Artificial Bee Colony (ABC) algorithm and Its Implementation in Clustering

Yangzhuoran (Fin) Yang (28150295)

1 Introduction

Artifical Bee Colony (ABC) algorithm is a meta-heuristic optimization algorithm recently introduced by Karaboga (2005). It simulate the behaviour of a honey bee swarm in the attempt to find the optimal solution. As an general optimization algorithm, it does not limit to Clustering problem. We now introduce each component seperately the give the ABC algorithm in the pseudo-code form. Part of the notation and formulations are adopted from Dervis Karaboga and Ozturk (2011).

1.1 Overview

Other than parameter initialization and solution evaluation, the ABC algorithm can be structured into three phases: the employed bee phase, the onlooker bee phase, and the scout bee phase. Each phase mimic the behavior of a group of bees in a honey bee swarm. The employed bee and the onlooker bee search locally while the scout bee is in charge of the global search. In other words, the employed bee and the onlooker bee emphasis intensification by producing better solutions based on the current solution set, while the scout bee emphasis diversification search solutions independently from the current set of solutions.

1.2 Initialization

To mimic the behavior of a bee swarm, the ABC algorithm needs parameter that defines the size of the swarm: the numer of food sources, or the number of solutions in the solution set. We denote this number as SN (swarm size). The swarm size is one of the most important parameter in the ABC algorithm, as a large swarm size increases the accuarcy and decrease efficiency. We will dicuss the impact of swarm size in more detail in the parameter section.

After SN being decided, the ABC algorithm will simulate the position of initial food sources (the set of solutions) $z_i : i = 1, 2, \ldots, SN$. The way to simulate the food sources has been tailored in differnt problems in the literature: they can be evenly assigned across the solution space (Vega Yon and Muñoz (2017)), randomly generated from a distribution (Dervis Karaboga and Ozturk (2011)), or they can be randomly selected from different data points for the problem of clustering. The main idea is to cover the solution space as much as possible.

Once the position of the inition solutions has been determined, the fitness $f_i : i = 1, 2, \ldots, SN$ can be calculated from corresponding cost function/objuective function. The quality of the nectar $fit_i : i = 1, 2, \ldots, SN$ in the ABC algorithm can be calculated correspondly, using Equation (1)

$$fit_i = \frac{1}{1/f_i} \tag{1}$$

In the case when the cost function produces negative fitness, the quality of the nectar can be calculated by:

$$fit_i = 1 + |f_i| \tag{2}$$

The decision on the nectar calculation largerly depends on the problem to solve. The solution with best fitness and will be recorded. We can now proceed to main component of the iteration, with the first phase: the employed bee phase.

1.3 The Employed Bee

The number of employed bees is the same as the number of food sources SN. For each employed bee at each food source, the bee implements a neighbourhood search to find a new solution by combine the neighbour find with the current position using

$$\nu_{ij} = z_{ij} + \phi_{ij}(z_{ij} - z_{kj}) \tag{3}$$

where $i \in \{1, 2, \ldots, SN\}$ is the index of the current position, and $j \in \{1, 2, \ldots, SN\}$ is the randomly generated index of the neighbour. If we denote D as the number of elements we need to optimize in one solution (the number of dimensions), then $k \in \{1, 2, \ldots, D\}$ is the randomly generated index denoting the position of the element. phi_{ij} is a random number simulated using a uniform distribution with bound -1 and 1.

After finding the new solution, the employed make a decision on whether to jump from the current solution to the new solution by compare the quality of the nectar of two positions. If the bee decides to jump to the new postion with the higher value of fit_i, it will forget the old position, i.e. the old solution was not stored in the memory of the algorithm.

1.4 The Onlooker Bee

The onlooker bee performs the same local search as the employed bee. The difference is the onlooker bee does not implement the search on each and every food source, but selectively perform the search based on the quality of the nectar of each food source. After the employed bee phase, the algorithm calculate the probabilities p_i of the onlooker bee selecting each food source based on the following equation:

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{4}$$

Different schemes can be used to calculate the probability, depending on the problem to solve. Vega Yon and Muñoz (2017) uses the equation bellow

$$p_i = a \times \frac{fit_i}{\max_i(fit_i)} + b \tag{5}$$

with $a = 0.9$ and $b = 0.1$. Probability values are calculated by using the quality of nectar fit_i and normalized by dividing maximum fit_i.

The number of onlooker bees is the same as the employed bee or the number of food sources SN. Each onlooker bees selects a solution as the base solution, finds a new solution like the employed bee using equation (3), and choose whether to switch solution using the same greedy approach.

1.5 The Scout Bee

The last two phases focus on the local search of solution. The scout bee component prevent the algorithm from stucked in a local optim. After the employed bee and the onlooker bee phase, whether the food source has been moved by one of the bees has been recorded. If a food source has not been improved (moved) up to some some certain number of iteration, called the limit, it is abandoned and a new solution would be found in the scout bee phase. It ransomly generate a solution in the solution space using the following equation

$$z_i^j = z_{min}^j + \delta_i^j(z_{max}^j - z_{min}^j) \tag{6}$$

where z_i is the abandoned source and $j \in 1, 2, \ldots, D$ is the index of dimension. δ_i^j is simple generated from a uniform distribution with bound 0 and 1. The replaced food source can be selected using a differnt method in different problems. The main idea is to expand the range of selecting new solution.

1.6 Summary

The ABC algorithm can be summarized in the following pseudo-code.

```
Algorithm: Artifical Bee Colony
 1. Load the training data
 2. Generate the initial food sources 1,2,...,SN
 3. Evaluate the quality of nectar (the fitness of initial solutions)
 4. While (Condition not met)
        The employed bee phase
 5.     For each employed bee{
          Produce new solution using neighbourhood search
          Calculate the fitness
          Selecte the better fitted solution Greedily }
 6.     Calculate the probabilities of selecting each solution
        The onlooker bee phase
 7.     For each onlooker bee{
          Select a solution based on the probabiliy calculated above
          Produce new solution using neighbourhood search
          Calculate the fitness
          Selecte the better fitted solution Greedily }
 8.     Abandon the solution that the number of unimproved iteration reach the limit
        The scout bee phase
 9.       Increase the number of food source to SN by finding new solution randomly
10.     Record the best solution among all food sources
11. End
```

The updated solution in each iteartion is the best solution among all the food sources in that iteration, and the final solution chosen by the algorithm is the best solution among all the food sources tried in the past up until the time the condition is met. Stopping condition are required to determine when to stop the algorithm. Dervis Karaboga and Ozturk (2011) choose to use the maximum number of iteration as the stopping condition. Vega Yon and Muñoz (2017) includes the number of umimproved iteration: the algorithm stops when the result does not improve up to a certain number.

The ABC algorithm aims to chieve a balance between intensification and diversification through three different phases mimicing three different types of bee.

1.7 The parameters

There are three parameeters in the ABC determines the quality of optimization and needs to be taken carefully: 1. the number of food sources (SN) which is also the number of empleyed bees or the onlooker bees; 2. the value of limit that decides when to abandon food sources; 3. the stopping conditions, such as the maximum cycle number (MCN) or the maximum number of unchanged iteration.

1.7.1 The number of food sources (SN)

A large numer of food sources can increase the excution time for each iteration dramatically. It is the number of neighbourhood serach and greedy evaluation conducted in the employed bee phase or the onlooker bee phase. For each iteration, there is effectively $2 \times SN$ neighbourhood search conducted by the employed bee and the onlooker bee. While a large SN increase the time it needs to run for each iteration, it also means the local search is more throughly implemented: the number of solution we are consider at the same time is large. Therefore, there is a higher chance of finding a better solution.

1.7.2 The limit

The limit controls the balance between intensification and diversification. If the limit is set to be too large, a useless food source needs more time to be droped and more time is needed to triger the global search

implemented by the scout bee, which may leads to unnecessary intensification over diversification. If the limit is set to be too small, a promising food source may be droped before it runs the time it needs to produce a good solition, and the scout bee is trigered too early, leading to diversification more then intensification.

1.7.3 The stopping condition

If we consider the maximum cycle number (MCN) as the stopping condition, a small value of the MCN may stops the algorithm too early then the optimization settles down to the optimal solution. If MCN is set to be too large, the algorithm may run many extral iterations without any improvement, leading to a poor efficiency. The maximum number of unchanged iteration works the same way: the algorithm may stops before the solution converges.

1.8 Comparing to others and its own features

We compare the ABC algorithm to two other meta-heurtistics: The Simulated Annealing and the Genetic Algorithm. The ABC algorithm share some of the same features of other meta-heuristics while having its own technic to some of other problem. We now look at how the problem specific elements in those algorithms are presented in the ABC algorithm.

1.8.1 Escape local optimum differently from Simulated Annealing

The simulated annealing differs from the basic neighbourhood by allowing the algorithm to accept a solution with worse fitness with certain probabilities. With such a feature that mimic the process of ammealing metals to climb the "hill" in the curve of objective function, it can escape the local minimum (when it is a minimization problem). The probability of accepting a worse move decreases with time.

The ABC algorithm does not have the feature to accept a worse move if we consider the path of solution in ABC as the best solution among all the food soures in each iteration. However, it has a similar functionality to avoid being stucked in the local minimum: it abandons the food source that has not been improved for some certain amount of iteration. By droping unimproved food source, it effectively drops the solutions that are in the local minimum. By consider the all the food sources collectively and add in new food sources from the whole space, it expands its search spane.

1.8.2 Reproduce like Genetic Algorithm (GA)

The genetic algorithm imitates the process of evolution. It selects multiple solutions (selection), create a new solution from those existing solutions (crossover), modify the solution to create another new solution (mutation), calculate the objective value of the new solution (evaluation) and determine whether to replace a existing solution using the new solution (update).

Many components in the ABC algorithm largely share the same process in the genetic algorithm. The bees finding neighbours is corresping to selction in GA. Creating new solution using the neighbour is corresping to the crossover in GA. calculating the fitness of the new solution and deciding whether to take the new solution are corresping to evaluation and update. The differece is for the step of mutation. In GA, mutation is done separatly from crossover, but in ABC the mutation step is embeded in the crossover: after selecting a neighbour, the bee create the new solution not only based on the esisting ones, but also adding a random component, the random number generation in Equation (3).

1.8.3 Its own features

1.8.3.1 Additional global search

The scout bee phase in the ANC provides addtional global search that is lacked in both the simulated annealing and the genetic algorithm. Although the diversification can be achieved using different parameters in those two methods, using a sepeartly component generats certain exposure to the whole solution space.

1.8.3.2 Tendency towards better performed solution

The onlooker bees find the food sources with better quality and conduct seach aroud them: the create new solutions using good solutions. This improves the effeciency of the algorithm as intuitively good solutions may cluster together. Simiar feature can be found in come modified genetic algorithm. The way to select the parent solutions may be weighted towards good solutions.

References

Karaboga, D (2005). "An idea based on honey bee swarm for numerical optimization". In: URL: https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf.

Karaboga, Dervis and Celal Ozturk (2011). "A novel clustering approach: Artificial Bee Colony (ABC) algorithm". In: *Applied soft computing* 11.1, pp. 652–657.

Vega Yon, George and Enyelbert Muñoz (2017). *ABCoptim: An implementation of the Artificial Bee Colony (ABC) Algorithm*. R package version 0.15.0. URL: https://github.com/gvegayon/ABCoptim.