Improving Artificial Bee Colony Algorithm with Evolutionary Operators

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

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

Abstract. In this paper, we analyze the effect of replacing 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 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.

Keywords: ABC algorithm - recombination - parameter tuning

1 Introduction

Swarm intelligence is the study of computational systems 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. Tereshko and Loengarov [1] consider a bee colony as a dynamical system, where gathering information from an environment and adjusting its behavior in accordance to it produce an intelligent decision-making from enhancing the level of communication among the individuals. Taking the Tereshko and Loengarov's ideas into account, Karaboga et al. [2, 3] propose 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.

In general, metaheuristics have a major drawback: they need some parameter tuning that is not easy to perform in a thorough manner. Those parameters are not only numerical values but may also involve the use of search components. Those parameters may have a great influence on the efficiency and effectiveness of the search. The ABC metaheuristic is not the exception [3].

Some works have deal with this problem, mainly by investigating value ranges of parameters more suitable to the solution of problems with certain features [4, 5]. Regarding the search components, several ways to create the initial solutions were presented in [6, 7] and local search methods to generate solutions at scout bee step have been researched in [6–8]. In [9], the authors changed the way to

calculate the selection probability of a solution by introducing the Euclidean distance between two solutions in the probability equation. Furthermore, different approaches introducing partial changes in the method to generate new food positions at each employed and onlooker bee step have been proposed [7, 9–11].

The cited proposals consider different solutions or add factors into the original method to create a candidate food position, but no new entirely distinct ways to generate it are 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. In this sense, we consider the application of other mechanisms to generate new source positions using the recombination operators, whose effect on the ABC performance has not yet been studied and their impact could be more significant than the traditional approach. Consequently, the objective of this article is to analyze the effect of using six different recombination operators to create new food positions at the employed bee step of the ABC algorithm.With respect to the methodology followed to analyze the results, we study the performance of these ABC variants with respect to the traditional ABC version considering both the solution quality and computational effort. Therefore, the effectiveness and efficiency of three of the six proposed operators in comparison with the original ABC version is shown.

The rest of this article is organized as follows. In Section 2 and 3, we describe ABC and the suggested 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. Finally, we present our principal conclusions and future lines of research.

2 Artificial Bee Colony Algorithm

The artificial bee colony is one of the several algorithms have been developed depending on different intelligent behaviors of honey bee swarms. In the ABC 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. Three kinds of artificial bees can act on the food sources: *(i)* scout bees search new food sources in the environment surrounding the nest in a random way; *(ii)* onlooker bees wait in the nest waggle dances exerted by the other bees to establish food sources; and *(iii)* employed bees are associated with a particular food source. The last kind of bees finds and exploits new food sources, memorizes their locations, loads a portion of nectar to the beehive, and unloads it to the food area in the hive.

In the Algorithm 1 a pseudo-code of the ABC [2, 3] is shown. At the first step, an initial population of SN solutions is randomly generated. 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.

Algorithm 1 ABC Algorithm

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

The employed bees generate a candidate food position, v_i, from the old one in memory, 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. The value ϕ_{ij} is a random number between [-1, 1], which 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.

An artificial onlooker bee chooses a food source, x_k, depending on the probability value associated with that food source and, then it applies the Equation 1 to obtain the candidate food position. In other words, a solution is chosen using the proportional selection method. But with this method a high pressure is applied to the selection. In order to avoid it in our experiments, the proportional selection method is replaced by the binary tournament selection [12].

When a candidate source position, v_i, is created and evaluated by an artificial bee, its fitness is compared with the old x_i one. If the new solution is equal or better than the old one, the new one takes the place in the memory. In other words a greedy selection mechanism is applied to select between v_i and x_i.

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. This is simulated by generating a random new position to replace the abandoned one.

3 Recombination Operators

To produce a candidate food position from the old one in memory, the employed bees uses a variation operator as the one presented in Equation 1. This technique, which generates one new candidate solution by combining the information

contained in two existing ones, is similar to the recombination operators from the evolutionary algorithms. In consequence, this section presents the recombination operators included in the experimental section. The most of them comes from the real-coded genetic algorithms literature and has never been used with this metaheuristic. The proposed ABC algorithms 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. The resting parameters l ($l \in (1, 2, \ldots, D) \wedge l \neq j$) of v_i come from x_i.

Arithmetical Recombination (ABC_{AX}). This operator [13] 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 [13] 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 [14] is similar to the AX, 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.

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

Previous operators only change one parameter $j \in \{1, 2, \ldots, D\}$ in v_i and the other ones are copied from x_i but, in what follows, more than one parameter j in the solution v_i is different than x_i ones. Another change is that the value of each parameter remains 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. The operators are the following.

Binomial Recombination ($\mathbf{ABC}_{BX}$). 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 [15]. 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.

$$v_{ij} = \begin{cases} x_{kj} & \text{if } u \leq \rho \\ x_{ij} & \text{otherwise} \end{cases} \quad (5) \qquad v_{ij} = \begin{cases} x_{kj} & \text{if } v_{ij} = x_{ij} \\ v_{ij} & \text{otherwise} \end{cases} \quad (6)$$

One-Point Recombination ($\mathbf{ABC}_{1PX}$). 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 ($\mathbf{ABC}_{mPX}$). In this operator [16], 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 [17]. 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. The control parameter *limit* is defined by $limit = SN \times D$ [2]. The stop criterion is to achieve the maximum number of function evaluations, computed as suggested in [17] ($5000 \times D$). 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 in order to compensate for the family-wise-error derived from the performance of multiple comparisons, using the Holm's method. Furthermore, multiple comparisons using the Tukeys 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 $f7$. Additionally, we study the hit rate and the distribution of the error values presented for all algorithms. Moreover the computational effort to find the best solution is evaluated.

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 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 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

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	+

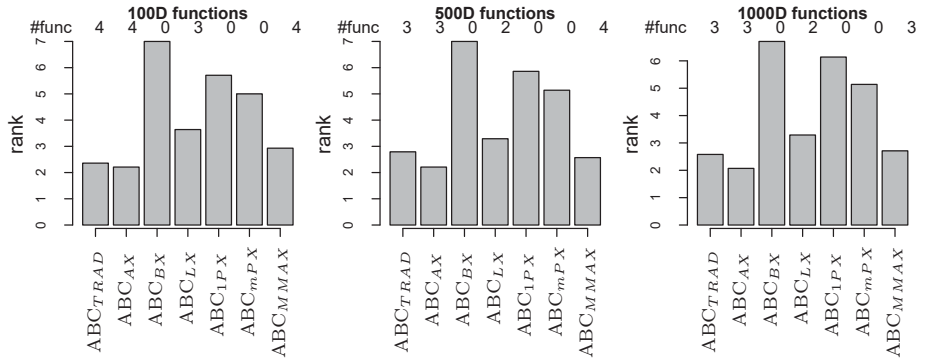

Fig. 1. Performance comparison based on the average rank over 7 functions. The ranking was computed using the average error value of each algorithm.

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 ($\#func = 0$).

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,

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**

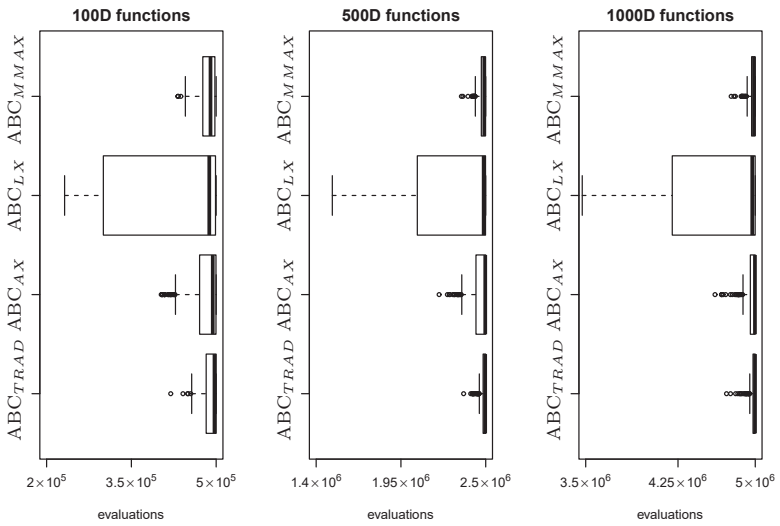

Fig. 2. Boxplot of number of evaluations to find the best solutions for each dimension.

the Figure 2 presents box-plots 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 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, the difference between ABC_{LX} and ABC_{MMAX} is also presented.

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 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 Conclusions

This article analysis the effect of considering other methods to generate a candidate food position in the artificial bee colony algorithm. For that purpose, an experimental evaluation of six recombination operators used to generate new solutions is carried out to solve the CEC' 2008 benchmark functions. These operators are taken from the real-coded genetic algorithm literature and they are used for the first time in the ABC framework.

The results obtained in these experiments suggest the use of the Arithmetical recombination as an interesting alternative for the traditional method to create new food positions. Since this operator presents a very good performance in terms of the average quality rank and the computational effort for all considered functions and dimensions. Furthermore, the performance of the use of the remaining operators that adjust variables in the solutions (Linear and and Max-Min-Arithmetical recombination operators) is similar to the behavior of the traditional algorithm. Instead the ABC algorithms, which apply Binomial, One-Point, and Multi-Point recombination operators and only copy parts of other solutions to create a new one, seems no good alternatives to be used in the ABC framework because of the low quality of their solutions for any function and dimension.

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.

References

1. V. Tereshko and A. Loengarov, "Collective decision-making in honey bee foraging dynamics," *Computing and Information Systems Journal, ISSN 1352-9404*, 2005.
2. 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, no. 3, pp. 459–471, Nov. 2007.
3. 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.
4. S. Kockanat and N. Karaboga, "Parameter tuning of artificial bee colony algorithm for gaussian noise elimination on digital images," in *2013 IEEE INISTA*, 2013, pp. 1–4.
5. B. Akay and D. Karaboga, "Parameter tuning for the artificial bee colony algorithm," in *Proceedings of the 1st International Conference on Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, ser. ICCCI '09. Springer-Verlag, 2009, pp. 608–619.
6. B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682 – 5687, 2010.
7. 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.
8. D. Aydın, T. Liao, M. A. Montes de Oca, and T. Stützle, *Improving Performance via Population Growth and Local Search: The Case of the Artificial Bee Colony Algorithm*. Springer Berlin Heidelberg, 2012, pp. 85–96.
9. 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.
10. B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Science*, vol. 192, pp. 120–142, 2012.
11. 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.
12. M.-D. Zhang, Z.-H. Zhan, J.-J. Li, and J. Zhang, *Tournament Selection Based Artificial Bee Colony Algorithm with Elitist Strategy*. Cham: Springer International Publishing, 2014, pp. 387–396.
13. 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.
14. A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 205–218.
15. G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1989, pp. 2–9.
16. 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*. Morgan Kaufmann Publishers Inc., 1989, pp. 10–19.
17. 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," Nature Inspired Computation and Applications Laboratory, USTC, China, Technical Report, 2007.