SOLVING OPTIMIZATION FUNCTIONS USING ARTIFICIAL BEE COLONY ALGORITHMS

Nuria Gómez Blas, Luis Fernando de Mingo López, Juan Castellanos Peñuela

Abstract: Artificial bee colony is an optimization algorithm, which imitates the real acts of honey bees. The most important components of ABC algorithm are its food source, employed and unemployed bees. The main theme of this algorithm is to arrive at the best food source. This paper takes the advantage of a novel evolutionary algorithm, called artificial bee colony (ABC), to improve the capability of k-means in finding global optimum clusters in nonlinear partitional clustering problems. The shown method is the combination of k-means and ABC algorithms.

Keywords: Clustering, Artificial Bee Colony, Swarm Intelligence.

ITHEA Keywords: F.1.1 Theory of Computation - Models of Computation, I.2.6 Artificial Intelligence.

MSC: 68Q32 Computational learning theory, 68T05 Learning and adaptive systems.

Introduction

Artificial bee colony algorithm (ABC) is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm, proposed by [Karaboga, 2005]. In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts. It is assumed that there is only one artificial employed bee for each food source. In other words, the number of employed bees in the colony is equal to the number of food sources around the hive. Employed bees go to their food source and come back to hive and dance on this area. The employed bee whose food source has been abandoned becomes a scout and starts to search for finding a new food source. Onlookers watch the dances of employed bees and choose food sources depending on dances. The main steps of the algorithm are given below.

In ABC, a population based 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 is equal to the number of solutions in the population. At the first step, a randomly distributed initial population (food source positions) is
generated. After initialization, the population is subjected to repeat the cycles of the search processes of the employed, onlooker, and scout bees, respectively. An employed bee produces a modification on the source position in her memory and discovers a new food source position. Provided that the nectar amount of the new one is higher than that of the previous source, the bee memorizes the new source position and forgets the old one. Otherwise she keeps the position of the one in her memory. After all employed bees complete the search process, they share the position information of the sources with the onlookers on the dance area. Each onlooker evaluates the nectar information taken from all employed bees and then chooses a food source depending on the nectar amounts of sources. As in the case of the employed bee, she produces a modification on the source position in her memory and checks its nectar amount. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one. The sources abandoned are determined and new sources are randomly produced to be replaced with the abandoned ones by artificial scouts.

The most important components of ABC algorithm are its food source, employed and unemployed bees. The main theme of this algorithm is to arrive at the best food source. The standard pseudo code of the ABC algorithm is presented in Figure 1.

The most basic ABC algorithm consists of three phases. They are initialization, employed, onlooker and scout bees phase. Each phase is replayed until the maximum count of iterations is reached. In the initial phase, the count of solutions and the control parameters are fixed. The employed bees phase deals with the search of new high quality food sources in the nearby locality of old food source. The new food source is then evaluated for its fitness, which is then followed by the comparison of the old and the new food source by means of greedy selection. The collected knowledge about the food source is distributed among the onlooker bees present in the beehive. In the next phase, the onlooker bees follow a probabilistic approach to select the food sources with respect to the information provided by the employed bees. This is followed by the calculation of the fitness function of the food source, which is located nearby the selected food source. Finally, the old and the new food sources are compared by the greedy selection. In the final phase, the employed bees turn to scout bees, when their solutions cannot be enhanced within a predefined count of iterations. The solutions so found by the bees are dropped out. At this point, the scout bees search for new food source again. Using this functionality, the poor solutions are dropped out. These three phases continue its process until the stopping point is reached [Karaboga et al, 2012]; [Karaboga and Ozturk, 2011].
Figure 1: Pseudo-code of artificial bee colony algorithm.

```plaintext
1: Input: Training data;
2: Produce initial population $i = 1$ to $MC$
3: Calculate the fitness function of the population
4: Fix counter=1
5: Do

// Employed bees phase
6: Search for the food source;
7: Calculate the fitness function;
8: Employ greedy selection process;
9: Compute the probability for the food source;

// Onlooker bees phase
10: Select food source based on the probability values;
11: Generate new food source;
12: Calculate the fitness function;
13: Apply greedy selection process;

// Scout bees phase
14: If food source drops out then swap it with new food source;
15: Save the best food source;
16: Counter +=1;
17: While counter=MC;
```

![Image](image.png)

**Initialization phase**

The colony of artificial bees includes three groups of bees, employed bees, onlooker bees and scout bees. The food source vectors are initialized by scout bees. The initialization formula is shown as follow,

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

(1)

where $x_{mi}$ is a solution vector to be optimization problem, $l_i$ and $u_i$ are the lower and upper bound of the parameter, respectively, and $i \in 1 \cdots n$ with $n$ the dimension of the problem to solve.
Employed bees phase

Employed bees search for new food source with more nectar within the neighborhood of the food source \( x_{mi} \) in their memory. They evaluate the profitability (fitness) after they find a neighbor food source. Next equation shows a way to determine the neighbor food source.

\[
v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki})
\]  

(2)

where \( x_k \) is a random chosen food source, \( i \) is a random parameter index and \( \phi_{mi} \) is a random number within the range \([-a, a]\). Thereafter, the fitness is calculated and the formula is as below,

\[
\text{fit}(x_m) = \begin{cases} 
\frac{1}{1 + f_m(x_m)} & \text{if } f_m(x_m) > 0 \\
1 + \text{abs}(f_m(x_m)) & \text{if } f_m(x_m) < 0
\end{cases}
\]  

(3)

where \( f_m(x_m) \) is the objective function value of solution \( x_m \).

Onlooker bees phase

Employed bees share their food source information with onlooker bees in the hive by dancing, and then onlooker bees choose the food sources by the probability. The probability is calculated by fitness values provided by employed bees. The probability value \( p_m \) with which \( x_m \) selected by an onlooker bee could be calculated by equation

\[
p_m = \frac{f_m(x_m)}{\sum_{i=1}^{SN} f_i(x_i)}
\]  

(4)

After a food source \( x_m \) for an onlooker bee is probabilistically chosen, a neighborhood source \( v_{mi} \) is determined by using next equation, and its fitness value is computed. Hence, more onlookers are recruited to richer sources and positive feedback behavior appears.

\[
v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki})
\]  

(5)

Scout bees phase

The unemployed bees who choose their food source randomly are called scouts. Employed bees whose solutions cannot be improved through a predetermined number of trials, specified by the user of the ABC algorithm and called “limit” or “abandonment criteria” herein, become scouts and their solutions are
abandoned. Then the scouts start to search for new food source randomly. For instance, if \( _X^m \) solution has been abandoned, the new solution discovered by the scout who was the employed bee of \( _X^m \) can be defined by (1). Hence those sources which are initially poor or have been made poor by exploitation are abandoned and negative feedback behavior arises to balance the positive feedback.

The typical application of ABC algorithm is to solve the optimization problem. In general, the problem has to be converted to the problem to find optimal parameter vector that minimize the predefined objective function. Then the artificial bees search for the best solution by an iterative strategy: moving towards better solutions by means of neighbor search mechanism while abandoning poor solutions.

Figure 2 (a, b) shows preliminary obtained results of Rastrigin function using Scatter of Bees tool (http://mf.erciyes.edu.tr/abc/software.htm), each execution with the same colony size and the same number of cycles as random values are generated based on the range.

![Figure 2a](image-url)
Figure 2b

Figure 2 (a, b): Optimization of Rastrigin function using artificial bee colony algorithm. Note that obtained results are different using the same colony size and number of iterations, due to the domain range, bee distribution, food source (fitness function).

**Experimental Results**

Simulations are performed based on the ABC Scatter tool, see Figure 2, using Rastrigin and Sphere models. Initial parameters such as the size of the bee colony, the number of cycles, the number of iterations can be seen on Table 1.

<table>
<thead>
<tr>
<th>Colony size</th>
<th>50, 100, 500, 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain range</td>
<td>[-5; 5] Rastrigin, [-100; 100] Sphere</td>
</tr>
<tr>
<td># Iterations</td>
<td>200</td>
</tr>
<tr>
<td>Limit</td>
<td>20</td>
</tr>
<tr>
<td># runs</td>
<td>1</td>
</tr>
</tbody>
</table>

It can be verified that as the size of the colony increases, the number of exploratory bees also increases and the average performance is improved in each cycle (see Figure 3 (a, b, c, d) ).
Figure 3a.

Figure 3b.
Figure 3c.

Figure 3d.

Figure 3: Optimization results of Rastrigrin function
(using different colony size: a) 50; b) 100; c) 500; d) 1000).
Figure 4a.

Figure 4b.
Figure 4c.

Figure 4d.

Figure 4 (a, b, c, d): Optimization results of Sphere function (using different colony size a) 50; b) 100; c) 500; d) 1000).
One of the characteristics of using the sphere model is that the bees do not disperse when looking for the food source, but all of them converge from the outside to the inside to the point 0:0 as shown in Figures 3 (a, b, c, d) and 4 (a, b, c, d). In addition, with respect to Rastrigin, the value of exploratory bees with a colony size of 50 differs, since from the cycle between 50 and 100 the first explorers are visualized, and another of the differences is that the objective function for Sphere case is less than 2.5, so the average performance is better.

Within the software engineering field, several approaches, techniques and tools have been designed and developed to support the recovery of the architecture and the remodularization of legacy software systems, including clustering, which is a technique that provides a high level of abstraction of software architectures by dividing a system into significant subsystems and forming groups of items or entities (clusters) so that there is similarity between entities [Torres et al, 2009].

The similarity between entities is determined based on their characteristics or behavior and supported by algorithms, one of which is called k-means [Armano and Farmani, 2014]. Using the K-means algorithm [Scanniello et al, 2010], software entities are classified as groups, this algorithm defines a centroid for each cluster to identify and iteratively refine the centroid by minimizing the average distance/similarity of the software entities to their centroids (see Figure 5a)).

In fact, to identify the best partition of the clustered software entities, the algorithm is run iteratively using different configurations and in addition to that metrics such as MoJo are used which is based on a heuristic approach that approximates the exact measurement values to find similarity between classes. The pseudo-code proposed by [Alam and Baulkani, 2016] to combine the ABC algorithm with K-means is presented in Figure 5b.
inputs:
    D := \{d_1, \ldots, d_n, d_1, \ldots, d_n\} // the dissimilarity matrix
    p // number of clusters
outputs:
    C := \{c_1, \ldots, c_p\} // cluster centroids
    m : I \rightarrow C // cluster membership

procedure k-means
    set I := \{d_1, \ldots, d_n\}
    set C := \{c_1, \ldots, c_p\} to initial value // rnd. selection of d_1,..., d_n
    for each d_j \in I
        m(i_j) := \arg\min_{s \in \{1..p\}} D(d_j, c_s)
    end
    while m has changed
        for each j \in \{1..p\}
            c_j := average of d_i whose m(i) = j
        end
        for each d_j \in I
            m(i_j) := \arg\min_{s \in \{1..p\}} D(i_j, c_s)
        end
    end
    return C, m
end

Figure 5a: K-means algorithm
Initialize the algorithm parameters
Pre-process the document
Randomly assign the population of food sources
Determine the fitness of the population by (6)
While
   For each employed bee
      Produce new food source;
      Calculate fitness of the food source;
      Employ K-means and greedy selection;
   . Calculate the probability of food source by (8);
   . For each onlooker bee
      . Choose the food source with respect to step

   . Produce new food source and compute its fitness by (6);
   . Apply K-means and greedy selection
      . Compare and swap the solutions if new source is better;
   . Save the best food source;
. end while (termination condition not met);

Figure 5b: ABC K-means algorithm
Conclusion and further work

A simulation, that supports the identification of similarity between different types of applications built under the object-oriented programming paradigm where the extraction and identification of classes, methods and their similarity will support architectural reconstruction or recovery, can be implemented using proposed research in this paper, based on combining the artificial bee colony algorithm with K-means.

Taking into account the capabilities of the ABC algorithm, this algorithm could be used to solve numerical problems under uncertainty conditions. A future research work that could be done consists on analyzing the similarity between classes using genetic algorithms, ABC and ABC k-means.

The techniques can be extended to various real-world problems such as classification and clustering of malware, email analysis (finding social graph among the users based on email contents, for instance) in digital forensics. Since unsupervised clustering algorithms do not give accuracy; the proposed algorithm can be applied to find a clustering algorithm for many real-life applications where clustering techniques are applied. The approach should enable users to experimentally compare various clustering algorithms and choose the one that best serves the problem.

Acknowledgements

This research was partially supported by:

- Spanish project Integración de Sistemas Cooperativos para Vehículos Autónomos en Tráfico Compartido (CAV) TRA2016-78886-C3-3-R.

Bibliography


Authors’ Information

Nuria Gómez Blas - Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: nuria.gomez.blas@upm.es

Major Fields of Scientific Research: Bio-inspired Algorithms, Natural Computing

Luis Fernando de Mingo López - Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: fernando.demingo@upm.es

Major Fields of Scientific Research: Artificial Intelligence, Social Intelligence

Juan Castellanos Peñuela - Departamento de Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn, 28660 Madrid, Spain; e-mail: jcastellanos@fi.upm.es

Major Fields of Scientific Research: Formal Languages, Neural Networks