- ORIGINAL ARTICLE -

Artificial Bee Colony Algorithm Improved with Evolutionary Operators

Algoritmo de colonia de Abejas Artificiales Hibridado con Algoritmos Evolutivos

Gabriela Minetti[1] and Carolina Salto[1,2]

[1] *Facultad de Ingeniería, Universidad Nacional de La Pampa, Argentina*
{minettig, saltoc}@ing.unlpam.edu.ar
[2] *CONICET, Argentina*

Abstract

In this paper, we design, implement, and analysis the replacement of the method to create new solutions in artificial bee colony algorithm by recombination operators, since the original method is similar to the recombination process used in evolutionary algorithms. For that purpose, we present a systematic investigation of the effect of using six different recombination operators for real-coded representations at the employed bee step. All the analysis is carried out using well known test problems. The experimental results suggest that the method to generate a new candidate food position plays an important role in the performance of the algorithm. Computational results and comparisons show that three of the six proposed algorithms are very competitive with the traditional bee colony algorithm.

Keywords: ABC algorithm, parameter tuning, recombination

Resumen

En este trabajo, se ha diseñado, implementado y analizado el reemplazo del método para crear nuevas soluciones en algoritmos basados en colonia de abejas artificiales por operadores de recombinación, ya que el método original es similar al proceso de recombinación usado en los algoritmos evolutivos. Para cumplir con este propósito, se presenta una investigación sistemática del efecto de usar seis operadores de recombinación distintos en el procedimiento llevado a cabo por la abeja empleada. Para la experimentación se utilizan casos de pruebas complejos, habitualmente utilizados en la literatura. Los resultados obtenidos sugieren que el método generador de nuevas fuentes de comida afecta el desempeño del algoritmo. A partir del análisis y comparaciones de los resultados, se observa que tres de las seis propuestas algorítmicas son competitivas con respecto al algoritmo basado en colonia de abejas tradicional.

Palabras claves: Algoritmo ABC, configuración paramétrica, recombinación

1 Introduction

Swarm intelligence is a research field inspired by the cooperation of large numbers of homogeneous agents in the environment. An ant colony, a flock of birds, a honeybee or an immune system are typical examples of swarm systems. Many algorithms that simulate these models have been proposed in order to solve a wide range of problems. In this line, Karaboga et al. [1] proposed an optimization algorithm based on the intelligent behavior of honey bee swarm, called Artificial Bee Colony (ABC) algorithm. Considering that it works with a set of solutions, ABC is classified as a population-based metaheuristic.

Like most metaheuristics, the parameter tuning is a drawback in the ABC algorithms, which is not easy to perform in an thorough manner [1]. Morover, the parameters may have a great influence on the efficiency and effectiveness of the search. They are not only numerical values but may also involve the use of search components. This problem was addressed by several researchers, mainly by studying more suitable value ranges of parameters to the solution of the noise elimination problem [2]. Akay and Karaboga [3] modified a larger number of variables at each employed and onlooker bee step than in the original ABC version. In [4, 5, 6], the authors presented different modifications to the method to generate a new food position by adding information of the best global solution. Diwold et al. [5] also changed the way to calculate the selection probability of a solution by introducing the Euclidean distance between two solutions in the probability equation. Alatas [7] used chaotic maps to create the initial solutions and a local search to generate

Citation: G. Minetti and C. Salto. "*Artificial Bee Colony Algorithm Improved w ith Evolutionary O perators*". Journal of Computer Science & T echnology, vol.18, no.2, pp. 114-124, 2018.
DOI: 10.24215/16666038.18.e13
Received: February 21, 2018. **Revised:** June 11, 2018. **Accepted:** June 19, 2018.

the solution at the scout bee step. Furthermore, ABC was hybridized with Differential Evolution (DE) in order to propose different ways to create the initial solutions and local search methods to generate solutions at scout bee step [8, 9]. Jadon et al. [10] also used the hybridization with DE to improve the employed, onlooker, and scout bee phases. In [11], Yan et al. presented an ABC hybridized with Genetic Algorithms to exploit the search space as an extra step in the ABC algorithm. Another hybridization was presented by Li et al. in [12], where the bacterial foraging optimization algorithm is used as a local search step.

The aforementioned proposals considered different solutions, added factors into the original method to create a candidate food position, or introduced components of another metaheuristics, but no new entirely distinct ways to generate new food positions were presented. The mechanism used by ABC to produce a new candidate solution is very similar to the procedure carried out by the recombination operators for real-coded spaces in the evolutionary algorithm literature. The effect of this change on the ABC performance had not been studied and their impact could be more significant than the traditional approach, becoming this in the objective of this work. In this sense, the application of other mechanisms to generate new source positions using the recombination operators is a very promising approach in order to improve the ABC performance.

For this purpose, we design and implement the replacement of the original method with genetic recombination operators. Six recombination operators for real-coded spaces are considered in this step, they are: Arithmetical, Binomial, Linear, One Point, Multi-Point, and Max-Min Arithmetical recombinations. As a consequence,six new ABC variants arise. The methodology devised in this work is targeted to characterize these variants regarding the performance and it can be summarized as follows. We evaluate these ABC variants using a large scale global optimization test suite (IEEE CEC'2008 [13]) and compare them against the traditional ABC approach. On the one side, this study includes both the solution quality and computational effort in order to analyze the effectiveness and efficiency of the all approaches. Furthermore, we explore the fitness evolution through the search to give a better insight into the internal behavior of each algorithm. On the other side, a comparison with the state-of-the-art solutions for the considered benchmarks is carried out. Thus, the performance of the best ABC variants are compared and ranked with the first six algorithms of CEC'2008 competition. A preliminary version of this paper appears in [14].

The rest of this article is organized as follows. In Section 2 and 3, we describe the ABC algorithm and the used recombination operators, respectively. Section 4 explains the experimental analysis and the methodology used. Then, we study and analyze the results obtained by the different ABC variants in Section 5. Moreover, a comparison between our results with state-of-the-art algorithms is shown in the Section 6. Finally, we present our principal conclusions and future lines of research.

Algorithm 1 ABC Algorithm

1: Initialize the population of solutions $x_i, i = 1,2,...,SN$;
2: Evaluate the population;
3: **repeat**
4: The employers generate the new solutions, v_i, from each x_i, using Eq. 1;
5: The employers evaluate the new solutions, v_i;
6: The employers apply the greedy selection mechanism;
7: The onlookers generate the new solutions, v_i, from the selected x_i, using Eq. 1;
8: The onlookers evaluate the new solutions, v_i;
9: The onlookers apply the greedy selection mechanism;
10: The scouts determine the abandoned solutions and replace them with new solutions, x_i
11: Memorize the best solution found so far;
12: **until** *the stop criterion is meet*
13: **return** The best solution;

2 Artificial Bee Colony Algorithm

The artificial bee colony algorithm is a simple optimization tool, motivated by the intelligent behavior of honey bees. In this kind of algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount in this source corresponds to the quality (fitness) of the associated solution. ABC provides a population-based search procedure in which solutions are modified by three kinds of artificial bees: employed, onlooker, and scout bees.

The employed bees are associated with a particular food source. The onlooker bees wait in the hive for waggle dances exerted by the other bees to establish food sources. These kinds of bees find and exploit new food sources, memorize their locations, load a portion of nectar to the beehive, and unload it to the food area in the hive. Finally, the scout bees search new food sources in the environment surrounding the hive in a random way.

At the first step, an initial population of SN solutions is randomly generated (see lines 1 and 2 in the Algorithm 1). Each solution x_i $(i = 1,2,...,SN)$ is a D-dimensional vector. Here, D is the dimension of the function or problem to be optimized. Secondly, this population is iteratively modified by the employed, onlooker and scouts bees (see lines 3-12).

A candidate food position, v_i, is generated by employed bees from the old v_i in memory (lines 4-5), modifying only the parameter j as is shown in the Equation 1:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{1}$$

where $k \in \{1,2,...,SN\} \wedge k \neq i$ and $j \in \{1,2,...,D\}$ are randomly chosen indexes. A random number between [-1, 1], called ϕ_{ij}, is used to control the generation of food sources around x_i and compare the two

food positions viewed by a bee. As the difference between the parameters x_{ij} and x_{kj} decreases, the perturbation on the position x_{ij} gets decreased, too. Thus, as the search approaches the optimum solution in the search space, the step length is adaptively reduced. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value, e.g. to its limit value .

Regarding the probability value associated with a food source, an artificial onlooker bee chooses a food source, x_k. Then this bee applies the Equation 1 to obtain the candidate food position, v_i, (lines 7-8). As a consequence, the proportional selection method is used to chose a solution, which inherently applies a high pressure on the selection. In order to avoid it, the proportional selection method is replaced by the binary tournament selection [15] in our experiments.

After the creation and evaluation of v_i, its fitness is compared with x_i. If v_i is equal or better than x_i, v_i takes the place in the memory. In this way, a greedy selection mechanism is applied to select between v_i and x_i (lines 6 and 9).

A food source is assumed to be abandoned when a position cannot be improved further through a predetermined number of cycles, known as "*limit*" for abandonment. When a food source is abandoned by the employed bees, it is replaced with a new food source found by the scouts (line 10). This is simulated by generating a random new position to replace the abandoned one.

3 Our ABC Proposal

The employed bees use a variation operator, as the one used (Equation 1) in the fourth line of the Algorithm 1, to produce a candidate food position from the old one in memory. This operator generates one new candidate solution by combining the information contained in two existing ones. The resulting process is similar to the recombination operation from the evolutionary algorithms, which combines the genetic information from two parents. Therefore, this section presents the recombination operators used as ABC variation operator in our proposal. The most of them comes from the real-coded Genetic Algorithms literature and has never been used with this metaheuristic. The proposed ABC variants use the recombination operator to create a single trial vector. In what follows we name the ABC algorithm using the traditional variation operator as ABC_{TRAD}.

Let us assume that x_i and x_k ($i,k \in \{1,2,\ldots,SN\}$) are the two solutions considered to produce a candidate food position v_i. For a randomly chosen parameter j ($j \in (1,2,\ldots,D)$) in the solution, the operators sketch in the following description can be incorporated to the ABC instead of using Equation 1 in line 4. The resting parameters l ($l \in (1,2,\ldots,D) \wedge l \neq j$) of v_i come from x_i.

3.1 Variating only one parameter in the candidate food position

The following operators only change one parameter $j \in \{1,2,\ldots,D\}$ in v_i and the other ones are copied from x_i. Furthermore, these three operators (Linear, Arithmetical, and Max-Min-Arithmetical recombinations) are specific for the real-coded genetic algorithms.

Arithmetical Recombination (ABC_{AX}). This operator [16] chooses the parameter values of the candidate food position (v_{ij}) somewhere around and between the parameter values of x_{ij} and x_{kj}. Let $\lambda \in (0,1)$ be an uniform random value, which are chosen for each new candidate solution. Then, the j-th parameter value of the candidate solution v_i is computed according to Equation 2.

$$v_{ij} = (1-\lambda)x_{ij} + \lambda x_{kj} \tag{2}$$

Max-Min-Arithmetical Recombination (ABC_{MMAX}). In this case, the operator [16] generates four candidate solutions according to Equation 3. After evaluating them, a greedy selection mechanism is considered. Thereupon, four additional evaluations per new candidate solution are required. For the first two candidates, a $\lambda \in (0,1)$ value is used that is an uniform random value.

$$\begin{aligned} v_{ij1} &= \lambda x_{ij} + (1-\lambda)x_{kj} \\ v_{ij2} &= (1-\lambda)x_{ij} + \lambda x_{kj} \\ v_{ij3} &= min(x_{ij}, x_{kj}) \\ v_{ij4} &= max(x_{ij}, x_{kj}) \end{aligned} \tag{3}$$

Linear Recombination (ABC_{LX}). This operator [17] is similar to the Arithmetical recombination, but the λ remains fix and can take two possible values 0.5 and 1.5. Three candidate solutions v_i are generated according to Equation 4. After evaluating each new candidate position, the best one is considered. Consequently, this method requires three additional evaluations per new candidate solution.

$$\begin{aligned} v_{ij1} &= 0.5(x_{ij} + x_{kj}) \\ v_{ij2} &= 1.5x_{ij} - 0.5x_{kj} \\ v_{ij3} &= -0.5x_{ij} + 1.5x_{kj} \end{aligned} \tag{4}$$

3.2 Variating many parameters in the candidate food position

The next three operators change more than one parameter j in the solution v_i, keeping the value of each one without any adjustment, i.e., it is only copied from x_i or x_k, depending of the operator, but no combinations of values for this parameter are made. One of this operator (Binomial Recombination) is specifically designed for the real-coded genetic algorithms, while the other two comes from the binary codification.

Binomial Recombination (ABC_{BX}). The parameter values of the candidate food position are chosen from x_i or x_k (see Equation 5), depending on a random value u to be lower than the probability parameter $\rho \in (0,1)$, which is a defined by the user [18]. For this work, the ρ value was set to 0.5. Moreover, BX generates the candidate food position ensuring that it gets at least one variable from the k-th food position, as indicated in Equation 6.

$$v_{ij} = \begin{cases} x_{kj} & \text{if } u \leq \rho \\ x_{ij} & \text{otherwhise} \end{cases} \quad (5)$$

$$v_{ij} = \begin{cases} x_{kj} & \text{if } v_{ij} = x_{ij} \\ v_{ij} & \text{otherwhise} \end{cases} \quad (6)$$

One-Point Recombination (ABC_{1PX}). This operator randomly selects a cut point $p \in (1,D)$ and the tails of x_i and x_k are swapped to get the candidate food position, as is seen in Equation 7.

$$v_i = \{x_{i1},\ldots,x_{ip},x_{k(p+1)},\ldots,x_{kD}\} \quad (7)$$

Multi-Point Recombination (ABC_{mPX}). In this operator [19], m different cut points ($m_p \in (1,D-1)$) are chosen at random with no duplicates and sorted into ascending order. Then, the variables between successive cut points are exchanged between x_i and x_k to produce a new candidate food position, as is shown in Equation 8.

$$v_i = \{x_{i1},\ldots,x_{im_1},x_{k(m_1+1)},\ldots,x_{k(m_2)},x_{i(m_2+1)},\ldots,x_{iD}\} \quad (8)$$

4 Experimental Design

In this section we describe the experimental design used in this work to compare ABC_{TRAD} with the six different algorithmic variants: ABC_{AX}, ABC_{BX}, ABC_{LX}, ABC_{1PX}, ABC_{mPX}, and ABC_{MMAX}. Furthermore, the execution environment and the analysis methodology are detailed in this section.

A popular choice for evaluating the performance of algorithms in the literature is to use the IEEE CEC'2008 test suite [13]. This benchmark is specially designed with large scale real-parameter minimization problems (i.e. of dimensions D=100, 500, and 1000). The mean and standard deviation of the error value are used to measure the performance of the algorithmic variants. The error is calculated as the difference between the current value of the global optimum and the best found value. Particularly, for function F7, the absolute value of the obtained optimum is recorded and compared, because for that function, the globally optimal function value is unknown.

In the experiments, the seven ABC variants use the same parameter settings. The colony size SN was set to 50. Furthermore, the control parameter $limit$ is defined by $limit = SN \times D$ [20]. Finally, the stop criterion is to achieve the maximum number of function evaluations, computed as suggested in [13] ($5000 \times D$). Notice that we are not using highly specialized ABC algorithms, since our goal is not to outperform other algorithms, for the considered test suite, but to analyze the influence of the different mechanisms to generate candidate food sources in the behavior of the proposed algorithms.

Because of the stochastic nature of the algorithms, we performed 30 independent runs of each test to gather meaningful experimental data and apply statistical confidence metrics to validate our results and conclusions. Before performing the statistical tests, we first checked whether the data followed a normal distribution by applying the Shapiro-Wilks test. Since the results do not follow a normal distribution, the non-parametric Kruskal-Wallis (KW) test is applied. Then, the pair-wise Wilcoxon test is used, in order to assess individual differences in the performance of the algorithms. This pair-wise test must be adjusted to compensate the family-wise-error derived from the performance of multiple comparisons, using the Holm's method. Multiple comparisons using the Tukey's range test are used. All tests are carried out considering as significance value $\alpha = 0.01$.

5 Analysis of Results

In the following, we present an analysis of the results obtained by the six ABC variants described in Section 3 and ABC_{TRAD} to solve the CEC'2008 functions. Our main goals are to offer different and efficient mechanisms that will be used by the employed bees to generate candidate food positions. For that purposes, we analyze the quality of results considering the error values obtained by ABC for the functions from $f1$ to $f6$ and the objective values for $f7$. Moreover, the computational effort to find the best solution is evaluated.

5.1 Comparison of the All ABC Variants

In the Table 1, the mean error values found by the ABC variants are shown. The best values are bold faced. From this table, two well differentiable groups of algorithms can be observed when the quality of the solutions is considered, regardless of the dimension. The first group, which consists of ABC_{TRAD}, ABC_{AX}, ABC_{LX}, and ABC_{MMAX}, obtains the lowest error values, and some of them find the optimal values for the functions $f1$, $f5$, and $f6$ in all runs. In the second group, which is composed of the remaining algorithms, their best solutions are far from the optimum value (error values bigger than 1.28E+02). These differences among the algorithms is statistically supported by post-hoc tests after KW

Table 1: Mean error values from *f1* to *f6*, and mean objective values for *f7*.

Func.	Dim.	ABC$_{TRAD}$	ABC$_{AX}$	ABC$_{BX}$	ABC$_{LX}$	ABC$_{1PX}$	ABC$_{mPX}$	ABC$_{MMAX}$	KW
f1	100	**0.00E+00**	**0.00E+00**	4.88E+05	2.09E-01	1.42E+05	5.30E+04	**0.00E+00**	+
	500	**0.00E+00**	**0.00E+00**	3.03E+06	3.98E+00	1.80E+06	1.23E+06	**0.00E+00**	+
	1000	**0.00E+00**	**0.00E+00**	6.20E+06	3.81E+00	4.28E+06	3.29E+06	1.03E-03	+
f2	100	**6.35E+01**	7.65E+01	1.58E+02	8.71E+01	1.38E+02	1.28E+02	8.05E+01	+
	500	1.39E+02	**1.01E+02**	1.83E+02	1.14E+02	1.72E+02	1.64E+02	1.10E+02	+
	1000	1.60E+02	**1.10E+02**	1.87E+02	1.25E+02	1.80E+02	1.76E+02	1.24E+02	+
f3	100	**7.87E+00**	1.23E+01	4.39E+11	2.31E+01	5.92E+10	1.47E+10	1.70E+02	+
	500	**1.30E+01**	3.03E+01	3.44E+12	6.84E+01	1.50E+12	8.13E+11	8.35E+02	+
	1000	**1.49E+01**	5.51E+01	7.36E+12	1.81E+02	4.08E+12	2.68E+12	1.70E+03	+
f4	100	1.30E+00	**4.46E-02**	2.25E+03	3.35E+00	9.66E+02	5.42E+02	2.53E-01	+
	500	1.81E+01	5.17E+00	1.24E+04	2.60E+01	8.50E+03	6.78E+03	**1.30E+00**	+
	1000	4.10E+01	1.30E+01	2.52E+04	6.02E+01	1.97E+04	1.66E+04	**2.54E+00**	+
f5	100	**0.00E+00**	**0.00E+00**	4.15E+03	2.28E-02	1.11E+03	4.48E+02	**0.00E+00**	+
	500	**0.00E+00**	**0.00E+00**	2.58E+04	2.35E-01	1.50E+04	1.02E+04	**0.00E+00**	+
	1000	**0.00E+00**	**0.00E+00**	5.51E+04	2.04E-01	3.84E+04	2.97E+04	**0.00E+00**	+
f6	100	**0.00E+00**	**0.00E+00**	2.13E+01	3.79E-02	1.97E+01	1.80E+01	3.59E-02	+
	500	**0.00E+00**	**0.00E+00**	2.15E+01	4.63E-01	2.10E+01	2.06E+01	1.29E-01	+
	1000	**0.00E+00**	**0.00E+00**	2.15E+01	5.32E-01	2.12E+01	2.10E+01	1.56E-01	+
f7	100	-9.09E+02	-9.14E+02	-6.98E+02	**-9.73E+02**	-7.25E+02	-7.40E+02	-9.48E+02	+
	500	-5.50E+03	-5.52E+03	-3.12E+03	**-5.76E+03**	-3.33E+03	-3.33E+03	-5.61E+03	+
	1000	-1.16E+04	-1.15E+04	-6.03E+03	**-1.19E+04**	-6.37E+03	-6.34E+03	-1.16E+04	+

Table 2: Minimum error values from *f1* to *f6*, and minimum objective values for *f7*.

Func.	Dim.	ABC$_{TRAD}$	ABC$_{AX}$	ABC$_{LX}$	ABC$_{MMAX}$
f1	100	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	500	**0.00E+00**	**0.00E+00**	1.06E-01	**0.00E+00**
	1000	**0.00E+00**	**0.00E+00**	2.94E-01	**0.00E+00**
f2	100	**5.74E+01**	6.84E+01	7.67E+01	7.24E+01
	500	1.33E+02	**9.62E+01**	1.05E+02	1.04E+02
	1000	1.55E+02	**1.02E+02**	1.17E+02	1.19E+02
f3	100	9.23E-01	2.49E+00	**1.63E+00**	1.34E+02
	500	3.80E+00	2.02E+01	**1.20E+00**	7.55E+02
	1000	**8.64E+00**	4.62E+01	5.34E+01	1.57E+03
f4	100	**0.00E+00**	**0.00E+00**	1.90E+00	**0.00E+00**
	500	1.30E+01	2.00E+00	2.11E+01	**0.00E+00**
	1000	3.32E+01	1.02E+01	5.10E+01	**7.00E-03**
f5	100	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	500	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	1000	**0.00E+00**	**0.00E+00**	2.00E-03	**0.00E+00**
f6	100	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	500	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	1000	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.00E-03
f7	100	-9.74E+02	-1.00E+03	**-1.06E+03**	-1.04E+03
	500	-5.71E+03	-5.70E+03	**-5.88E+03**	-5.83E+03
	1000	-1.19E+04	-1.22E+04	-1.20E+04	**-1.23E+04**

test (p-values $<$ 4.9E-10), as shown in the last column with the symbol +. The poor performance of algorithms in the second group is due to the method to generate the new candidate solution, which does not introduce any adjustment to the parameters at the employed bee step through the search process. Moreover, multi-point and binomial operators reduce the representational bias at the expense of increasing the disruption of parameters.

The ranking of the variants across the all dimensions is shown in the Figure 1. To obtain these ranks, the mean errors of all variants on a same function and dimension were ranked from best (rank 1) to worst (rank 7). In the case of ties, average ranks are computed. Additionally, the #$func$ row on top of each bar indicates the number of functions where each ABC variant finds the optimum value. This figure empirically confirms the differences between the two above mentioned groups, since ABC$_{AX}$, ABC$_{TRAD}$, ABC$_{MMAX}$, and ABC$_{LX}$ (in this order) are the best ranked algorithms. Furthermore, the three first algorithms solve optimally between three and four functions, while the all algorithms of the second group are not able to find the optimal value for any function and dimension (#$func = 0$).

5.2 Comparison of the Best ABC Variants

Regarding the previous analysis, we continue the result study considering the first group of algorithms. The Table 2 shows the minimum values obtained by each algorithmic variant considering all functions and dimensions. Additionally, the Figure 2 presents boxplots that show the distribution of the evaluations to find the best solution observed for each studied ABC variant on the 7 benchmark functions. On the basis of the results shown in the Figure 2, an important difference between ABC$_{LX}$ and the rest of the algorithms is observed. ABC$_{LX}$ needs a small computational effort, i.e. the lowest number of evaluations to find its best solution. But when the minimum error values from Table 2 are considered, ABC$_{LX}$ presents a poor performance to solve the CEC'2008 benchmark functions in comparison with the result quality obtained by ABC$_{AX}$, ABC$_{TRAD}$, and ABC$_{MMAX}$. Following with the analysis of the number of evaluations to find the best solutions, the results of the post-hoc tests remark statistically significant differences between ABC$_{TRAD}$ and both ABC$_{LX}$ and ABC$_{MMAX}$, and between ABC$_{AX}$ and ABC$_{LX}$ for 100D and 500D. In the case of 1000D, differences between ABC$_{LX}$ and ABC$_{MMAX}$ is also exists.

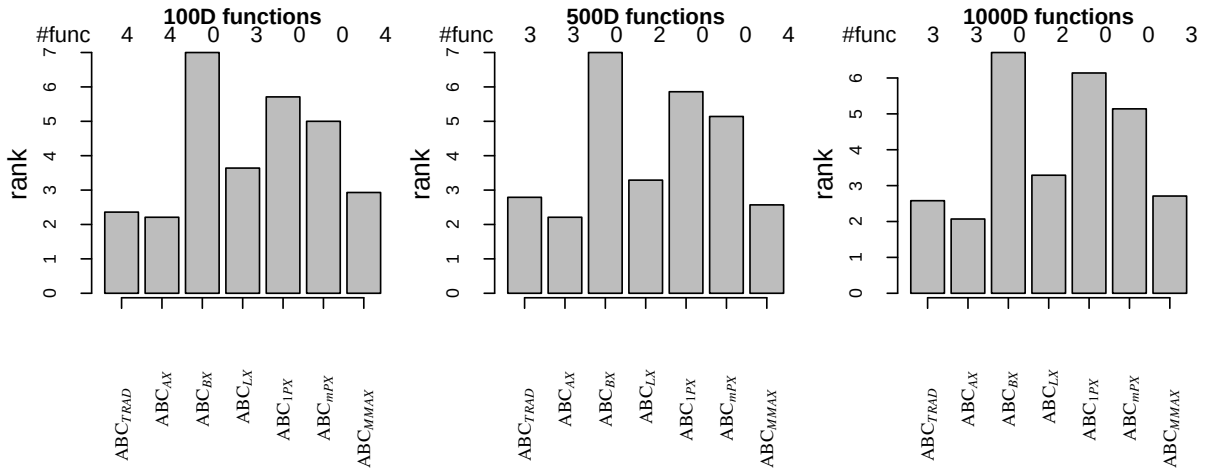


Figure 1: Performance comparison based on the average rank over 7 functions. The ranking was computed using the average error value of each algorithm.

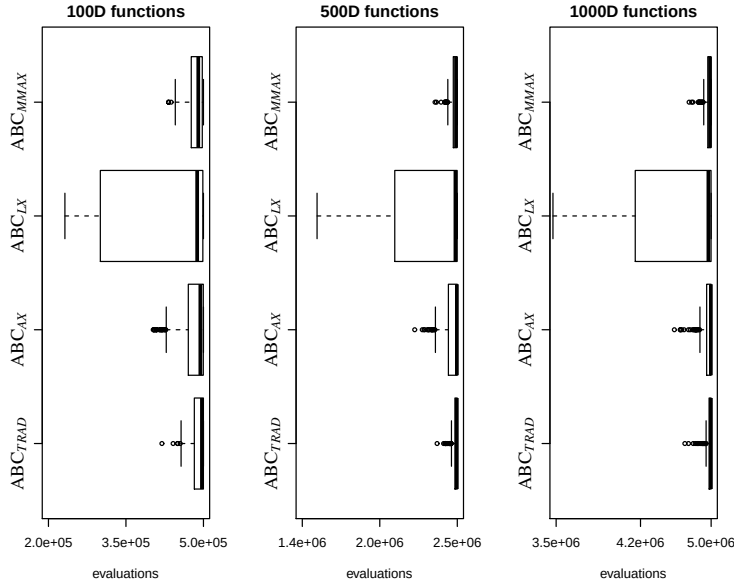


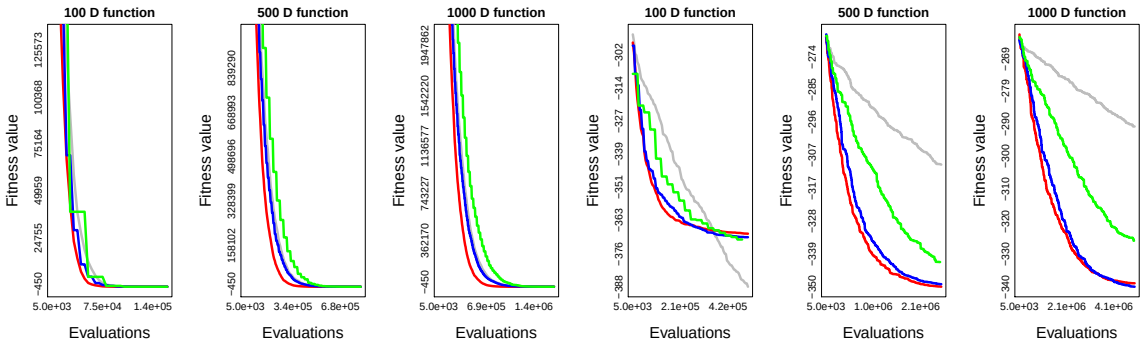
Figure 2: Boxplot of number of evaluations to find the best solutions for each dimension.

5.3 Analysis of Internal Behavior of the Best Four ABC Variants

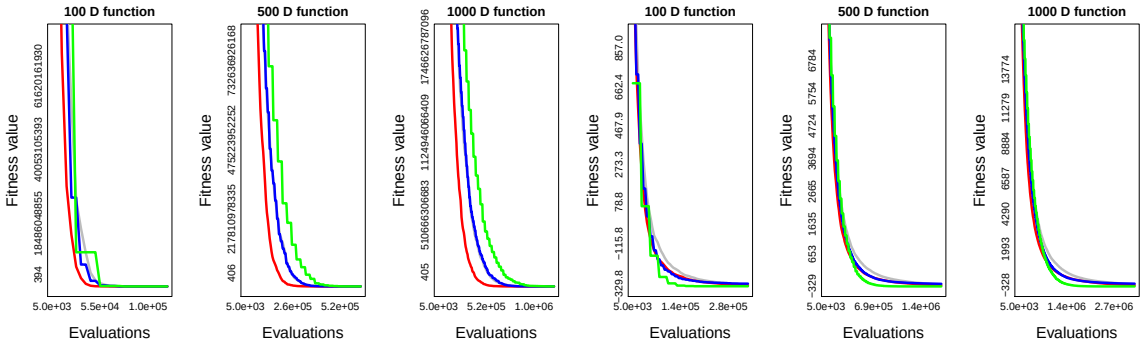
Let us now proceed with the internal behavior analysis of the best performing ABC algorithms. The Figure 3 shows the evolution of the best fitness along the execution of each algorithm for the problem instances under all dimensions. On the one hand, we observe that regardless of the problem dimension, ABC_{AX} drives to good solutions in fewer evaluations than the rest. This observation suggests that this algorithm leads to work out high quality solutions. Furthermore and as observed in previous analysis (see Section 5.2), ABC_{MMAX} is the ABC variant that needs more evaluations to find their best solutions, particularly for $f2$. On the other hand, the ABC_{TRAD} and ABC_{LX} behaviors oscillate from the ABC_{AX} to the ABC_{MMAX} ones. Furthermore, after an initial period of evolution, the curves belonging to the all ABC variants overlapped to the ABC_{AX}, which is an expected situation since the algorithms obtain best solutions of similar quality.

5.4 Discussion

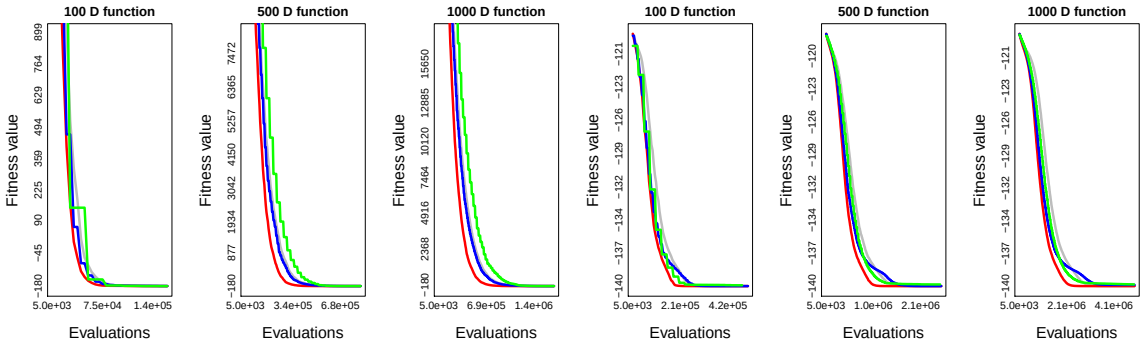
Summarizing, the ABC_{AX} algorithm obtains the best tradeoff between the solution quality and the effort to find the best result. As a consequence, these results suggest that practitioners developing ABC-based solutions for applied optimization could adopt more



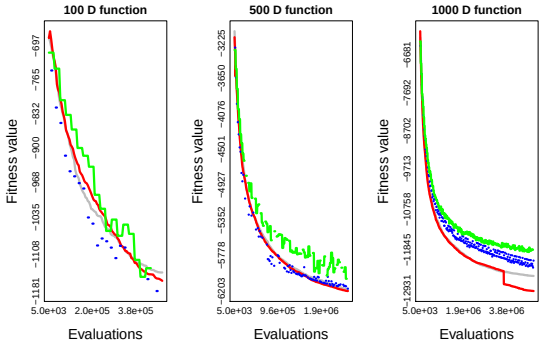
(a) *f1* function (b) *f2* function



(c) *f3* function (d) *f4* function



(e) *f5* function (f) *f6* function



(g) *f7* function

Figure 3: Evolution of the best fitness during the search for the all functions and dimensions.

efficient choices to generate new food positions, at the employed bee step. These choices can be based on standard recombination operators, such us the AX operator.

6 Comparison with the State-of-the-Art Algorithms

In this section, we compare the performance of ABC_{AX}, the best algorithm proposed in this work, with other metaheuristics that solve the same benchmark, in order to verify their performance. It is noticeable that state of the art only reports ABC results for functions until 200 dimensions [21, 22]. As a consequence, ABC_{AX} is compared with the first six algorithms of CEC'2008 competition. These algorithms are described in Section 6.1 and the comparison is carried out in Section 6.2.

6.1 The First Best CEC'2008 Algorithms

These algorithms are: the Multiple Trajectory Search (*MTS*) [23], a self-adaptive differential evolution algorithm ($jDEdynNP-F$) [24], a MultiLevel Cooperative Coevolution (*MLCC*) framework [25], a Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO) [26], a Differential Evolution with Self-adaptation and Cooperative Co-evolution (DEwSAcc) Algorithm [27], and an Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO) [28].

The MTS [23] uses multiple agents to search the solution space concurrently. Each agent chooses a local search method that best fits the landscape of a solution's neighborhood, an agent may find its way to a local optimum or the global optimum.

In $jDEdynNP-F$ [24], a self-adaptive control mechanism is used to change the control (F and CR) parameters [29, 30] during the optimization process. Additionally, the algorithm incorporates a method for gradually reducing the population size [31], and a mechanism for changing the sign of the F parameter.

For the *MLCC* [25], first a set of problem decomposers is constructed based on the random grouping strategy with different group sizes. Then, the evolution process is divided into a number of cycles, and at the start of each cycle MLCC uses a self-adapted mechanism to select a decomposer according to its historical performance. Since different group sizes capture different interaction levels between the original objective variables, MLCC is able to self adapt among different levels.

DMS-PSO uses a dynamic and randomized neighborhood topology, which is improved by combining a classic local search algorithm, Quasi-Newton method [26]. In this way, the whole population is divided into a large number sub-swarms, these subswarms are regrouped frequently by using various regrouping schedules and information is exchanged among the particles in the whole swarm.

In DEwSAcc [27], the original differential evolution is extended by log-normal self-adaptation of its control parameters and combined with cooperative coevolution as a dimension decomposition mechanism.

In order to improve the PSO's searching ability and efficiency, EPUS-PSO [28] is hybridized with the population manager. This manager eliminates redundant particles and hires new ones or maintains the particle number according to the solution searching status to make the process more efficient.

6.2 Results Comparison

As we can observe in the preceding descriptions, the CEC'2008 winner metaheuristics need complex improvements for solving these large scale real-parameter minimization functions. Instead, ABC_{AX} only changes the original method to create a candidate food position by a very simple recombination operator. Also, it is remarkable that our proposal neither incorparates local search methods nor adaptive control parameters. In this way, our approach requires less numerical effort than the remaining ones.

Furthermore, the simplicity of our proposal allows to provide state-of-the-art solutions for the considered benchmark. Thus, ABC_{AX} is ranked second among the CEC'2008 winner metaheuristics. Besides, ABC_{AX} optimally solves the same number of functions than the best performing CEC'2008 metahuristics (*MTS*), as shown in Figure 4. To obtain these ranks, the mean errors of all variants on a same function and dimension were ranked from best (rank 1) to worst (rank 7). In the case of ties, average ranks are computed. Additionally, the $\#func$ row on top of each bar indicates the number of functions where each ABC variant finds the optimum value. To obtain these ranks, we used the same method described in the previous section.

7 Conclusions

This article analysis the effect of changing the methods to generate a candidate food position in the artificial bee colony algorithm. In order to do this, six different recombination operators are considered to create new food positions at the employed bee step. These operators are taken from the real-coded genetic algorithm literature and they are used for the first time in the ABC algorithm. In order to verify the performance of our best performing ABC variant (ABC_{AX}), a comparison from a point of view of the solution quality is carried out considering this algorithm with the state-of-the-art CEC'2008 metaheuristics.

The experimental results of the Arithmetical operator application show a very good performance in terms of the average quality rank and the computa-

Table 3: Mean error values from *f1* to *f6* and minimum objective values for *f7* obtained by ABC_{AX} and the first six algorithms in the CEC'2008 competition.

Func.	Dim.	ABC_{AX}	*MTS* [23]	*jDEdynNP-F* [24]	*MLCC* [25]	*DMS-PSO* [26]	*DEwSAcc* [27]	*EPUS-PSO* [28]
1	100	**0.00E+00**	**0.00E+00**	5.68E-14	6.82E-14	**0.00E+00**	5.68E-14	7.47E-01
	500	**0.00E+00**	**0.00E+00**	9.32E-14	4.30E-13	**0.00E+00**	2.10E-09	8.45E+01
	1000	**0.00E+00**	**0.00E+00**	1.14E-13	8.46E-13	**0.00E+00**	8.79E-03	5.53E+02
2	100	7.65E+01	**1.44E-11**	4.29E+01	2.53E+01	3.65E+00	8.25E+00	1.86E+01
	500	1.01E+02	**7.32E-06**	8.46E+00	6.67E+01	6.89E+01	7.57E+01	4.35E+01
	1000	1.10E+02	**4.72E-02**	1.95E+01	1.09E+02	9.15E+01	9.61E+01	4.66E+01
3	100	1.23E+01	**5.17E-08**	1.12E+02	1.50E+02	2.83E+02	1.45E+02	4.99E+03
	500	3.03E+01	**5.04E-03**	6.61E+02	9.25E+02	4.67E+07	1.81E+03	5.77E+04
	1000	5.51E+01	**3.41E-04**	1.31E+03	1.80E+03	8.98E+09	9.15E+03	8.37E+05
4	100	4.46E-02	**0.00E+00**	5.46E-14	4.39E-13	1.83E+02	4.38E+00	4.71E+02
	500	5.17E+00	**0.00E+00**	1.47E-12	1.79E-11	1.61E+03	3.64E+02	3.49E+03
	1000	1.30E+01	**0.00E+00**	2.17E-04	1.37E-10	3.84E+03	1.82E+03	7.58E+03
5	100	**0.00E+00**	**0.00E+00**	2.84E-14	3.41E-14	**0.00E+00**	3.07E-14	3.72E-01
	500	**0.00E+00**	**0.00E+00**	4.21E-14	2.13E-13	**0.00E+00**	6.90E-04	1.64E+00
	1000	**0.00E+00**	**0.00E+00**	3.98E-14	4.18E-13	**0.00E+00**	3.58E-03	5.58E+00
6	100	**0.00E+00**	**0.00E+00**	5.68E-14	1.11E-13	**0.00E+00**	1.13E-13	2.06E+00
	500	**0.00E+00**	6.18E-12	1.49E-13	5.34E-13	2.00E+00	4.80E-01	6.64E+00
	1000	**0.00E+00**	1.24E-11	1.47E-11	1.06E-12	7.76E+00	2.30E+00	1.89E+01
7	100	-9.14E+02	-1.49E+03	-1.48E+03	**-1.54E+03**	-1.14E+03	-1.37E+03	-8.55E+02
	500	-5.52E+03	-7.08E+03	-6.88E+03	**-7.44E+03**	-4.20E+03	-5.75E+03	-3.51E+03
	1000	-1.15E+04	-1.40E+04	-1.35E+04	**-1.47E+04**	-7.51E+03	-1.06E+04	-6.62E+00

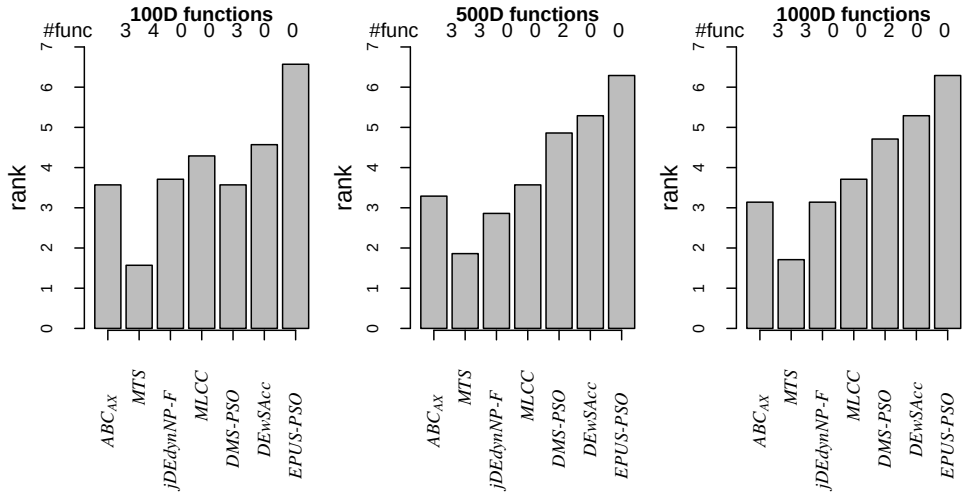


Figure 4: Performance comparison based on the average rank over 7 functions. The ranking was computed using the average error value of each algorithm.

tional effort for all considered functions and dimensions. As a consequence, this operator arises as a promising alternative to the traditional method for creating new food positions. Furthermore, the performances of the ABC variants applying LX and MMAX (the other operators that adjust one variable in the solutions) are similar to the behavior of the traditional ABC. Instead the ABC variants applying BX, 1PX, *m*PX operators, which only copy parts of other solutions to create a new one, seem no good alternatives to be used as optimization tool, because of the low quality of their solutions for any function and dimension.

Moreover, the comparison from a point of view of performance ranks the ABC_{AX} in the second place among the CEC'2008 winner metaheuristics. It is remarkable the algorithmic simplicity of our proposal in front of the complex improvements introduced by the compared algorithms. Furthermore, this ABC variant solves competitively high dimensionality benchmarks (100, 500, and 1000) whereas the ABC algorithms in the literature provide solutions for benchmarks with less than 200 dimensions. Consequently, ABC_{AX} becomes in a relevant metaheuristic in the state-of-the-art of ABC algorithms.

As future research line, we will experiment with combinatorial problems to extend the study presented in this work. These kind of problems has different solution representations, imposing new methods to generate food positions. Another future line is related

with the parallelization of the ABC algorithm using the current parallel programming models.

Acknowledgments

The authors acknowledge the support of Universidad Nacional de La Pampa and the Incentive Program from MINCyT. The second author is also funded by CONICET.

Competing Interests

Funding: This study was funded by the Universidad Nacional de La Pampa and CONICET.

Conflict of Interest: Author Gabriela Minetti declares that she has no conflict of interest. Author Carolina Salto declares that she has no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108 – 132, 2009.

[2] S. Kockanat and N. Karaboga, "Parameter tuning of artificial bee colony algorithm for gaussian noise elimination on digital images," in *2013 IEEE INISTA*, pp. 1–4, 2013.

[3] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Science*, vol. 192, pp. 120–142, 2012.

[4] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166 – 3173, 2010.

[5] K. Diwold, A. Aderhold, A. Scheidler, and M. Middendorf, "Performance evaluation of artificial bee colony optimization and new selection schemes," *Memetic Computing*, vol. 3, no. 3, p. 149, 2011.

[6] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," *Applied Soft Computing*, vol. 11, no. 2, pp. 2888 – 2901, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[7] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682 – 5687, 2010.

[8] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 11, pp. 2741 – 2753, 2012.

[9] W. feng Gao and S. yang Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687 – 697, 2012.

[10] S. S. Jadon, R. Tiwari, H. Sharma, and J. C. Bansal, "Hybrid Artificial Bee Colony algorithm with Differential Evolution," *Applied Soft Computing*, vol. 58, pp. 11 – 24, 2017.

[11] X. Yan, Y. Zhu, and W. Zou, "A hybrid artificial bee colony algorithm for numerical function optimization," in *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 127–132, Dec 2011.

[12] L. Li, F. Zhang, C. Liu, and B. Niu, "A hybrid Artificial Bee Colony algorithm with bacterial foraging optimization," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 127–132, 2015.

[13] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, , and Z. Yang, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

[14] G. F. Minetti and C. Salto, "Improving artificial bee colony algorithm with evolutionary operators," in *XXIII Congreso Argentino de Ciencias de la Computación (CACIC 2017).*, pp. 93–102, 2017.

[15] M.-D. Zhang, Z.-H. Zhan, J.-J. Li, and J. Zhang, *Tournament Selection Based Artificial Bee Colony Algorithm with Elitist Strategy*, pp. 387–396. Cham: Springer International Publishing, 2014.

[16] F. Herrera, M. Lozano, and A. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," *International Journal of Intelligent Systems*, vol. 18, no. 3, pp. 309—-338, 2003.

[17] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*, pp. 205–218, Morgan Kaufmann, 1991.

[18] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2–9, Morgan Kaufmann Publishers Inc., 1989.

[19] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Biases in the crossover landscape," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 10–19, Morgan Kaufmann Publishers Inc., 1989.

[20] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm," *J. of Global Optimization*, vol. 39, pp. 459–471, Nov. 2007.

[21] X. Y., F. P., and Y. L., "A simple and efficient artificial bee colony algorithm," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[22] R. A. and P. M., "An improved self-adaptive artificial bee colony algorithm for global optimisation," *International Journal of Swarm Intelligence (IJSI)*, vol. 1, no. 2, 2014.

[23] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3052–3059, June 2008.

[24] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 2032–2039, June 2008.

[25] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1663–1670, June 2008.

[26] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3845–3852, June 2008.

[27] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3718–3725, June 2008.

[28] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai, "Solving large scale global optimization using improved particle swarm optimizer," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1777–1784, June 2008.

[29] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 215–222, 2006.

[30] B. Boskovic, S. Greiner, J. Brest, and V. Zumer, "A differential evolution for the tuning of a chess evaluation function," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1851–1856, 2006.

[31] J. Brest and M. Sepesy Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, pp. 228–247, Dec 2008.