Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems

Dervis Karaboga and Bahriye Basturk

Erciyes University, Engineering Faculty, The Department of Computer Engineering
karaboga@erciyes.edu.tr, bahriye@erciyes.edu.tr

Abstract. This paper presents the comparison results on the performance of the Artificial Bee Colony (ABC) algorithm for constrained optimization problems. The ABC algorithm has been firstly proposed for unconstrained optimization problems and showed that it has superior performance on these kind of problems. In this paper, the ABC algorithm has been extended for solving constrained optimization problems and applied to a set of constrained problems .

1 Introduction

Constrained Optimization problems are encountered in numerous applications. Structural optimization, engineering design, VLSI design, economics, allocation and location problems are just a few of the scientific fields in which CO problems are frequently met [1]. The considered problem is reformulated so as to take the form of optimizing two functions, the objective function and the constraint violation function [2]. General constrained optimization problem is to find $\boldsymbol{x}$ so as to

$$\text{minimize } f(x), \qquad x = (x_1, \ldots, x_n) \in \mathbb{R}^n$$

where $\boldsymbol{x} \in \mathbb{F} \in \mathbb{S}$. The objective function f is defined on the search space $\mathbb{S} \subseteq \mathbb{R}^n$ and the set $\mathbb{F} \subseteq \mathbb{S}$ defines the feasible region. Usually, the search space $\mathbb{S}$ is defined as a n-dimensional rectangle in $\Re^n$ (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x(i) \leq u(i), \quad 1 \leq i \leq n$$

whereas the feasible region $\mathbb{F} \subseteq \mathbb{S}$ is defined by a set of m additional constraints ($m \geq 0$):

$$g_j(\boldsymbol{x}) \leq 0, \text{ for } j = 1, \ldots, q$$

$$h_j(\boldsymbol{x}) = 0, \text{ for } j = q+1, \ldots, m.$$

P. Melin et al. (Eds.): IFSA 2007, LNAI 4529, pp. 789–798, 2007.

At any point $\boldsymbol{x} \in \mathbb{F}$, the constraints g_k that satisfy $g_k(\boldsymbol{x}) = 0$ are called the active constraints at $\boldsymbol{x}$. By extension, equality constraints h_j are also called active at all points of $\mathbb{S}$ [3].

Different deterministic as well as stochastic algorithms have been developed for tackling constrained optimization problems. Deterministic approaches such as Feasible Direction and Generalized Gradient Descent make strong assumptions on the continuity and differentiability of the objective function [4,5]. Therefore their applicability is limited since these characteristics are rarely met in problems that arise in real life applications. On the other hand, stochastic optimization algorithms such as Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Particle Swarm Optimization (PSO) do not make such assumptions and they have been successfully applied for tackling constrained optimization problems during the past few years [6,7,8,9,16].

Karaboga has described an Artificial Bee Colony (ABC) algorithm based on the foraging behaviour of honey bees for numerical optimization problems [11]. Karaboga and Basturk have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) on unconstrained problems [12]. In this work, ABC algorithm is extended for solving constrained optimization (CO) problems. Extension of the algorithm depends on replacing the selection mechanism of the simple ABC algorithm with Deb's [13] selection mechanism in order to cope with the constraints. The performance of the algorithm has been tested on 13 well-known constrained optimization problems taken from the literature and compared with Particle Swarm Optimization (PSO) and Differential Evolution (DE) [14]. The Particle Swarm Optimization (PSO) algorithm was introduced by Eberhart and Kennedy in 1995 [15]. PSO is a population based stochastic optimization technique and well adapted to the optimization of nonlinear functions in multidimensional space. It models the social behaviour of bird flocking or fish schooling. The DE algorithm is also a population based algorithm using crossover, mutation and selection operators. Although DE uses crossover and mutation operators as in GA, the main operation is based on the differences of randomly sampled pairs of solutions in the population. Paper is organized as follows. In Section II, the ABC algorithm and the ABC algorithm adapted for solving constrained optimization problems are introduced. In Section III, a benchmark of 13 constrained functions are tested. Results of the comparison with the PSO and DE algorithms are presented and discussed. Finally, a conclusion is provided.

2 Artificial Bee Colony Algorithm

2.1 The ABC Algorithm Used for Unconstrained Optimization Problems

In ABC algorithm [11,12], the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. First half of the colony consists of the employed artificial bees and the second half includes the onlookers. For

every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose the food source has been abandoned by the bees becomes a scout.

In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. At the first step, the ABC generates a randomly distributed initial population $P(G = 0)$ of SN solutions (food source positions), where SN denotes the size of population. Each solution x_i $(i = 1, 2, ..., SN)$ is a D-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles, $C = 1, 2, ..., MCN$, of the search processes of the employed bees, the onlooker bees and scout bees. An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution). Provided that the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise she keeps the position of the previous one in her memory. After all employed bees complete the search process, they share the nectar information of the food sources and their position information with the onlooker bees on the dance area. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. As in the case of the employed bee, she produces a modification on the position in her memory and checks the nectar amount of the candidate source. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, p_i , calculated by the following expression (1):

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{1}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees (BN).

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression (2):

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

where $k \in \{1, 2,..., SN\}$ and $j \in \{1, 2,..., D\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i. $\phi_{i,j}$ is a random number between [-1, 1]. It controls the production of neighbour food sources around $x_{i,j}$ and represents the comparison of two food positions visually by a bee. As can be seen from (2), as the difference between the parameters of the $x_{i,j}$ and $x_{k,j}$ decreases, the perturbation on the position $x_{i,j}$ gets decrease, too. Thus, as the search approaches to 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. In this work, the value of the parameter exceeding its limit is set to its limit value.

The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by producing a position randomly and replacing it with the abandoned one. In ABC, providing that a position can not be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called "*limit*" for abandonment. Assume that the abandoned source is x_i and $j \in \{1, 2,..., D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as in (3)

$$x_i^j = x_{\min}^j + \text{rand}(0, 1)(x_{\max}^j - x_{\min}^j) \tag{3}$$

After each candidate source position $v_{i,j}$ is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food has an equal or better nectar than the old source, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one.

It is clear from the above explanation that there are four control parameters used in the ABC: The number of food sources which is equal to the number of employed or onlooker bees (SN), the value of *limit*, the maximum cycle number (MCN).

Detailed pseudo-code of the ABC algorithm is given below:

1: Initialize the population of solutions $x_{i,j}, i = 1 \ldots SN, j = 1 \ldots D$
2: Evaluate the population
3: cycle=1
4: **repeat**
5: Produce new solutions $v_{i,j}$ for the employed bees by using (2) and evaluate them
6: Apply the greedy selection process
7: Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ by (1)
8: Produce the new solutions $v_{i,j}$ for the onlookers from the solutions $x_{i,j}$ selected depending on $P_{i,j}$ and evaluate them
9: Apply the greedy selection process

10: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{i,j}$by (3)
11: Memorize the best solution achieved so far
12: cycle=cycle+1
13: **until** cycle=MCN

2.2 The ABC Algorithm Used for Constrained Optimization Problems

In order to adapt the ABC algorithm for solving constrained optimization problems, we adopted Deb's constrained handling method [13] instead of the selection process (greedy selection) of the ABC algorithm described in the previous section since Deb's method consists of very simple three heuristic rules. Deb's method uses a tournament selection operator, where two solutions are compared at a time, and the following criteria are always enforced: **1)** Any feasible solution is preferred to any infeasible solution, **2)** Among two feasible solutions, the one having better objective function value is preferred, **3)** Among two infeasible solutions, the one having smaller constraint violation is preferred.

Because initialization with feasible solutions is very time consuming process and in some cases it is impossible to produce a feasible solution randomly, the ABC algorithm does not consider the initial population to be feasible. Structure of the algorithm already directs the solutions to feasible region in running process due to the Deb's rules employed instead of greedy selection. Scout production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population.

In order to produce a candidate food position from the old one in memory, the adapted ABC algorithm uses the following expression:

$$\upsilon_j = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \text{ , } if \quad R_j < MR \\ x_{ij} \qquad\qquad\qquad\qquad \text{, otherwise} \end{cases} \tag{4}$$

where $k \in \{1, 2,..., SN\}$ is randomly chosen index. Although k is determined randomly, it has to be different from i. R_j is randomly chosen real number in the range [0,1] and $j \in \{1, 2,..., D\}$. MR, modification rate, is a control parameter that controls whether the parameter x_{ij} will be modified or not. In the version of the ABC algorithm proposed for constrained optimization problems, artificial scouts are produced at a predetermined period of cycles for discovering new food sources randomly. This period is another control parameter called scout production period (SPP) of the algorithm. At each SPP cycle, it is controlled if there is an abandoned food source or not. If there is, a scout production process is carried out.

Pseudo-code of the ABC algorithm proposed for solving constrained problems is given below:

1: Initialize the population of solutions $x_{i,j}, i = 1 \ldots SN, j = 1 \ldots D$
2: Evaluate the population
3: cycle=1

4: **repeat**
5: Produce new solutions $v_{i,j}$ for the employed bees by using (4) and evaluate them
6: Apply selection process based on Deb's method
7: Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ by (1)
8: Produce the new solutions $v_{i,j}$ for the onlookers from the solutions $x_{i,j}$ selected depending on $P_{i,j}$ and evaluate them
9: Apply selection process based on Deb's method
10: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{i,j}$by (3)
11: Memorize the best solution achieved so far
12: cycle=cycle+1
13: **until** cycle=MCN

3 Experimental Study and Discussion

In order to evaluate the performance of the ABC algorithm, we used a set of 13 benchmark problems can be found in [16]. This set includes various forms of objective function such as linear, nonlinear and quadratic. The performance of the ABC algorithm is compared with that of the differential evolution (DE) and particle swarm optimization (PSO) algorithms.

3.1 Settings

PSO employs Deb's rules for constraint handling. The swarm size is 50 and the generation number is 7000. Hence, PSO performs 350 000 objective function evaluations. Cognitive and social components are both set to 1. Inertia weight is uniform random real number in the range [0.5,1]. All equality constraints are converted into inequality constraints, $|h_j| \leq \epsilon$ with ϵ=0.001 [16].

In DE, F is a real constant which affects the differential variation between two solutions and set to 0.5 in our experiments. Value of crossover rate, which controls the change of the diversity of the population, is chosen to be 0.9 as recommended in [17]. Population size is 40 , maximum generation number is 6000 and it uses Deb's rules.

In ABC, the value of modification rate (MR) is 0.8, colony size ($2 * SN$) is 40 and the maximum cycle number (MCN) is 6000. So, the total objective function evaluation number is 240 000 as in DE. The value of limit" is equal to SN x D where D is the dimension of the problem and SPP is also SNxD. Experiments were repeated 30 times each starting from a random population with different seeds.

3.2 Results and Discussion

The results of the experiments for the ABC algorithm are given in Table 1. Comparative results of the best, mean and worst solutions of the investigated algorithms are presented in Table 2, Table 4 and Table 3, respectively.

Table 1. Statistical Results Obtained by the ABC algorithm for 13 test functions over 30 independent runs using 240.000 objective function evaluations

Problem	Optimal	Best	Mean	Worst	Std. Dev.
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803598	0.792412	0.749797	0.012
g03	1.000	1.000	1.000	1.000	0.000
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.484	5185.714	5438.387	75.358
g06	-6961.814	-6961.814	-6961.813	-6961.805	0.002
g07	24.306	24.330	24.473	25.190	0.186
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.634	680.640	680.653	0.004
g10	7049.25	7053.904	7224.407	7604.132	133.870
g11	0.75	0.750	0.750	0.750	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.760	0.968	1.000	0.055

Table 2. The Best Solutions Obtained by DE, PSO and ABC algorithms for 13 test functions over 30 independent runs. – Means That No Feasible Solutions Were Found. Na = Not Available.

P	Optimal	PSO [16]	DE	ABC
g01	-15.000	15.000	-15.000	-15.000
g02	0.803619	0.669158	0.472	0.803598
g03	1.000	0.993930	1.000	1.000
g04	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	5126.484	5126.484	5126.484
g06	-6961.814	-6161.814	-6954.434	-6961.814
g07	24.306	24.370153	24.306	24.330
g08	0.095825	0.095825	0.095825	0.095825
g09	680.63	680.630	680.630	680.634
g10	7049.25	7049.381	7049.248	7053.904
g11	0.75	0.749	0.752	0.750
g12	1.000	1.000	1.00	1.000
g13	0.053950	0.085655	0.385	0.760

As seen from Table 2, the ABC algorithm has found the global minimum of the seven of thirteen problems (g01, g03, g04, g06, g08, g11, g12) through 240 000 cycles. On five functions (g02, g04, g05, g07, g10), the ABC algorithm produced results quite close to the global optima. On one problem, g13, the ABC algorithm could not find the optima in the specified maximum number of cycles.

As seen from Table 2, PSO algorithm is better than ABC on three problems (g09,g10,g13) while the ABC algorithm shows better performance than PSO on four problems (g02, g03, g07, g12). Compared to DE, it is better than ABC on

Table 3. The Worst Solutions Obtained by DE, PSO and ABC algorithms for 13 test functions over 30 runs. – Means That No Feasible Solutions Were Found. Na = Not Available.

P	Optimal	PSO [16]	DE	ABC
g01	-15.000	-13.000	-11.828	-15.000
g02	0.803619	0.299426	0.472	0.749797
g03	1.000	0.464	1.000	1.000
g04	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	5249.825	5534.610	5438.387
g06	-6961.814	-6961.814	-6954.434	-6961.805
g07	24.306	56.055	24.330	25.190
g08	0.095825	0.095825	0.095825	0.095825
g09	680.63	680.631	680.631	680.653
g10	7049.25	7894.812	9264.886	7604.132
g11	0.75	0.749	1	0.750
g12	1.000	0.994	1.000	1.000
g13	0.053950	1.793361	0.990	1.000

Table 4. The Mean Solutions Results Obtained by DE, PSO and ABC algorithms for 13 test functions over 30 independent runs and total success numbers of algorithms. A Result In Boldface Indicates A Better Result Or That The Global Optimum (Or Best Known Solution) Was Reached. – Means That No Feasible Solutions Were Found.

P	Optimal	PSO [16]	DE	ABC
g01	-15.000	-14.710	-14.555	**-15.000**
g02	0.803619	0.419960	0.665	**0.792412**
g03	1.000	0.764813	**1.000**	**1.000**
g04	-30665.539	**-30665.539**	**-30665.539**	**-30665.539**
g05	5126.498	**5135.973**	5264.270	5185.714
g06	-6961.814	**-6961.814**	–	-6961.813
g07	24.306	32.407	**24.310**	24.473
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.630**	**680.630**	680.640
g10	7049.25	7205.5	**7147.334**	7224.407
g11	0.75	0.749	0.901	**0.750**
g12	1.000	0.998875	**1.000**	**1.000**
g13	0.053950	**0.569358**	0.872	0.968

four functions (g07, g09, g10, g13) as the ABC algorithm is better than DE on three problems (g02, g06, g11) with respect to the best results.

From the worst results given in Table 3, PSO is better than ABC on three problems (g05, g06, g09) while ABC outperforms PSO on eight problems (g01, g02, g03, g07, g10, g11, g12, g13). DE show better performance on two problems with respect to the ABC algorithm on three problems (g07, g09,g13) while ABC is better on six problems (g01, g02, g05, g06, g10, g11).

Similarly, with respect to the mean solutions in Table 4, PSO shows better performance with respect to the ABC algorithm on five problems (g05, g06, g09, g10, g13) and ABC algorithm is better than PSO on six problems (g01, g02, g03, g07, g11, g12). DE has better performance than ABC on four problems (g07, g09, g10, g13) while ABC is better than DE on five problems (g01, g02, g05, g06, g11).

From the mean results presented in Table 4, it can be concluded that the ABC algorithm performs better than DE and PSO.

Consequently, the ABC algorithm using Deb's rules can not find the optimum solution for g05, g10 and g13 for each run. g05 and g13 are nonlinear problems and the value of ρ ($\rho = |\mathbb{F}| / |\mathbb{S}|$, F:Feasible Space, S:Search Space) for g05 and g13 is %0.000. Also, these problems have nonlinear equality constraints. g10 is linear and the value of ρ is %0.0020 for this problem. However, g10 has no linear equality and nonlinear equality constraints. Therefore, it is not possible to make any generalization for the ABC algorithm such that it is better or not for a specific set of problems. In other words, it is not clear what characteristics of the test problems make it difficult for ABC.

4 Conclusion

A modified version of the ABC algorithm for constrained optimization problems has been introduced and its performance has been compared with that of the state-of-art algorithms. It has been concluded that the ABC algorithm can be efficiently used for solving constrained optimization problems. The performance of the ABC algorithm can be also tested for real engineering problems existing in the literature and compared with that of other algorithms. Also, the effect of constraint handling methods on the performance of the ABC algorithm can be investigated in future works.

References

1. K. E. Parsopoulos, M. N. Vrahatis, *Particle Swarm Optimization Method for Constrained Optimization Problems*, Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies,pp. 214–220. IOS Press, 2002.
2. A. R. Hedar, M. Fukushima, *Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization*
3. Z. Michalewicz, M. Schoenauer, *Evolutionary Algorithms for Constrained Parameter Optimization Problems*, Evolutionary Computation, 4:1 (1995), pp. 1–32.
4. C. A. Floudas, P. M. Pardalos, *A collection of test problems for constrained global optimization algorithms*, In: LNCS. Vol. 455. Springer-Verlag (1987).
5. D. M. Himmelblau, *Applied Nonlinear Programming*, McGrawHill (1972).
6. J. A. Joines, C. R. Houck, *On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with gas*, In: Proc. IEEE Int. Conf. Evol. Comp. (1994) 579-585.
7. X. Hu and R. C. Eberhart,*Solving constrained nonlinear optimization problems with particle swarm optimization.* In Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002.

8. X. Hu, R. C. Eberhart, Y. H. Shi, *Engineering optimization with particle swarm*, IEEE Swarm Intelligence Symposium, 2003: 53-57.
9. K. E. Parsopoulos, , M. N. Vrahatis,A Unified Particle Swarm Optimization for solving constrained engineering optimization problems, dvances in Natural Computation, Pt. 3 , pages 582-591, 2005. Lecture Notes in Computer Science Vol. 3612.
10. A. E. M. Zavala, A. H. Aguirre, E. R. V. Diharce, *Constrained optimization via particle evolutionary swarm optimization algorithm (PESO)* , In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO'05), pp. 209–216.
11. D. Karaboga, *An Idea Based On Honey Bee Swarm For Numerical Optimization*, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
12. B. Basturk, D.Karaboga, *An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization*, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.
13. D.E. Goldberg, K. Deb, A comparison of selection schemes used in genetic algorithms Foundations of Genetic Algorithms, edited by G. J. E. Rawlins, pp. 69-93,1991.
14. R. Storn, K. Price, , *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces.*, Journal of Global Optimization, 11(1997), pp. 341–359.
15. J. Kennedy, R. C. Eberhart,*Particle swarm optimization*,1995 IEEE International Conference on Neural Networks, 4 (1995), 1942–1948.
16. A. E. M. Zavala, A. H. Aguirre, E. R. V. Diharce, *Constrained optimization via particle evolutionary swarm optimization algorithm (PESO)* , In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO'05), pp. 209–216.
17. D. Corne, M. Dorigo, F. Glover.,(Editors), *New Ideas in Optimization*, McGraw-Hill, 1999.