On the computational performance of artificial bee colony programming strategy

Yassine BOUDOUAOUI
Applied Automation Laboratory, F.H.C., University of Boumerdès
Av. de l'indépendance, 35000 Boumerdès, Algeria
tahayac@yahoo.fr,

Hacene HABBI
Applied Automation Laboratory, F.H.C., University of Boumerdès
Av. de l'indépendance, 35000 Boumerdès, Algeria
habbi_hacene@hotmail.com.

Abstract—In recently published works, a novel artificial bee colony programming algorithm (ABCP) is proposed to solve symbolic regression problem which is a very important practical issue. Symbolic regression is a process of obtaining a mathematical model using given finite sampling of values of independent variables and associated values of dependent variables. This paper addresses the analysis of the computational performance of ABCP strategy with respect to two key control parameters. Considering typical benchmark problems, general rules are deduced through various simulations performed under different settings.

Keywords— Artificial bee colony programming; symbolic regression; swarm optimization; automatic programming.

I. Introduction

Symbolic regression aims to find a mathematical model expressed in a symbolic form that best fits a set of data samples. Traditional linear and nonlinear regression methods fit parameters to an equation of a given form. On the other hand, symbolic regression method constructs mathematical equations by composing both parameters and equational forms. Equivalently, it searches nonlinear equations through the manipulation of equational forms and parameters simultaneously while solving a given modeling problem. Symbolic regression method attempts to find the best combination of variables (inputs and outputs), symbols, and coefficients to develop an optimal model satisfying a set of fitness cases.

To solve the problem of symbolic regression, evolutionary programming techniques are extensively used. Evolving models and evolutionary operators are manipulated in evolutionary computing (EC) based techniques, such as evolutionary programming [1], evolution strategies, genetic algorithms [2], differential evolution [3], genetic programming [4], as well as swarm based algorithms such as artificial immune system [5], particle swarm optimization [6], ant colony optimization [7], honey-bees optimization [8], and artificial bee colony optimization [9].

Invented by Cramer in 1985 [10], genetic programming (GP) is the most popular technique used in symbolic regression. GP can be defined as an extended version of genetic algorithms (GAs), where the main difference relies on the structure and the meaning of the representation [4, 11]. GP and GAs have common operators which are the crossover, the mutation, and the permutation operators, while the main difference between GAs and GP consists in individuals used by GAs which are linear strings of fixed length (chromosomes). Alternatively, the individuals manipulated by GP are nonlinear entities of different sizes and shapes (parse trees) [11].

In [12], a new technique for constructing programs through Ant Colony Optimization (ACO) using the tree adjunct grammar (TAG) formalism is presented and the results are very promising. Immune programming uses artificial immune system, as optimal search engine as reported in [13].

Swarm intelligence is an artificial intelligence concept which involves studies of collective behaviors in decentralized natural or artificial systems. Swarm based algorithms have shown good results in many important applications, such as optimization [14, 15], pattern recognition [16], machine learning [17], clustering [18-20], data mining [21], and function approximation [13].

Artificial bee colony algorithm, introduced by Karaboga in [22], simulates the foraging behavior of honey bee swarms. The ABC algorithm was tested on a wide range of real-world problems and compared to other well-known evolutionary computing such as particle swarm optimization, genetic algorithm, and differential evolution. The comparison results demonstrated clearly the performance of ABC algorithm which shows considerable improvements on most population-based algorithms.

For practical implementation of population-based methods, there is always a need to carefully address the problem of setting the number of function evaluations (FEs). The number of FEs, which is the size of the population by the number of iterations, impacts directly the solution quality as well as the running time. For computational performance, it is desired to deal efficiently with this compromise in a way to minimize the number of FEs without having to deteriorate the quality of the solution. Throughout this work, an analysis study on the computational performance of artificial bee colony programming (ABCP) algorithm is presented. The ABCP was originally introduced by Karaboga in [23] and shows important features that might need further assessment, in particular the control parameter settings. Here, ABCP's performance with respect to evaluations number is studied in order to make general rules on control parameters setting which is a challenging issue in the analysis of the performance of meta-heuristic optimization methods. The results might help to set wisely the ABC control parameters to obtain the best results by means of a minimal number of function evaluations, i.e. a short running time.

Briefly, this article is organized as follows. Section II introduces the concept of artificial bee colony (ABC) optimization. Section III describes the artificial bee colony programming (ABCP) concept. The computational performance of the ABCP is analyzed through simulations by considering different benchmark problems in Section IV. Concluding remarks are finally given in Section V.

II. Artificial bee colony optimization

Artificial bee colony (ABC) optimization is a swarm intelligence based technique which was originally proposed by Karaboga [22, 24] to solve numerical function optimization. ABC algorithm simulates the foraging behavior of honey bees that are categorized into three main groups: employed bees, onlooker bees and scout bees. Based on two essential leading modes of honey bee colony which are recruitment to a food source and abandonment of a source, the process of bees seeking for sources

with high amount of nectar is the one applied to find the optimal solution for a given optimization problem.

In ABC model, three main phases are considered: employed bee phase, onlooker phase and scout phase. Employed bees investigate their food sources and share the nectar and the position information of these sources with onlooker bees. Based on a greedy selection, onlooker bees will have to choose food sources with high profitability. The employed bee whose food source has been abandoned by the bees becomes a scout bee. The algorithmic structure of ABC concept defines the position of a food source as a possible solution to the optimization problem. The nectar amount of that source represents the fitness of the associated solution. Food source positions are generated using the following equation:

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right) \qquad (1)$$

Each solution xi, (i=1,2,…,SN), is a D-dimensional vector of optimization parameters, where SN is the size of the colony. In the employed phase, an employed bee produces a modification on the position of the food source in her memory and finds a neighboring food source according to the following expression:

$$v_{ij} = x_{ij} + \emptyset_{ij}\left(x_{ij} - x_{kj}\right) \qquad (2)$$

where $\emptyset_{ij}$ is a random number in the interval [-1, 1] and $k \in \{1,2,\ldots,SN\}$ with $k \neq i$ and $j \in \{1,2,...,D\}$ are randomly chosen indexes, D is the dimension of the problem. Greedy selection between the old and the updated food source position is performed by the employed bee based on fitness value evaluation. This valuable information about the position and the quality of the food sources are shared with the onlooker bees.

In the onlooker phase, an onlooker bee evaluates the information provided by the employed bees and selects a food source depending on its probability value P_i. The probability of a food source being selected by the onlooker bees increases as the fitness value of a food source increases. After selecting the food source, an onlooker bee produces a modification on the position of that site using the same mechanism as in (2). Greedy selection is also applied by onlooker bees so that new food sources with high nectar are memorized. During scout phase, any solution that cannot be improved through a predefined number of generations will be abandoned and replaced by a new position that is randomly determined by a scout bee according to (1).

III. Artificial bee colony programming

Similarly to GP and associated GA concepts, artificial bee colony programming (ABCP) is an adaptation of artificial bee colony algorithm that deals with the problem of symbolic regression. The representation of a food source defined in ABC as a string with a fixed length cannot simply be used in ABCP because of the complex structure of the solutions. Alternatively, a parse tree representation is used as proposed in [4]. The food source position is composed of terminals and functions such as arithmetic operations, mathematical functions and logical functions.

The quality of each food source called fitness measurement is measured by evaluating the performance of each individual, and shows how much the result of obtained function fits with the target one. The success rate, desired to be close to 100%, is used in fitness measurement procedure, as defined in [12]. The success rate is chosen to indicate the success ratio of finding the exact solution by ABCP and is given as:

$$\text{Success rate} = \frac{\text{Number of successful runs}}{\text{total number of runs}} \qquad (3)$$

The steps of ABCP are described as follows. After initial colony generation using Ramped half-and-half method (to avoid duplicate individuals as suggested by Koza in [4]), the optimization process starts with the employed bee phase where new functions are generated and evaluated subsequently. This step is followed by the onlooker bee phase, which consists in producing and evaluating new functions depending on their quality. After employed and onlooker bee phases, the algorithm checks the unimproved functions for which the number of fail trials exceeds the limit value. Any solution that cannot be improved will be replaced by a scout bee which generates a new function by using the grow method with considering duplications. These steps are iteratively executed until the termination criterion has been satisfied.

The adaptation of the ABC to the problem of automatic programming consists in the mechanism of producing candidate solutions, called information sharing mechanism. The solutions' structures in ABCP are tree based, different than those used in ABC. Therefore, the search processes used in ABC cannot be used directly. The information sharing mechanism consists in choosing randomly a tree node from a neighborhood solution (x_k) either a function in the probability of P_{ip} (set to 0.9) or a terminal in the probability of 1-P_{ip}. Then, a tree node in the current solution (x_i) is also randomly chosen in P_{ip} probability distribution. Then, the node from the neighborhood solution (x_k) will replace the node from the current solution (x_i) to obtain the candidate solution (v_i).

The pseudo-code of the ABCP is defined as follows.

1: Generate initial functions (x_i) with Ramped half-and-half method

2: Evaluate the initial functions

3: repeat

4: FOR each employed bee {

Produce new function (v_i) by using information sharing mechanism

Evaluate the functions

Apply greedy selection process between (x_i) and (v_i)}

5: Calculate the probability values pi for the functions

6: FOR each onlooker bee {

Select a function xi depending on pi probabilistically

Produce new function (v_i) by using information sharing mechanism

Evaluate the functions

Apply greedy selection process between (x_i)and (v_i)}

7: If there is an abandoned function then replace it with a new function generated by grow method by a scout using the grow method

8: Memorize the best solution so far

9: cycle = cycle + 1

10: until maxcycle

IV. Experiments and results

In this Section, the computational performance of the automatic programming ABCP strategy is analyzed through different

simulations with respect to two control parameters which are the colony size and the number of iterations. To this end, four symbolic regression benchmark problems are considered. These problems, which are described in Table I, are taken from the literature [23, 25]. For fair comparison, same parameter values are set for all the considered problems, except the colony size and the number of iterations which are managed in a way to maintain the number of function evaluations constant. This implies that the number of function evaluations which is equal to the size of the colony by the number of iterations is taken almost same for all tests. The aim is to study the impact of the number of function evaluations on ABCP's performance by considering different colony sizes and iterations with different experiments conducted on a a same running period. Table II shows the control parameters used for ABCP in this study.

In each experiment, 50 runs were conducted. Table III shows the success rate for 100 different runs with same parameter settings. Case 1, case 2 and case 3 represent respectively experiments conducted with population size of 250 and 30 iterations, population of 150 individuals and 50 iterations, and 100 individuals and 75 iterations. Those parameters are set to get the same number of function evaluations which is fixed at the value 7500. The success rate is computed according to (3).

TABLE I. SYMBOLIC REGRESSION BENCHMARK PROBLEMS

Functions	**Fitcases**
F1=x^2+x	20 uniformly points [-1,1]
F2=x^3+x^2+x	
F3=x^4+x^3+x^2+x	
F4=x^5+x^4+x^3+x^^2+x	
F5=cos(2x)	

TABLE II. CONTROL PARAMETERS FOR ABCP

Parameter	**Value**
Initial max. depth	6
Max depth	15
Terminal set	The variable x and the constant 1.0
Function set	F = {+,−,*,/,sin}
Successful run	An individual hits on all fitness cases
# Of runs	100 Independent runs

TABLE III. THE SUCCESS RATE OF ABCP

	Case 1	**Case 2**	**Case 3**
F1	94%	92%	90%
F2	56%	46%	42%
F3	46%	26%	26%
F4	14%	18%	18%

The performance of the ABCP strategy obtained for different colony sizes and different number of iterations is shown in Table III. For function F1, the three experiments found the exact solution almost all the time. This is due to the slight structural complexity of the target function. For functions F2 and F3, the experiments with high colony size outperform the remaining experiments. However, this does not hold for function F4 where it can be seen that employing high colony size induced bad rate success, and for the same number of evaluations, the result gets better with the increase of the number of iterations.

As a result, one might notice that when the number of iterations increases, i.e. the size of the colony decreases, the success rate gets worst. We can then conclude that high colony size can give better results for functions with small complexity, while functions with high complexity need to be processed with high number of iterations for concluding results.

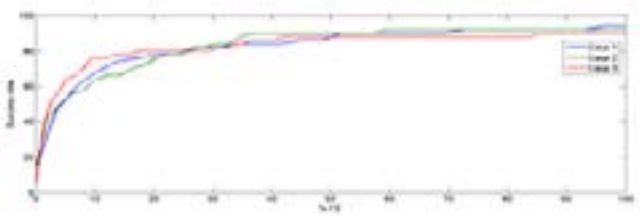

Fig.1. Evolution of success rate for test function F1.

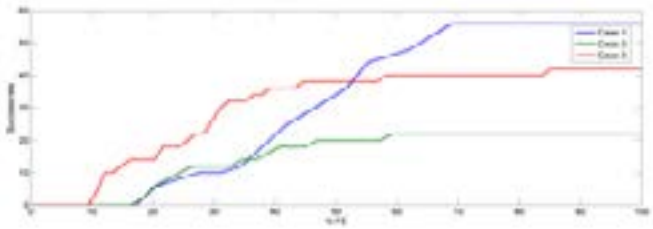

Fig.2. Evolution of success rate for test function F2.

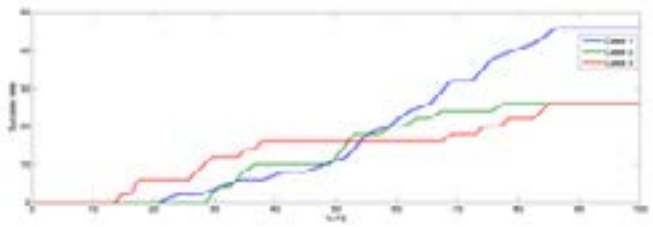

Fig.3. Evolution of success rate for test function F3.

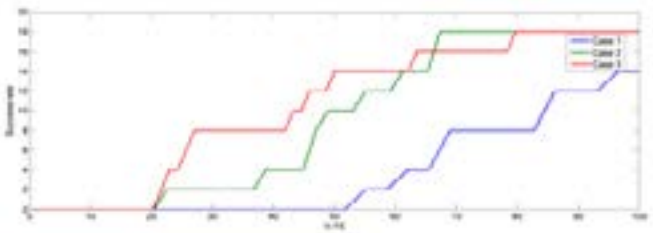

Fig.4. Evolution of success rate for test function F4.

Figures 1-4 depict the evolution of success rate for each test case with regard to the number of function evaluations. It is clear that ABCP with high population size (case 1) achieves the best success rate except for function F4 where high population size shows the worst results, which strength the conclusion deducted from the result of successful rate.

V. CONCLUSION

This paper presents an analysis study on the computational performance of artificial bee colony programming (ABCP) strategy with respect to some control parameters setting. The ABCP is a newly introduced swarm-based approach to evolve programs using the artificial bee colony optimization algorithm. In this study, we have shown the results of some experiments conducted on four symbolic regression benchmark problems. More precisely, we have evaluated the influence of two different control parameters on solution quality, namely the population size and

the number of iterations. The population size and the number of iterations determine the evaluation number which impacts the run time. The run time of evolutionary computing algorithm is a major problem. So the wise choice of the population size and the number of iterations to get the better results with the small evaluation number is always desirable

The obtained results for different test cases might serve as a basement to set general rules on the setting of control parameters of the ABCP algorithm for a given automatic modeling problem. Future works may focus on applications to high-dimensional scientific and engineering problems solving using ABCP strategy.

References

[4] U. K. Chakraborty, "Genetic and evolutionary computing," ed: Elsevier, 2008.

[5] D. P. Searson, "GPTIPS 2: an open-source software platform for symbolic data mining," in Handbook of genetic programming applications, ed: Springer, 2015, pp. 551-573.

[6] R. Funaki, H. Takano, and J. Murata, "Tree structure based differential evolution for optimization of trees and interactive evolutionary computation," in Society of Instrument and Control Engineers of Japan (SICE), 2015 54th Annual Conference of the, 2015, pp. 331-336.

[7] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection vol. 1: MIT press, 1992.

[8] [5] L. N. De Castro and F. J. Von Zuben, "Artificial immune systems: Part I–basic theory and applications," Universidade Estadual de Campinas, Dezembro de, Tech. Rep, vol. 210, 1999.

[9] J. Sun, C.-H. Lai, and X.-J. Wu, Particle swarm optimisation: classical and quantum perspectives: CRC Press, 2016.

[10] [7] M. Boryczka, "Ant Colony Programming: Application of Ant Colony System," Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications: Emerging Trends and Applications, p. 248, 2009.

[11] H. A. Abbass, "MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach," in Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, 2001, pp. 207-214.

[12] F. Harfouchi, H. Habbi, C. Ozturk, and D. Karaboga, "Modified multiple search cooperative foraging strategy for improved artificial bee colony optimization with robustness analysis," Soft Computing, vol. 22, pp. 6371-6394, 2018.

[13] [10] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in Proceedings of the first international conference on genetic algorithms, 1985, pp. 183-187.

[14] Z. Gan, T. W. Chow, and W. N. Chau, "Clone selection programming and its application to symbolic regression," Expert Systems with Applications, vol. 36, pp. 3996-4005, 2009.

[15] H. A. Abbass, X. Hoai, and R. I. Mckay, "AntTAG: A new method to compose computer programs using colonies of ants," in Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, 2002, pp. 1654-1659.

[16] P. Musilek, A. Lau, M. Reformat, and L. Wyard-Scott, "Immune programming," Information Sciences, vol. 176, pp. 972-1002, 2006.

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

[18] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," Advances in Engineering Software, vol. 105, pp. 30-47, 2017.

[19] O. Lamraoui and H. Habbi, "H-infinity fuzzy emulator design for multivariable control of drum boiler-turbine unit," in Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of, 2014, pp. 428-433.

[20] J. Kennedy, "Particle swarm optimization," in Encyclopedia of machine learning, ed: Springer, 2011, pp. 760-766.

[21] O. Lamraoui, Y. Boudouaoui, and H. Habbi, "Heat transfer dynamics modelling by means of clustering and swarm methods," International Journal of Intelligent Engineering Informatics, vol. 7, pp. 346-365, 2019.

[22] [Y. Boudouaoui, H. Habbi, and F. Harfouchi, "Swarm Bee Colony Optimization for Heat Exchanger Distributed Dynamics Approximation With Application to Leak Detection," in Handbook of Research on Emergent Applications of Optimization Algorithms, ed: IGI Global, 2018, pp. 557-578.

[23] H. Habbi, Y. Boudouaoui, D. Karaboga, and C. Ozturk, "Self-generated fuzzy systems design using artificial bee colony optimization," Information Sciences, vol. 295, pp. 145-159, 2015.

[24] A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," Artificial Intelligence Review, vol. 44, pp. 23-45, 2015.

[25] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department2005.

[26] D. Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, "Artificial bee colony programming for symbolic regression," Information Sciences, vol. 209, pp. 1-15, 2012.

[27] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," Applied soft computing, vol. 8, pp. 687-697, 2008.

[28] S. Shirakawa, S. Ogino, and T. Nagao, "Automatic construction of programs using dynamic ant programming," in Ant Colony Optimization-Methods and Applications, ed: InTech, 2011.