, in order to make the algorithm able to cope with highly non-linear objective functions, there is a need to give chances of change to all the dimensions. Therefore, a weight $w_i(k)$ is associated to each food source, and it is calculated using the formula:

$$w_i(k) = \frac{f(\mathbf{x}_i(k)) - f_{min}(k)}{f_{max}(k) - f_{min}(k)}, \;\; i = 1, \ldots, N, \tag{20}$$

where

$$f_{min}(k) = \min_{1 \leq i \leq N} f(\mathbf{x}_i(k)) = f(\mathbf{x}_{min}(k)), \;\; f_{max}(k) = \max_{1 \leq i \leq N} f(\mathbf{x}_i(k)). \tag{21}$$

For constrained optimization problems, in the context of applying the Deb's Rules for handling the constraints (as presented in Section 3), the weights cannot be defined as in (20), since the objective function's values are considered for comparisons only in the feasible regions. Therefore the weights for constrained optimization problems are defined as:

$$w_i(k) = \frac{i}{N}, \;\; i = 1, \ldots, N, \tag{22}$$

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. The candidate food source position is determined by applying equation (19), like in the original ABC algorithm, on one dimension uniformly pseudo-randomly sampled between 1 and n, but unlike in the original ABC algorithm, equation (19) is also applied on the remaining dimensions, but only after successfully passing a simple probabilistic test: $rnd() < w_i(k)$. 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). It was found experimentally that the weights defined by (22) provide a better convergence (exploitation) even for global optimization problems without constraints. Therefore, as a final design decision, a hybridization of the weights defined by (20) and (20) was adopted, with the definition (20) favored at the beginning of the iterations for a better exploration, and the definition (22) favored at the end of the iterations for a better convergence of the method. As a rule, for each bee the decision is taken at iteration $k+1$ based on a simple probabilistic test: if $rnd() < [(k+1)/iter_{max}]^{1/2}$ then definition (22) is adopted, else definition (20) is adopted.

For the *Scout Bees Phase*, it was found experimentally that the originally proposed formula for the stagnation count parameter *limit* gives a too high value. Therefore there were experimentally tried different values of *limit* 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 used test optimization problems with $limit = 4 \times n$ (see also [11], [6]), which was proposed as an appropriate formula. It is admitted that some further research is necessary in order to confirm the proposed

formula by broadening the set of test optimization problems.

The second modification in the *Scout Bees Phase* proposed the determination of the reset position of the scout bee in a uniformly pseudo-random manner inside a current hyper-sphere $H(\mathbf{c}(k+1), r(k+1))$, with the center in $\mathbf{c}(k+1)$ and the radius $r(k+1)$ given by:

$$r(k+1) = \max_{1 \le i \le N} \{\|\mathbf{c}(k+1) - \mathbf{x}_i(k+1)\|_2\}, \tag{23}$$

where $\mathbf{c}(k+1)$ is the weighted mass center of the employed bees population:

$$\mathbf{c}(k+1) = \frac{\sum_{i=1}^{N} w_i'(k+1)\mathbf{x}_i(k+1)}{\sum_{i=1}^{N} w_i'(k+1)}, \tag{24}$$

with the weights $w_i'(k+1)$ given by:

$$w_i'(k+1) = 1 - w_i(k+1), \;\; i = 1, \ldots, N, \tag{25}$$

and the weights $w_i(k+1)$ defined in (20).

Note: the design of the *1P-ABC* algorithm allows approaching constrained optimization problems without any changes, provided that a constrained optimization methodology based on Deb's rules is employed.

6 Testing and Results

The purpose of the testing phase was to prove that the new proposed *1P-ABC* algorithm is competitive when compared to other known *ABC* variants. For comparison three *ABC* variants were chosen: the original *ABC* algorithm, the improved *GABC* variant [5], and another improved *F-ABC* variant ([6]).

In order to conduct the tests, an appropriate testing methodology was devised (see also [11], [14], [6]). 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 it is 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 the author's opinion such methodologies are not recognizing the importance of success rate and are concealing it from reporting. A very efficient method (with a fast convergence), but having a low success rate, cannot be considered better than a less efficient method, but having 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.

In the testing phase of a meta-heuristic optimization method, it is important to select an appropriate testbed of diverse test functions for comparison purposes. Depending on the quality of the selected testbed, the results can somewhat be extrapolated to the general set of optimization problems, although due to the difficulty of the general global optimization problem (an NP-complete problem), there is still no guarantee that the proposed method can successfully approach any mathematical model. A testbed of 11 known scalable multimodal optimization functions (10 unconstrained and 1 constrained, see [15], [16], [17], [18], [19]) was for the tests run on the four compared optimization methods. The analytical expressions of the test functions in the used testbed are given below respecting the *CGOP* model.

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

$$\begin{gathered} f_1(\mathbf{x}) = 10n + \sum_{j=1}^{n}[x_j^2 - 10\cos(2\pi x_j)], \\ -5.12 \leq x_j \leq 5.12, \;\; j = 1, \ldots, n. \end{gathered} \tag{26}$$

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

$$\begin{gathered} f_2(\mathbf{x}) = \sum_{j=1}^{n}(|x_j \sin(x_j)| + 0.1|x_j|), \\ -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n. \end{gathered} \tag{27}$$

- *Alpine 2 Function* - highly multimodal, global maximum value of 2.808^n at $(7.917, 7.917, \ldots, 7.917)$: (30476.9172 for $n = 10$, 928842479.5682 for $n = 20$ and 28308255304346.7530 for $n = 30$):

$$\begin{gathered} f_3(\mathbf{x}) = \prod_{j=1}^{n}\sqrt{x_j}\sin(x_j), \\ 0 \leq x_j \leq 10, \;\; j = 1, \ldots, n. \end{gathered} \tag{28}$$

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

$$\begin{gathered} f_4(\mathbf{x}) = \frac{1}{4000}\sum_{j=1}^{n}x_j^2 - \prod_{j=1}^{n}\cos\left(\frac{x_j}{\sqrt{j}}\right) + 1, \\ -100 \leq x_j \leq 100, \;\; j = 1, \ldots, n. \end{gathered} \tag{29}$$

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

$$\begin{gathered} f_5(\mathbf{x}) = -\frac{1}{n}\sum_{j=1}^{n}x_j \sin\left(\sqrt{|x_j|}\right), \\ -500 \leq x_j \leq 500, \;\; j = 1, \ldots, n. \end{gathered} \tag{30}$$

- *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 -997867.45525 at $(9.9993, 9.9993, \ldots, 9.9993)$ for $n = 30$:

$$f_6(\mathbf{x}) = \sum_{j=1}^{n-1} \left[\log(x_j - 2)^2 + \log(10 - x_j)^2\right] - \left(\prod_{j=1}^{n-1} x_j\right)^{0.2}, \quad 2.0001 \leq x_j \leq 9.9999, \;\; j = 1, \ldots, n. \tag{31}$$

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

$$f_7(\mathbf{x}) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_n, x_1), \quad -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n, \tag{32}$$

where

$$g(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{1 + 0.001(x^2 + y^2)^2}. \tag{33}$$

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

$$f_8(\mathbf{x}) = -\frac{1}{n} \sum_{j=1}^{n} \sin(x_j) \sin^{2m}\left(\frac{j x_j^2}{\pi}\right), \quad m = 10, \;\; 0 \leq x_j \leq \pi, \;\; j = 1, \ldots, n. \tag{34}$$

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

$$f_9(\mathbf{x}) = 20 + e - 20e^{-0.2\left(\frac{1}{n}\sum_{j=1}^{n} x_j^2\right)^{1/2}} - e^{\frac{1}{n}\sum_{j=1}^{n} cos(2\pi x_j)}, \quad -30 \leq x_j \leq 30, \;\; j = 1, \ldots, n. \tag{35}$$

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

$$f_{10}(\mathbf{x}) = n - 1 + \sum_{j=1}^{n-1} \cos\left(\frac{|x_{j+1} - x_j|}{|x_j + x_{j+1}| + 10^{-10}}\right), \quad -10 \leq x_j \leq 10, \;\; j = 1, \ldots, n. \tag{36}$$

- *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 -0.821884162 for $n = 30$):

$$
\begin{aligned}
f_{11}(\mathbf{x}) &= -\left|\left\{\sum_{j=1}^{n}\cos(x_j)^4 - 2\prod_{j=1}^{n}\cos(x_j)^2\right\} \Big/ \left(\sum_{j=1}^{n} j x_j^2\right)^{0.5}\right|, \\
g_1(\mathbf{x}) &= 0.75 - \prod_{j=1}^{n} x_j \leq 0.0, \\
g_2(\mathbf{x}) &= \sum_{j=1}^{n} x_j - 7.5n \leq 0.0, \\
0.0 &\leq x_j \leq 10.0, \quad j = 1, \ldots, n.
\end{aligned}
\tag{37}
$$

Note that the global minimum -0.821884162 for $n = 30$ was found during testing and it is an improvement over -0.818056222 reported in [19].

All the test functions with global optima in origin were shifted, considering that the origin is favored by *ABC* type methods and this peculiarity could interfere in the results. In order to preserve the total number of particles when different *ABC* variants were compared the number of employed bees in the *1P-ABC* variant was the double $(2 \times N)$ of the number of employed bees in the other *ABC* variants (N). 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 $n = 10, 20$ and $N = 400$ for $n = 30$). *N/A* in the tables, depending on the context, means *Not Available* when a method already gave good results for a more restrictive tolerance and in this case it was not further tested for a less restrictive tolerance, or *Not Applicable* when the success rate of a method is 0% for the tested tolerance and therefore $\mu(NFE)$ could not be reported.

Table 1 presents the comparative testing results obtained for $n = 10$. From the success rate perspective, it can be observed that *1P-ABC* and *F-ABC* obtained the maximum percentage for all the test functions, while *ABC* was not able to solve f_{11} and showed incipient problems with f_3, f_4 and f_{10}, and *GABC* showed incipient 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, *1P-ABC* surpassed all the other *ABC* variants for all the test functions with the exception of f_1, for which *GABC* was faster.

Table 2 presents the comparative testing results obtained for $n = 20$. *1P-ABC* and *F-ABC* were the only methods able to solve all the test problems for the given testing conditions. From the success rate perspective *1P-ABC* showed incipient difficulties in solving f_3, f_8, f_{10} and f_{11}. *F-ABC* showed incipient difficulties in solving f_1, f_3, f_8, f_{10} and f_{11}. *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 with *ABC* for exactly the same functions. From the efficiency perspective, *1P-ABC* again surpassed the other *ABC* variants for all the test functions, with the exceptions of f_1 and f_8, for which *GABC* was faster.

Table 3 presents the comparative testing results obtained for $n = 30$. Again *1P-ABC* and *F-ABC* were the only methods capable to solve all the test problems for the given testing conditions. From the success rate perspective *1P-ABC* showed deterioration for f_3, f_8, f_{10} and f_{11}. *F-ABC* showed deterioration for f_1, f_3, f_7, f_8, f_{10} and f_{11}. *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 with *ABC* for exactly the same functions. From the efficiency perspective, *1P-ABC* again surpassed the other *ABC* variants for all the test functions with the exceptions of f_1 and f_8, for which *GABC* was faster.

Note that *1P-ABC* and *F-ABC* were the *ABC* variants able to solve all the test problems for all the tested search space dimensions.

Table 1. 1P-ABC versus *F-ABC*, *ABC* and *GABC*, $n = 10$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ 1P-ABC	$\mu(NFE)$ 1P-ABC	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	100%	52160	100%	61940	100%	93774	100%	**50350**
f_2	100%	**22616**	100%	24704	100%	60558	100%	33708
f_3	100%	**29166**	100%	40476	84%	187084	99%	208131
f_4	100%	**31701**	100%	38662	98%	90477	100%	64486
f_5	100%	**20796**	100%	26541	100%	63672	100%	26188
f_6	100%	**10560**	100%	11660	100%	34176	100%	14600
f_7	100%	**82601**	100%	110778	100%	258828	100%	103454
f_8	100%	**46621**	100%	54672	100%	98842	100%	61244
f_9	100%	**24708**	100%	27976	100%	96998	100%	48602
f_{10}	100%	**152040**	100%	220334	99%	645614	100%	566790
f_{11}^{**}	100%	**43554**	100%	53621	0%	N/A	89%	232420

Table 2. 1P-ABC versus *F-ABC*, *ABC* and *GABC*, $n = 20$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ 1P-ABC	$\mu(NFE)$ 1P-ABC	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	100%	136206	99%	164930	100%	250486	100%	**129496**
f_2	100%	**36208**	100%	40668	100%	144244	100%	83410
f_3	**99%**	**73547**	84%	105366	1%	125600	2%	918816
f_4	100%	**37620**	100%	44094	100%	130602	100%	80360
f_5	100%	**43232**	100%	58314	100%	228518	100%	71392
f_6	100%	**19988**	100%	22754	100%	97352	100%	43862
f_7	100%	**250411**	100%	507039	18%	1232733	26%	872115
f_8	89%	344840	66%	360543	100%	446884	100%	**280196**
f_9	100%	**40288**	100%	45974	100%	215866	100%	111490
f_{10}	**71%**	**631978**	66%	897662	0%	N/A	0%	N/A
f_{10}^{*}	N/A	N/A	N/A	N/A	33%	1532945	78%	1232751
f_{11}^{**}	70%	**66109**	**94%**	254994	0%	N/A	0%	N/A

Table 3. 1P-ABC versus *F-ABC*, *ABC* and *GABC*, $n = 30$, $runs = 100$, $tolerance = 0.1\%$, $N = 100$

Fn	$SR\%$ 1P-ABC	$\mu(NFE)$ 1P-ABC	$SR\%$ F-ABC	$\mu(NFE)$ F-ABC	$SR\%$ ABC	$\mu(NFE)$ ABC	$SR\%$ GABC	$\mu(NFE)$ GABC
f_1	100%	236614	91%	289129	100%	436558	100%	**218932**
f_2	100%	**48200**	100%	54936	100%	235984	100%	137964
f_3	**90%**	**137662**	29%	180861	0%	N/A	0%	N/A
f_4	100%	**48978**	100%	55412	100%	195762	100%	119122
f_5	100%	**66907**	100%	91539	100%	473928	100%	124644
f_6	100%	**33230**	100%	37790	100%	120422	100%	93148
f_7	**100%**	**429726**	94%	1153975	0%	N/A	0%	N/A
f_8	66%	982031	12%	1131014	100%	920902	100%	**595538**
f_9	100%	**53744**	100%	61606	100%	339068	100%	177614
f_{10}	18%	**1151994**	**20%**	1866919	0%	N/A	0%	N/A
f_{10}^{*}	N/A	N/A	N/A	N/A	0%	N/A	6%	2579333
f_{11}^{**}	27%	**83072**	**42%**	403339	0%	N/A	0%	N/A

7 Conclusions

The paper proposed a novel global optimization method, *1P-ABC*, as a variant of the known *ABC* method, designed to eliminate some of the drawbacks experienced by *ABC* mainly related to a poor exploitation capability (which makes the algorithm relatively slow) and poor success rates when approaching optimization problems with a highly non-regular arrangement of the modes. The proposed algorithm was described in details so that it can be easily implemented in any programming language suitable for scientific computing applications. The novel *1P-ABC* algorithm was tested against three variants of *ABC*, the original one and other two variants, *GABC* and *F-ABC*. The testing was conducted by employing an original testing methodology which emphasizes the importance of both success rate and efficiency. There was also considered 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) most of them with known global solutions was constructed for testing purposes. The novel *1P-ABC* variant, together with *F-ABC* variant, were the only ones capable to solve all the test functions for all the tested 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 arrangement of the modes (the modes disposed in a lattice structure parallel with the coordinate system), while the methods *1P-ABC* and *F-ABC* clearly provided better success rates for the test functions presenting non-regularly arranged modes. From the efficiency perspective, *1P-ABC* surpassed the other *ABC* variants for almost all the test functions and search space dimensions. The final conclusion can be considered as a confirmation of the No Free Lunch theorem for optimization functions (see [20]), stating that no algorithm can outperform any other algorithm when performance is amortized over all optimization functions. The *1P-ABC* optimization method needs to be further more extensively tested on larger sets of optimization problems and using various testing methodologies in order to confirm its effectiveness, and many further improvements are needed in order to make it a competitive method. Further research directions should consider hybridizing *1P-ABC* with other *ABC* variants in order to obtain an optimization method that provides consistency in results over more extended classes of test optimization functions. Also *1P-ABC*, or improved variants of it, will be used in approaching multimodal optimization problems that arise in Machine Learning applications.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department; 2005.

[2] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization. 2007;33:459-471.

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

[4] Bolaji AL, Khader AT, Al-Betar MA, Awadallah MA. Artificial bee colony algorithm, its variants and applications: A survey. Journal of Theoretical & Applied Information Technology. 2013;47:434-459.

[5] Zhu G, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation. 2010;217:3166-3173.

[6] Anescu G. A fast artificial bee colony algorithm variant for continuous global optimization problems. U.P.B. Sci. Bull., Series C. 2017;79(1):83-98.

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

[8] Anescu G. Gradual and Cumulative Improvements to the Classical Differential Evolution Scheme through Experiments, Annals of West University of Timisoara - Mathematics and Computer Science. 2016;54(2):13-35.

[9] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering. 2000;186:311-338.

[10] Mallipeddi R, Suganthan PN. Differential Evolution with Ensemble of Constraint Handling Techniques for solving CEC 2010 Benchmark Problems. In: 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain: July (2010) 18-23.

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

[12] Whitley D. A Genetic Algorithm Tutorial. Statistics and Computing. 1994;4:65-85.

[13] Kennedy J, Eberhart RC. Particle Swarm Optimization In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, (1995) 1942-1948.

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

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

[16] Momin J, Yang XS. A literature survey of benchmark functions for global optimization problems. Int. Journal of Mathematical Modelling and Numerical Optimisation. 2013;4:150-194.

[17] Molga M, Smutnicki C. Test functions for optimization needs; 2005.

Available: http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf
(Accessed Date:03/11/17).

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

[19] Mishra SK. Minimization of Keane's Bump Function by the Repulsive Particle Swarm and the Differential Evolution Methods; 2007.
Available: http://mpra.ub.uni-muenchen.de/3098/
(Accessed Date:03/11/17).

[20] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation. 1997;1:67-82.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/22263