Artificial Bee Colony Algorithm

Farah Alyasari
CSE 450 – Honor Contract
Spring of 2019

Abstract
Algorithms are not exclusive to the field of technology. This paper explores a representative swarm-based algorithm inspired by nature. Although recently defined in 2005, the Artificial Bee Colony Algorithm has already been applied to solve challenging problems in Computer Science.

Keywords: Bio-inspired Algorithms, Artificial Bee Colony

I. INTRODUCTION

The Artificial Bee Colony (ABC) algorithm is a new swarm-based optimization algorithm, inspired by the behavior of honey bees foraging plants [1]. Swarm-based algorithms are results from the study of swarm intelligence, which examines the behavior of large numbers of homogeneous agents working collectively in the same environment to search and optimize a solution to a problem [2]. In the ABC algorithm, the agents are the bees, the environment is nature, and the problem statement is to collect quality nectar from flower patches using an optimal route. In application, the agents, environment, and problem statement are changed respectively based on the search space. Also, because the ABC algorithm does not always yield the optimal path solution, it is considered a metaheuristics algorithm, meaning that it provides a sufficiently good solution to an optimization problem [3]. The following sections discuss the ABC pseudocode, efficiency, and selected applications of the ABC algorithm.

II. PSEUDOCODE

A. Overview

The main steps of the ABC algorithm are below, from [4], defining the original ABC algorithm:

1. Initialize the food source positions.
2. Employed bees use their memory to search for new food sources with more nectar than their neighboring food source.
3. After the employed bees find a new food source, fitness is calculated for this new food source
4. The greedy approach is used to choose between the neighboring food source and the new food source
5. Employed bees share food source information with onlooker bees
6. Onlooker bees use the fitness information shared from the employed bees to select a food source to forage probabilistically
7. Onlooker bees evaluate the fitness of neighboring food sources of the selected food source using a greedy approach
8. Scout bees occur when employed bees exceed a predefined limit
9. Scout bees search for food sources randomly

The ABC algorithm consists of three essential components: employed bees, unemployed bees, and food sources. Employed bees exploit food sources close to their hives [6]. They carry with them information about

the source distance and direction of the food source from the nest [4]. After they exploit the food, they share information about this source to the unemployed bees. There are two types of unemployed bees: onlooker bees and scout bees. Onlooker bees select food sources based on the information found by the employed bees and select the sources with the highest fitness [7]. Scout bees are employed initially. They became scout bees when they abandoned their food sources to search for new and more fit food sources [7]. In the original ABC algorithm, the scout bees search randomly without internal or external clues. During the algorithm, bees show two modes of behavior to communicate together. The first mode is recruiting bees to rich food sources resulting in positive feedback, and the second mode is when bees abandon poor sources initiating negative feedback [6]. The positive feedback occurs when onlook bees select highly fit food sources, and negative feedback occurs when employed bees leave a set of food sources and become scout bees. The feedback is important to keep the colony organized, and help the bees collectively find a sufficiently right solution.

B. Details

The ABC algorithm is intended to find a sufficiently good solution to a global optimization problem. As [6] stated, a global optimization problem is "defined as finding the vector x that minimizes an object function f(x)", seen in equation (1), and according to the conditions in equations (2 – 4)

$$\text{minimize } f(\vec{x}), \text{ for a vector } \vec{x} = (x_1, \dots, x_n,) \in R^n \quad (1)$$

$$l_i \leq x_i \leq u_i, \quad i = 1, \dots, n \quad (2)$$

The problem $f(\vec{x})$ is defined "on a search space, S, which is n – dimensional rectangle in R^n ($S \in R^n$)", and equation (2) gives the lower and upper bounds which constrain the variable domains [6].

There are several classifications of optimization problems, one distinction is between problem which there are no constraints on the variables and problems in which there are constraints on the variables. This distinction for unconstrained or constrained problems hints if there are more bounds or equalities/inequalities. The ABC algorithm can be used for both unconstrained and constrained problems. Equations (3) and (4) have p = 0, and q = 0 for unconstrained problems, and nonzero values for constrained problems. Equation (1) must be fulfilled subject to equations (3) and (4), meaning that for (3) and (4) must be satisfied in the process of solving for (1).

$$g_j(\vec{x}) \leq 0, \text{ for } j = 1, \dots, p \quad (3)$$

$$h_j(\vec{x}) = 0, \text{ for } j = p+1, \dots, q \quad (4)$$

As defined by [6], the schema of the ABC algorithm is as seen in Figure 1 and corresponds to the main steps in the overview section. The Initialization phase corresponds to step 1. Each food source is a possible solution vector with n variables, per equation (1). The number of food sources initialized is equal to the population size, where x_m is the food source and SN is the population size, and equation (5) is one approach proposed by [6] to initialize the i^{th} food source for all m food sources. Note that u_i and l_i can be distinct for each food source, x_m.

$$x_{mi} = l_i + \text{rand}(0,1) * (u_i - l_i) \quad (5)$$

```
Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
  Memorize the best solution achieved so far
UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)
```

Figure 1

The Employed Bees phase corresponds to steps 2 to 5 of the pseudocode in the overview section. The new food source, V_m, is found by randomly selecting a food source, parameter index, and a number within the range of [-a, a]. After a new food source is chosen, "the fitness $fit_m(\vec{x}_m)$ might be calculated for minimization problems using the following formula" [6], where $f_m(\vec{x}_m)$ is used to minimize the value of $\vec{x}_m$:

$$fit_m(\vec{x}_m) = \begin{cases} \frac{1}{1 + f_m(\vec{x}_m)} & if\ f_m(\vec{x}_m) \geq 0 \\ 1 + |f_m(\vec{x}_m)| & if\ f_m(\vec{x}_m) < 0 \end{cases} \qquad (6)$$

The fittest $\vec{x}_m$ is then communicated from the employed bees to the onlooker bees, who are waiting for this piece of information to help them select a food source. Steps 6 and 7 of the pseudocode correspond to the onlooker bees' phase. Equation 7 is one way to probabilistically select food source based on fitness [6].

$$p_m = \frac{fit_m(\vec{x}_m)}{\sum_{m=1}^{SN} f_m(\vec{x}_m)} \qquad (7)$$

Figure 2 shows a summary of variables in the previous equations, for clarity.

Variable	Meaning	Used in Equation
$f(\vec{x})$	a global optimization problem	(1)
$\vec{x}$	a vector to minimize	(1)
l_i	lower bound on $\vec{x}$	(2)
u_i	upper bound on $\vec{x}$	(2)
$g_j(\vec{x})$	constraint on $f(\vec{x})$	(3)
$h_j(\vec{x})$	Constraint on $f(\vec{x})$	(4)
p	zero or nonzero based on $f(\vec{x})$ lower limit	(3)
q	zero or nonzero based on $f(\vec{x})$ upper limit	(3), (4)
x_{mi}	*i*th food source from m total food sources	(5)
$\vec{x}_m$	$\vec{x}_m$ are m food sources	(6)
$\mathbf{f}_m(\vec{x}_m)$	Minimizes $\vec{x}_m$	(6)
$fit_m(\vec{x}_m)$	Selects the most fit $\vec{x}$ of m food sources	(6)
SN	Population size	Implied in (1), (6)
p_m	Probability of selecting a food source m	(7)

Figure 2

III. EFFICIENCY

The ABC algorithm has an advantage over other swarm-based because it applies fewer control parameters [4]. The control parameters of the ABC algorithm are population size, the maximum cycle limit, and limit [6]. The limit is the maximum number of cycles bees keep the food source as it is, without it improving before it is replaced by another food source [9]. The maximum cycle limit is used as one stopping condition for the algorithm. The population size includes the employed bees, onlooker bees, and scout bees. In addition to fewer control parameters, the ABC algorithm performs with comparable efficiency to other swarm-based algorithms [4]. [10] conducted an experimental benchmark on the performance of the ABC algorithm on 50 numerical functions and found that ABC reached the optimal results in 24 functions. The best, worst, and mean running time of the ABC algorithm was noted for each function, with the worst running time among the 50 functions belonging to the Fletcher-Powell function with 10 dimensions. Figure 4 shows the running time of the ABC algorithm for each of the 7 functions in table 3. The results from in Figure 4 are documented by [10].

No.	Function
f1	Sphere
f2	Rosenbrock
f3	Ackley
f4	Griewank
f5	Rastrigin
f6	Zakharov
f7	Bohachevsky3

Figure 3

	Dim	ABC		
	10	Best	Worst	Mean
f1		6,09E-17	2,15E-16	1,1E-16
f2		0,023029	0,929684	0,372901
f3		4,11E-11	2,58E-10	1,3E-10
f4		8,3E-14	0,012247	**0,001082**
f5		0	4,83E-13	4,09E-14
f6		2,489346	21,94919	11,45552
	30	Best	Worst	Mean
f1		4,77E-16	9,64E-16	7,49E-16
f2		0,077905	1,352967	0,405661
f3		2,6E-10	1,29E-09	7,01E-10
f4		2,22E-16	3,44E-12	2,05E-13
f5		5,68E-14	3,31E-09	2,64E-10
f6		212,0386	295,6898	257,8872
	50	Best	Worst	Mean
f1		1,18E-15	2,08E-15	1,73E-15
f2		0,146785	1,711159	0,512477
f3		4,59E-10	3,02E-09	1,3E-09
f4		8,88E-16	2,8E-11	1,15E-12
f5		7,96E-12	0,001047	4,76E-05
f6		421,4083	578,6242	511,0507

Figure 4

The efficiency of the algorithm is excellent for both low dimensions and high dimensions, as it is only slightly worse as the dimension increases. For example, the simple Sphere function has it is best performance be 6,09E-17, after 40 dimensions are added, it performs only 2 scales worse with performance of 1,18E-15. This relationship is observed for other functions.

IV. APPLICATIONS

A. Shortest Path Problem

The Shortest Path Problem (SPP) is one of the most fundamental algorithmic graph problems. It is also an NP-hard combinatorial optimization problem that has longed challenged researcher [11]. Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects, which makes the SPP problem applicable to various real-world scenarios, such as routing protocols used to update forwarding tables in a network and to find an optimal path from one point to another point in geographical maps. Although Bellman-Ford, Dijkstra, and Floyd-Warshall are traditional methods of solving SPP, they are not efficient for large scale graphs, according to [11]. The ABC algorithm gives a procedure for efficiently solving large-scale and complex SPP graphs, which promoted [11] to use the ABC algorithm to solve the SPP problem. Figure 5 shows the ABC-SPP algorithm introduced by [11]'s work.

1. Initialize population of solution(paths) between s and t is generated by ABC algorithm, that included some edges $(i,j) \in E, s \le i,j \le t$,

$$x(i,j) = \begin{cases} 1, if\,(i,j) \in E, \\ 0, otherwise. \end{cases}$$

2. In each solution(path) of the initial population,
   if in the first edge and the last edge of this path sequentially,
   then first node is s and end node is t, continue,
   else change the initial population,
   end.
3. Compute penalty function as following formula:

$$\min E(x,K) = z + [K_1(\sum_{j,(1,j)\in E} x_{1j} - 1) + K_2(1 + \sum_{i,(i,m)\in E} x_{im})] + K_3 \ \Sigma_i(\Sigma_j(\Sigma_k(x_{ij} - x_{ki}))), \big((i,j),(k,i) \in E\big), i \ne s,t, \quad (1)$$

where $z = \sum_i \sum_j x_{ij} w_{ij}$, $(i,j) \in E, i,j,k = 1,2,\dots,m, i \ne j \ne k$, and penalty coefficients are $K_i \gg 1, i = 1,2,3$.

4. In each solution of evaluate fitness of the population.
5. Cycle=1,
6. **While** (stopping criterion(not met),
   a. Forming new population and select sites for neighborhood search.
   b. Select the fittest edge from each path.
   c. Assign remaining edges to search randomly and evaluate their finesses.
   d. Cycle=Cycle+1,
   end.

Figure 3

The control parameters of the ABC algorithms apply similarly to the ABC-SPP algorithm. The population size is the number of paths in total, the control limit and the limit are predefined before running the algorithm. Although the parameters apply the similarly, the ABC-SPP algorithm has a different search environment, thus different agents. The search environment of the ACP-SPP consists of a graph with nodes and edges. The optimal path for the SPP problem is comparable to the optimal food source in the ABC environment. To find the optimal path, first, the initializing process takes place to ensure that each edge has a weight. After the initialization phase, the algorithm begins the process of finding a path. A cost of a path x is the sum of the weights on the edges in x. High fitness of the path x corresponds to low cost, which is done by minimizing the function E, which takes the parameter K as the cost of taking this path. K is also referred to as a penalty. Throughout the algorithm, edges which do not connect d to s are discarded from the population. The ABC-SPP reaches to the shortest path in polynomial time [11].

B. Binary Knapsack Problem

The objective of the Knapsack problem is to place the most profitable subset of elements inside the knapsack without exceeding the weight capacity, it is considered as one of "most popular subset selection problem" [12]. Like the Shortest Path Problem, several researchers who achieved notable results in using the ABC algorithm to solve the Knapsack problem. For example, [13] procedure to generates sufficient solution to the 0 or 1 type of Knapsack problem within "0.01 % of the optimal solution in less than 10% of the required by exact algorithms". The 0 or 1 type of knapsack problem is the basic approach in which an item is either placed in the knapsack or not placed in the knapsack. Another notable work is achieved by [14]. In [14]'s algorithm, the content of the knapsack is analogous to the food sources in the original ABC algorithm. The search environment is the Knapsack hive, and the bees work similarly as in the original ABC algorithm to find near-optimal set of food sources. Figure 6 illustrates the optimization rate with this algorithm. As the number of bees increases, the solution is closer to optimal. As food distribution increases, the solution is less optimal. This observation makes sense as it becomes more challenging to find an optimal solution with more food sources and low number of bees. It is preferred to choose a high number of bees for respective food distribution.

1. Optimization rate

Food distribution / Bees number	5-15	15-45	45-75	75-100	100-150	>150=
10 50	24% 58%	21% 42%	19% 38%	16% 21%	12% 17%	<10% <15%
50 100	56% 71%	46% 59%	42% 51%	39% 47%	31% 41%	<25% <35%
100 150	70% 84%	61% 71%	54% 65%	49% 56%	41% 47%	<30% <35%
150 200	86% 91%	85% 90%	82% 88%	79% 87%	77% 87%	<70% <85%

Figure 4

V. CONCLUSION

The paper introduced the Artificial Bee Colony algorithm. The pseudo code was explained to help the reader visualize the algorithm. Detailed equations of the pseudocode were provided based on the original ABC paper, with some approaches on how to initialize, produce, and select food sources. This paper also illustrated two problems that can be solved using the original ABC or refined ABC algorithm. The original ABC provides a sufficiently good solution to the Shortest Path Problem, one of the most challenging graph problems. Also, the ABC algorithm makes it possible to solve the Knapsack problem. Solving both the Shortest Path and Knapsack problem helps address a variety of other specific problems which fall under these two general problems. In conclusion, the ABC algorithm servers an appropriate alternative to finding a sufficiently good solution to large-scale problems in which there it is inefficient to find the optimal solution.

VI. REFERENCES

[1] H. Jiang, “Artificial Bee Colony algorithm for Traveling Salesman Problem,” Proceedings of the 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering 2015, Dec. 2015.

[2] J. Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes. Chapter 5: Swarm Algorithms. 2011.

[3] P. Overton , The Secret Rules of Modern Living: Algorithms. 2015.

[4] I. Oliveira, R. Schirru, and J. Medeiros, “On the Performance of an Artificial Bee Colony Optimization Algorithm Applied to the Accident Diagnosis in a PWR Nuclear Power Plant ,” 2009.

[5] Online Supplement of the paper entitled "Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization", Artificial Bee Colony (ABC) Algorithm Homepage, 2009.

[6] D. Karaboga, “Artificial bee colony algorithm,” Scholarpedia, vol. 5, no. 3, p. 6915, 2010.

[7] D. Karaboga, “An Idea Based on Honey Bee Swarm for Numerical Optimization,” Oct. 2015.

[8] D. Karaboga, & B. Akay. Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization. 2009.

[9] B. Crawford, R. Soto, R. C. Aguilar, and F. Paredes, “A new Artificial Bee Colony algorithm for Set Covering Problems,” Electrical Engineering and Information Technology, 2014.

[10] A. Ozkis and A. Babalik, “Accelerated ABC (A-ABC) Algorithm for Continuous Optimization Problems,” Lecture Notes on Software Engineering, pp. 262–266, 2013.

[11] P. Mansouri, B. Asady, and N. Gupta, “Solve Shortest Paths Problem by Using Artificial Bee Colony Algorithm,” Advances in Intelligent Systems and Computing Proceedings of the Third International Conference on Soft Computing for Problem Solving, pp. 183–191, 2014.

[12] M. Shokouhifar, S. Sabet, and F. Farokhi , “A Novel Artificial Bee Colony Algorithm for the Knapsack Problem,” Scientific Association of Electrical Engineering , 2012.

[13] F. Faraokhi and M. Shokouhifar, “A novel artificial bee colony algorithm to solve the knapsack problem,” 2012.