Artificial Bee Colony Algorithm Based on Adaptive Strategy

Lu Cunkui

***Abstract*— The adaptive factor mechanism is proposed to improve the solution search equation of artificial bee colony algorithm, and the optimal neighbor food source is selected from the current ring neighborhood topology of the food source for mining, so as to balance the exploration and mining capabilities of the algorithm. In addition, in order to jump out of the local optimal solution in the search process of reconnaissance bees, a cross-mutation strategy is proposed to generate the reverse solution of the abandoned food source, which improves the search efficiency of the algorithm. Finally, the test function is used to verify that the performance of the improved algorithm is significantly improved**

***Index Terms*— ABC, self-adaptation, cross variation, global optimization.**

I. INTRODUCTION

The artificial bee colony (ABC) algorithm is a novel global optimization algorithm proposed in recent years, which has the characteristics of simple structure, few control parameters, and easy implementation[1]. The algorithm is similar to the particle swarm optimization (PSO) algorithm, which is designed by simulating the swarm intelligence behavior of natural organisms[2]. However, unlike the PSO algorithm, ABC simulates the intelligent honey collection behavior of the bee colony, which maximizes the honey collection of the entire bee colony according to the division of labor and cooperation of different types of bees, that is, finds the optimal solution to the problem to be optimized.At present, the ABC algorithm has been successfully used to solve many different types of practical optimization problems, such as parameter optimization, symbolic regression, neural networks, and vehicle routing problems[3]. In the standard ABC algorithm, employed bees and looker bees share a solution search equation to generate new candidate food sources[4].In order to improve the mining ability of the ABC algorithm and not easily fall into local optima, inspired by the local version of the local particle swarm optimization (LPSO) algorithm[5], this paper particle swarm optimization (LPSO) algorithm, this paper proposes an adaptive artificial bee colony algorithm algorithm. In the observation bee stage, the neighborhood search mechanism is used to select an optimal food source from the ring neighborhood topology of the current food source, and search for new candidate food sources within the neighborhood of the food source, which helps to enhance the mining ability of the algorithm and is not easy to fall into local optima[6]. In addition, in order to help scout bees save the search experience, a cross-mutation strategy is proposed to generate the reverse solution of the abandoned food source, so that scout bees can quickly find new excellent food sources[7]. In order to verify the performance of the algorithm in this paper, it is compared with three different types of improved algorithms[8].

Manuscript received March 15, 2024
Lu CunKui, Tiangong University

II. IMPROVED ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony algorithm (ABC), inspired by the natural behavior of bees, is a heuristic optimization algorithm[9]. It simulates three roles within a bee colony: employed bees, scout bees, and onlooker bees. In the context of cloud task scheduling, the optimal nectar source corresponds to the best task allocation method. Each nectar source represents a solution, with each solution mapping to a position in the search space of the problem. Bees collaborate by tracking the current best solution and collectively work towards finding better solutions through the objective function[10].

The fundamental principle of the ABC algorithm involves continuously tracking individual and global optimal solutions to explore the solution space. Employed bees conduct local searches based on the current solution at their position, updating their solutions to more optimal ones by evaluating the objective function. Scout bees search for new solutions within a specified range to discover improved alternatives. Onlooker bees select solutions with higher fitness by observing the solutions of other employed bees for replication. In each iteration, the ABC algorithm selects bees based on their fitness values. Bees with higher fitness have a greater probability of being chosen as scout or onlooker bees. Scout bees randomly select a position within the search range for exploration and update their solutions[11]. Onlooker bees choose a better-employed bee and replicate and update its solution. This way, the bee colony collaborates and competes in the problem's search space, gradually optimizing solutions. The strengths of the ABC algorithm lie in its robust global search capabilities and fast convergence. It is problem-agnostic, applicable to various optimization problems, and characterized by few parameters and simple implementation. However, challenges such as slow convergence and susceptibility to local optima exist. Therefore, this paper addresses these issues by improving search and selection strategies, introducing adaptive functions, and incorporating crossover mutation operators to overcome early convergence issues and late-stage local optima problems in the algorithm.

1. Generating Initial Nectar Sources

The bee colony operates without prior knowledge, and during the initialization phase, all bees are scout bees. Let the population of bees be denoted as N, where the number of employed bees and nectar sources is also N. The optimal nectar source obtained through the objective function represents the optimal solution for this task scheduling. The cloud computing task scheduling problem is a D-dimensional scheduling problem, where the position of the i-th nectar source, denoted as X_i, can be represented as $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ (i = 1, 2, 3, ..., N). The formula for generating initial feasible solutions is:

$$x_{ij} = x_{\min j} + (\text{int})rand(0,1)(\mathrm{x}_{\max j} - \mathrm{x}_{\min j}) \quad (2.1)$$

Where $x_{\min j}$ and $x_{\max j}$ are the boundary values of the nectar source position. According to the cloud computing task scheduling model, $[x_{\min j}, x_{\max j}]$ equals the respective IDs of the virtual machines, i.e., [0, number(VM)-1]. rand(0,1) represents a random number between 0 and 1, and int denotes rounding to the nearest integer.

2. Introducing an Adaptive Operator

In the original algorithm, the selection phase of the observer bees used a roulette wheel algorithm, which is simple in principle and easy to implement. This algorithm tends to select elite individuals with a higher probability, thus preserving excellent individuals during the evolutionary process. However, in the selection process, individuals with lower fitness have a smaller selection probability, which may lead to a decrease in diversity. The performance of the roulette wheel algorithm is strongly influenced by the fitness function. When the fitness function is designed poorly or inaccurately, the performance of the algorithm may be affected. Additionally, the roulette wheel algorithm tends to favor individuals with higher fitness and is difficult for individuals with lower fitness to obtain selection opportunities, which can easily lead the population to fall into local optima. The roulette wheel algorithm has no memory, and each individual selected is independent, unable to utilize past selection experience information.

To address these issues, this paper introduces an adaptive operator in the observation bee selection and following stage. Specifically, let the adaptive operator be sa. When an individual in the population is a global optimal solution, the attractor SA is determined according to formula 2.2. Otherwise, the attractor sa is determined according to formula 2.3.

$$\mathrm{sa} = \mathrm{x}_{\text{best}} * \mathrm{c}_1 + \mathrm{c}_2 \quad (2.2)$$

$$\mathrm{sa} = \mathrm{x}_i + (\mathrm{x}_{\text{best}} - \mathrm{x}_i) * \mathrm{c}_3 \quad (2.3)$$

Where x_{best} represents the optimal solution, i.e., the best nectar source, and x_i denotes the feasible solution of the i-th nectar source. c1, c2, and c3 represent a set of random numbers, where c1 in (0.5, 1.5), c2 in (-1, 1), and c3 in (-1, 1). Each onlooker bee converges towards the adaptive operator, ensuring that individuals searching within their neighborhood do not exceed boundary values. Utilizing the adaptive operator enhances the algorithm's search capability, accelerating early convergence, and reinforcing local search capabilities. Based on the adaptive factor, the formula for updating the position in the follow phase for onlooker bees can be expressed as:

$$\begin{cases} x_j^{'} = (x_j - sa_j)\%((\mathrm{sa}_j - \mathrm{x}_{\min}) * \mathrm{c}) / \mathrm{c} + sa_j, x_j < sa_j \\ x_j^{'} = (x_j - sa_j)\%((\mathrm{x}_{\max} - \mathrm{sa}_j) * \mathrm{c}) / \mathrm{c} + sa_j, x_j > sa_j \end{cases} \quad (2.4)$$

Where % is the modulo operator, c represents the algorithm's scaling factor, typically not exceeding 1.5. Here, x_j represents the current position, sa_j represents the adaptive operator corresponding to the current individual, x_{min} denotes the minimum boundary, and x_{max} represents the maximum boundary. The algorithm updates the position based on the adaptive algorithm with each iteration.

Therefore, the search equation for employed bees can be modified to:

$$x_{ij}^{'} = x_{\mathrm{ij}} + \phi(\mathrm{x}_{\mathrm{ij}} - x_{\mathrm{kj}}) + \alpha(\mathrm{x}_{\text{best}} - x_{\mathrm{ij}}) \quad (2.5)$$

Where x_{ij} represents the current position, phi denotes a random number between [-1,1], and x_{best} is the current global optimal solution. The addition of the guidance of the optimal solution to the search equation enhances the algorithm's search capability and accelerates its convergence rate.

3. Cross mutation operator is introduced

In the artificial bee colony algorithm, two parameters are set, which are the give up counter corresponding to each honey source, which is used to record the number of times that the honey source is not updated, and the maximum threshold, which is denoted as C and LIMIT respectively. When observing bees choose a new location for following, they will make greedy comparison. If the fitness of the new location is greater than the fitness of the original location, C_i will remain unchanged; otherwise, C_i will increase, indicating that the honey source in this search is eliminated. In the detection peak phase, the threshold is compared with the relative abandon counter, and if C_i<LIMIT, the global search is not re-performed. If C_i≥LIMIT, then the hired bees attached to the honey source will be transformed into scout bees, and L will be set to 0, and the global search will be conducted again. The original working mode of the scout bee phase is:

table I

Scout bee phase pseudo code

```
t=1;
  While(t= number of nectar sources)
    IF (C(i)≥LIMIT)
    Performs the global search location update formula
    t=t+1;
    ELSE
    t=t+1;
  END
END
```

The traditional method only compares the abandonment counter with the threshold value in the reconnaissance phase, and does not do any processing in the middle, which may cause the algorithm to fall into local optimal. The approach of this paper is to cross and mutate individuals when the abandoned counter reaches half of the threshold to increase the randomness of the algorithm and improve the global search ability of the algorithm. The specific working mode is:

Improved post-scout bee phase pseudo code

```
t=1;
  While(t= number of nectar sources)
    IF (C(i)≥LIMIT/2)
    Cross variation
    Fitness judgment
    t=t+1;
  END
  IF(C(i)≥LIMIT)
    Performs the global search location update formula
```

```
        t=t+1;
    END
END
```

4. Improved algorithm flow

The algorithm steps in this paper are as follows:

Step 1: Initialize parameters, initialize population size, number of nectar sources, discard counter, maximum threshold, boundary value, number of iterations, and random location of the initial nectar source.

Step 2: In the hiring bee stage, the hiring bee searches in the neighborhood according to the global optimal solution, calculates the fitness, makes greedy comparison, and selects a new honey source.

Step 3: Observe the bee stage, introduce adaptive operators into the position update formula of the bee stage, and gradually bring the individual closer to the excellent honey source. Calculate fitness, make greedy comparison, select new honey source.

Step 4: In the reconnaissance peak stage, the nectar source is traversed to determine the value of the abandoning counter corresponding to each nectar source. If the abandoning counter value exceeds half of the limiting threshold, the honeysource is cross-mutated. If the abandoning counter value exceeds the limiting threshold, the observation bees are transformed into reconnaissance bees, and a new honeysource is selected for exploration through global random search.

Step 5: If the algorithm reaches the maximum number of iterations, the optimal solution in the current honey source is returned. The algorithm ends.

The flow chart of the improved algorithm is as follows:

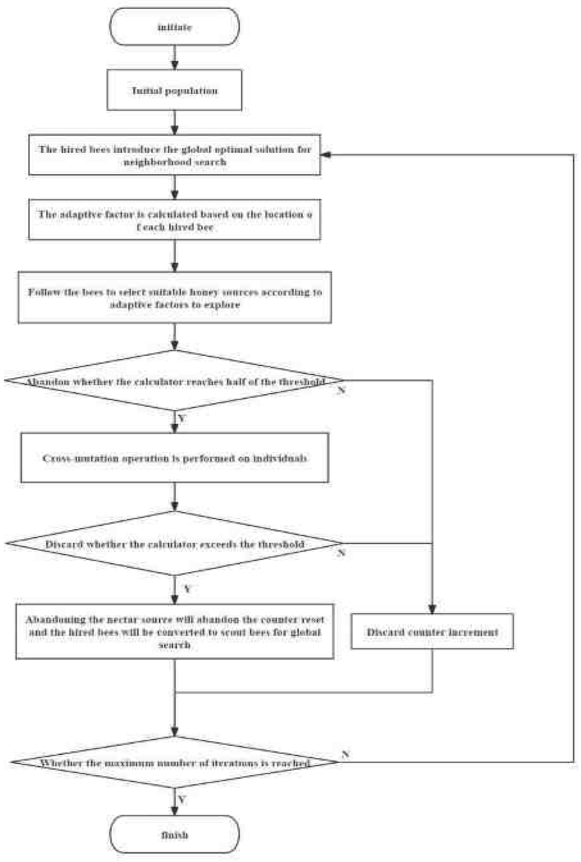
initiate
Initial population
The hired bees introduce the global optimal solution for neighborhood search
The adaptive factor is calculated based on the location of each hired bee
Follow the bees to select suitable honey sources according to adaptive factors to explore
Abandon whether the calculator reaches half of the threshold
N
Y
Cross-mutation operation is performed on individuals
Discard whether the calculator exceeds the threshold
N
Y
Abandoning the nectar source will abandon the counter reset and the hired bees will be converted to scout bees for global search
Discard counter increment
Whether the maximum number of iterations is reached
N
Y
finish

Figure. 1 Algorithm flow chart

III. Simulation results and analysis

In order to verify the performance of the algorithm, the improved algorithm is compared with the standard ABC algorithm, the gray wolf algorithm (GWO) and the genetic algo (GA) using the matlab simulation tool。

The fitness function of the algorithm is shown in Figure 3.1. Since the fitness function is calculated according to the value of the objective function, the objective function minimizes the task completion time, cost spent and load degree to the target value, the smaller the fitness function value obtained, the better the performance of the algorithm. As can be seen from Figure 3.1: The traditional artificial bee colony algorithm convergence speed is slow, and the algorithm is easy to fall into the local optimal situation in the later period, but the improved artificial bee colony algorithm convergence speed is also accelerated with the enhancement of search ability after the introduction of adaptive function in the bee observation stage, and the local optimal solution can be jumped out according to the cross-mutation operator, thus enhancing the global search ability of the algorithm.

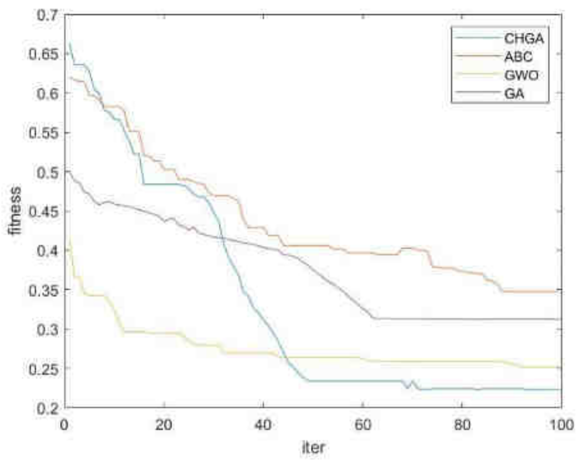

Figure 3.1 Comparison diagram of fitness function

It is clear from the figure that the improved algorithm performs better than other optimization algorithms.In the experimental results, the adaptability of the CHGA algorithm is obviously better than other algorithms. The traditional artificial bee colony algorithm has a slower convergence speed, and the algorithm is prone to fall into the local optimum in the later stage. However, the improved artificial bee colony algorithm introduces an adaptive function in the following bee stage. With the enhancement of the search ability, the convergence speed of the algorithm is also accelerated, and the cross-mutation operator can jump out of the local optimal solution, thereby enhancing the global search ability of the algorithm.

CONCLUSION

In order to improve the mining ability of the ABC algorithm, an adaptive mechanism is proposed to improve the solution search equation. In addition, in order to expand the excellence of the group with individuals, a cross-mutation strategy is proposed to generate new individuals.The improved algorithm is tested using the Matlab simulation platform, and the experimental results are compared with three well-known improved algorithms. The results show that the strategy in this paper can effectively improve the performance of the ABC algorithm, and has great advantages in terms of convergence speed and accuracy.The focus of the next research work is to apply the algorithm in this paper to solve practical optimization problems, such as component deployment problems in distributed software systems.

REFERENCE

[1] Mell P, Grance T. The NIST definition of cloud computing [J]. National Institute of Standards and Technology, 2011.

[2] Bala R, Gill B, Smith D, et al. Magic quadrant for cloud infrastructure and platform services [J]. Gartner, 2021.

[3] Ibrahim I M. Task scheduling algorithms in cloud computing: A review [J]. Turkish Journal of Computer Mathematics Education(TURCOMAT), 2021, 12(4): 1041-1053.

[4] Chen W, Deelman E. Workflowsim: A toolkit for simulating scientific workflows in distributed environments[C]// 2012 IEEE 8th international conference on E-science:IEEE,2012: 1-8

[5] Peng J-j, Zhi X-f, Xie X-l. Application type based resource allocation strategy in cloud environment [J]. Microprocessors Microsystems, 2016, 47: 385-391.

[6] Wu Z, Xiong J. A novel task-scheduling algorithm of cloud computing based on particle swarm optimization [J]. International Journal of Gaming Computer-Mediated Simulations, 2021, 13(2): 1-15.

[7] Huang X, Li C, Chen H, et al. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies [J]. Cluster Computing, 2020, 23: 1137-1147.

[8] Garg S K, Buyya R. Networkcloudsim: Modelling parallel applications in cloud simulations[C]// 2011 Fourth IEEE International Conference on Utility and Cloud Computing:IEEE,2011: 105-113.

[9] Al-dilami R A, Zahary A T, Al-Saqqaf A Z. Enhancing the Distribution of Idle Cost for Scheduling Tasks without Setup Cost in Cloud Computing [J]. Mathematical Problems in Engineering, 2021, 2021: 1-13.

[10] Chafi S-E, Balboul Y, Mazer S, et al. Cloud computing services, models and simulation tools [J]. International Journal of Cloud Computing, 2021, 10(5-6): 533-547.

[11] Mubeen A, Ibrahim M, Bibi N, et al. Alts: An adaptive load balanced task scheduling approach for cloud computing [J]. Processes, 2021, 9(9): 1514.