Research on cloud computing resource scheduling strategy based on genetic ant colony fusion algorithm

Jun Nie*
Computer College, Guangdong University of Science and Technology, Dongguan 523083, Guangdong, China

ABSTRACT

In this paper, ant colony algorithm and genetic algorithm are combined to improve the stability and efficiency of cloud computing resource scheduling. The strategy of this algorithm is to use the local optimal solution of genetic algorithm as the initial pheromone of ant colony algorithm, and introduce load balancing adjustment factor and transfer probability into ant colony algorithm. The experimental results show that the genetic ant colony fusion algorithm in this paper has better advantages in the number of iterations, time cost, power cost and load balancing, and achieves the goal of uniform distribution of cloud computing resources.

Keywords: Cloud computing, resource scheduling, fusion algorithm

1. INTRODUCTION

Cloud computing resource allocation can be explained as a process of reasonably allocating resources for different customers in a characteristic cloud environment, following some resource allocation rules[1]. With the increasing maturity of cloud computing technology, the allocation of cloud computing resources has become the focus of attention. At present, the related research on resource allocation strategy has achieved good results. For example, Reference[2] optimized the task scheduling and evaluated each task, but the results deviated greatly due to the different adaptation functions of genetic algorithms. The algorithm proposed in Reference[3] will cause search stagnation due to the lack of initial paths. In this paper, the two algorithms are fused together to give full play to their respective advantages. The experimental results show that the genetic ant colony fusion algorithm is very effective for resource allocation.

2. GENETIC ALGORITHM

Genetic algorithm originated in the early 1960s. It is a computational model based on the genetic and evolutionary mechanism of nature[4]. It searches for the optimal solution in real engineering problems by simulating the crossover and mutation of chromosomes in the evolution of natural organisms[5]. It has three elements: parameter coding, initial population setting and genetic operator. However, the traditional genetic algorithm has some defects, such as premature convergence, decreasing population diversity and weak local search ability.

3. ANT COLONY OPTIMIZATION

Ant colony optimization is a simulation and optimization algorithm that simulates the swarm intelligence behavior of ants foraging[6]. Ants release a certain amount of pheromones on the path they pass to release information to other ants. Ant Tong ant colony algorithm is only suitable for a certain scale of calculation. In the initial stage of the implementation process, it is prone to stagnation due to the small number of paths recorded in the tabu table. After a long period of time, the pheromone on the better path can be significantly higher than that on other paths[7]. With the passage of time, the difference becomes more and more obvious, and finally converge. By perceiving the strength of pheromones to choose the path to go next, groups cooperate with each other to better adapt to the environment.

* 13739149@qq.com

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 1250618
 · 0277-786X · doi: 10.1117/12.2661806

4. DESIGN OF GENETIC ANT COLONY FUSION ALGORITHM

4.1. Fusion point design

By analysing the characteristics of genetic algorithm and ant colony algorithm, the fusion point design of these two algorithms is shown in Figure 1:

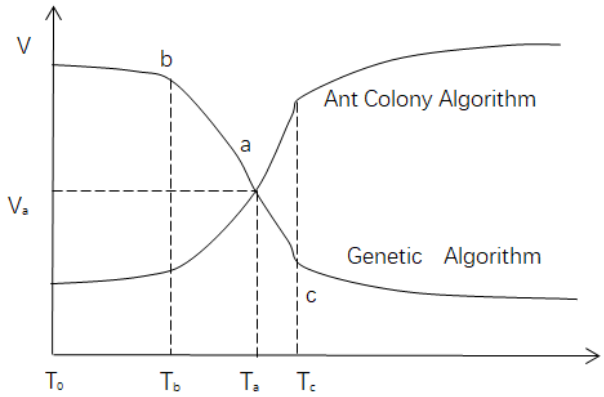

Figure 1. Design of genetic ant colony fusion point.

In Figure 1, in the start time period T_0, T_a, genetic algorithm maintains a high evolution rate, but in T_b. After time, because genetic algorithm cannot effectively use heuristic information, its individual evolution rate begins to decrease gradually. On the contrary, the ant colony algorithm performs the following tasks in the time period T_0, T_a Due to the lack of early pheromones, the individual evolution rate is low, but in T_b After time, the individual evolution rate of ant colony algorithm is improved. Therefore, Ta is the best time point for the end and start of these two algorithms.

The algorithm steps are as follows:

(1) The minimum number of operations $G\varphi_{min}$ and the maximum number of operations $G\varphi_{max}$ are defined.

(2) At the minimum evolution rate, Ge_{min} is the individual evolution rate of the actual iteration process of genetic algorithm.

(3) Algorithm end condition. In one case, if the number of iterations $G\varphi$ of the genetic algorithm is controlled within the range of $G\varphi_{min}, G\varphi_{max}$, and its chromosome evolution rate is smaller than the minimum evolution rate Ge_{min} for N consecutive generations, the genetic algorithm is ended. In another case, when the number of runs $G\varphi$ of the algorithm is greater than the maximum number of runs $G\varphi_{max}$, the genetic algorithm is also terminated.

4.2. Transformation of initial pheromone

After the fusion point design is completed, the local optimal solution of the first algorithm needs to be converted into the initial pheromone of the second algorithm. The method adopted in this paper is: after the first algorithm runs the harness, the chromosome individuals with the highest fitness function value are screened out according to the set proportion (the set value here is 25%), the screened chromosomes are converted into the number of each resource point of cloud computing, and each resource point is converted into the pheromone path (m, n) of ants. The formula is as follows:

$$\varphi_i^{GA}(t) = \theta \sum_j^{25\%M} x_j \tag{1}$$

$$\varphi_i(t) = \varphi_0 + \varphi_i^{GA}(t) \tag{2}$$

In equations (1) and (2), x_j represents the number of ants retained in resource point I by the individual with the jth genetic chromosome. θ represents pheromone conversion factor. The algorithm flow design in this paper is shown in Figure 2:

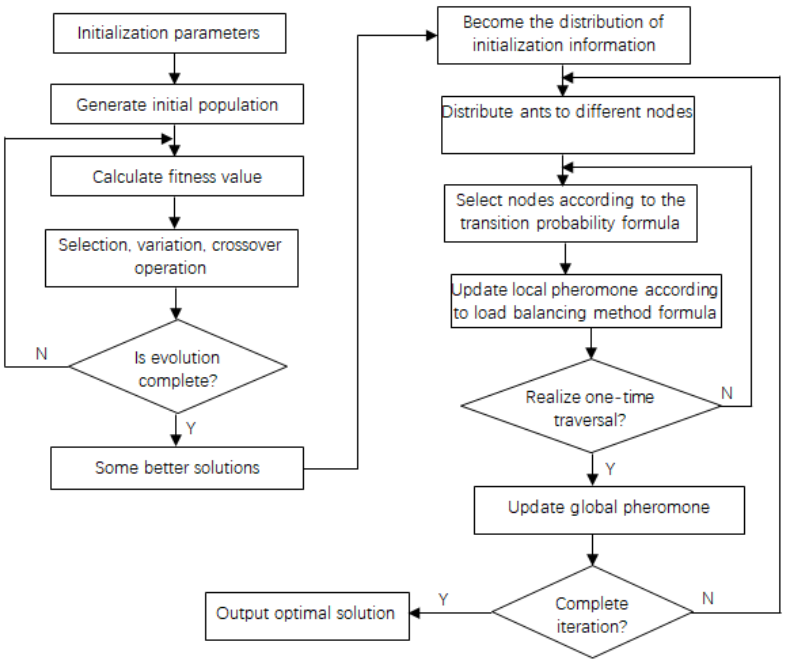

Figure 2. Flow chart of fusion algorithm.

4.3. Design of resource scheduling

The resource allocation of cloud computing is based on the reasonable allocation of N pending tasks uploaded by users to m computing resource nodes in the system under certain constraints. The design of resource allocation model requires that the task can be completed in a short time, and the power consumption of the system can also be kept at a low level. This paper designs a resource allocation model, which is specifically defined as follows:

Definition 1: task set $\mathbf{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_i}, \dots \boldsymbol{x_n}\}$ is the task to be processed, n refers to the number of tasks, $\boldsymbol{x_i}$is the ith task to be processed.

Definition 2: the calculation node set $\mathbf{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_j}, \dots \boldsymbol{y_m}\}$ is the calculation node included in the system, m refers to the number of calculation nodes, $\boldsymbol{y_j}$ refers to the jth allocable computing node.

Definition 3: $\boldsymbol{T_{exe}}$(i, j) represents the expected time to complete task I on node j, using $\boldsymbol{X_l(i)}$ represents the data length of task I, $\boldsymbol{Y_c(j)}$ represents the execution rate of the computing node j. Then $\boldsymbol{T_{exe}}(i, j)$ can be expressed as:

$$\boldsymbol{T_{exe}(i,j)} = \frac{\boldsymbol{X_l(i)}}{\boldsymbol{Y_c(j)}} \tag{3}$$

Definition 4: $\boldsymbol{T_d(i,j)}$ represents the expected time required to transfer task I to computing node j. $\boldsymbol{X_s(i)}$ represents the amount of data that task I needs to transmit, $\boldsymbol{Y_d(j)}$ represents the data transmission rate of computing node j, then $\boldsymbol{T_d(i,j)}$ can be expressed as:

$$\boldsymbol{T_d(i,j)} = \frac{\boldsymbol{X_s(i)}}{\boldsymbol{Y_d(j)}} \tag{4}$$

Definition 5: $\boldsymbol{T_{time}(i,j)}$ represents the expected time required to transfer the task I to the computing node j, and its value can be expressed as:

$$T_{time}(i,j) = T_e(i,j) + T_d(i,j) \tag{5}$$

Since the resource allocation of cloud computing is parallel, each computing node completes its own work alone, then the expected time cost of the system to complete all tasks is $\boldsymbol{T_c}$ can be expressed as:

$$T_c = \max(\textstyle\sum_{i=1}^{m} T_{time}(i,j)) \tag{6}$$

Power cost of completing resource allocation C_c can be expressed as:

$$C_c = \sum_{j=1}^{n}\sum_{i=1}^{m}(T_{time}(i,j) \times C_e + T_d(i,j) \times C_d) \tag{7}$$

In equation (7), C_e and C_d represents the power consumption of computing nodes in completing computing and transmission in unit time.

4.4 Improvement of ant colony algorithm

4.4.1 Improve Pheromone. In order to reduce the ant colony algorithm falling into local optimization, we set the pheromone Volatilization Coefficient to achieve the adaptability of the ant colony algorithm. The adaptive pheromone coefficient volatilization formula is:

$$\rho(t) = \begin{cases} 0.8\rho(t-1) & 0.8\rho(t-1) \geq \rho_{min} \\ \rho_{min} & other \end{cases} \tag{8}$$

In formula (8), $\boldsymbol{\rho(t)}$ is the pheromone coefficient at time *t*. And $\boldsymbol{\rho_{min}}$ is the minimum value of the pheromone coefficient. The minimum value is set to avoid the algorithm convergence speed from falling too fast.

4.4.2 Design Load Balancing Adjustment Factor. In order to avoid that too many tasks are assigned to the same node, resulting in overload of the node, a load balancing adjustment factor F is introduced here, as shown in equation (9):

$$F = 1 - \left(\frac{E_j - E_{avg}}{\sum_{j \subset v}\sum_{i \subset Task_j} T_{sum}(i,j)}\right) \tag{9}$$

The value of *F* in equation (9) can be calculated according to equations (3) and (4).

When the task $\boldsymbol{X_i}$ is assigned to the computing resource $\boldsymbol{Y_j}$ for execution, the pheromone of the computing resource $\boldsymbol{Y_j}$ is updated by the method of formula (10) here, and the other computing resources that are not assigned tasks are updated by the method of formula (11).

$$\boldsymbol{\tau_{ij}(t+1) = \left((1-\rho(t))\tau_{ij}(t) + \sum_{k=1}^{m} \triangle \tau_{ij}^{k}(t)\right) \times F} \tag{10}$$

$$\boldsymbol{\tau_{ix}(t+1) = \tau_{ij}(t) \times F} \tag{11}$$

4.5 Design of genetic algorithm

Using genetic algorithm in resource allocation of cloud computing requires the design of fitness function in combination with the actual resource allocation. The larger the fitness function value, the more likely the genetic chromosome will be retained to the next generation. In order to achieve the premise of shorter task execution and completion time, we also need to ensure low power consumption of the system. The fitness function of completion time is designed Time and power cost fitness function $Fitness_{time}$ *and* $Fitness_{power}$.

$$Fitness_{time}(S) = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} T_{sum}(i,j) / (n \times T_c)}{T_c} \tag{12}$$

$$Fitness_{power}(S) = \frac{1}{C_c} \tag{13}$$

In order to find a balance between completion time and power cost, the fitness function of genetic algorithm is:

$$\text{Fitness}(S) = a \times Fitness_{time}(S) + Fitness_{time}(S) \tag{14}$$

In equation (14), *a* represents the completion time weight value, and *B* represents the power consumption cost weight value. Between *a* and *b*, $a + b = 1$, $a \,\&\, b \in 0, 1$ is required.

5. SIMULATION EXPERIMENT

5.1. Experimental environment and parameter setting

The specific parameter settings of datacenter, virtual machine and cloudlet in cloud sim3.0 are shown in Table 1.

Table 1. Experimental environment settings.

Name	Parameter name	Value
Data center	Number of hosts	1-5
	Number of data centers	6
virtual machine	Number of virtual machines	45
	Number of processors	1-2
	Processing speed	100-1000MPIS
	Virtual machine memory	1024-2048MB
	Network bandwidth	200-500MB
Cloud task	Task length	1000-5000MI
	Number of tasks	30-450

The parameter settings of fusion algorithm are shown in Table 2:

Table 2. Parameter settings of this algorithm.

Parameter name	Value	Parameter name	Value
Population size	100	Pheromone heuristic factor	4
Crossover rate	0.55	Expectation heuristic factor	1
Variation rate	0.15	Pheromone intensity	1
Minimum evolution rate	0.15%	random number	0.55
Minimum evolutionary algebra	40	Pheromone transforming factor	0.5

5.2 Experimental environment and parameter setting

This fusion algorithm is compared with the algorithms proposed in References[8-10]. The comparison results of the three algorithms in terms of iteration times, time cost, power consumption cost, node load rate and so on are as follows:

(1) Algorithm iteration times

In the experiment, when the four algorithms gradually increase the number of tasks to 450, the number of iterations of the four algorithms also increases. However, the number of iterations of this fusion algorithm is less than that of the other three algorithms, as shown in Figure 3.

(2) Load balancing comparison

Figure 4 is a comparison diagram of the maximum value of the node load rate. The resource load rate of the four algorithms increases with the increase of the number of tasks, while the load rate of this fusion algorithm is smaller than that of the other three algorithms.

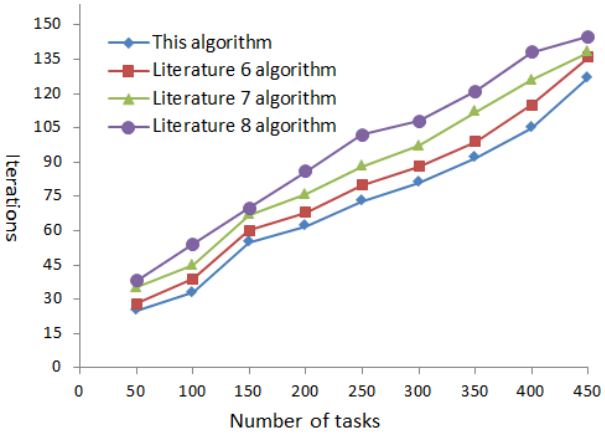

Figure 3. Comparison diagram of iteration times.

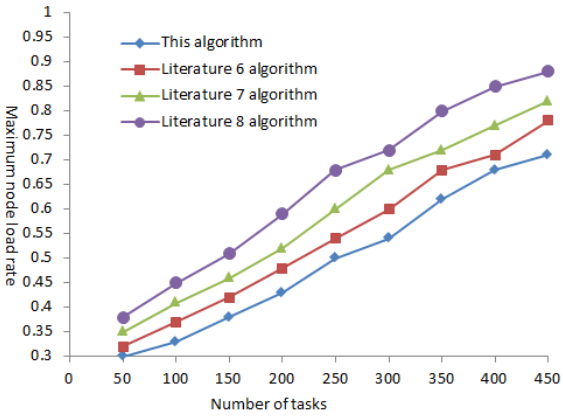

Figure 4. Load balancing comparison chart.

(3) Comparison of time cost and power consumption cost

Figure 5 is a comparison of the time cost of the algorithm. When the number of tasks is small, the time cost curve almost coincides. However, when the number of tasks is large, the fusion algorithm is better than the other three algorithms. Figure 6 is a comparison of the power consumption cost of the algorithm. When the number of tasks is the same, this fusion algorithm is better than the other three algorithms.

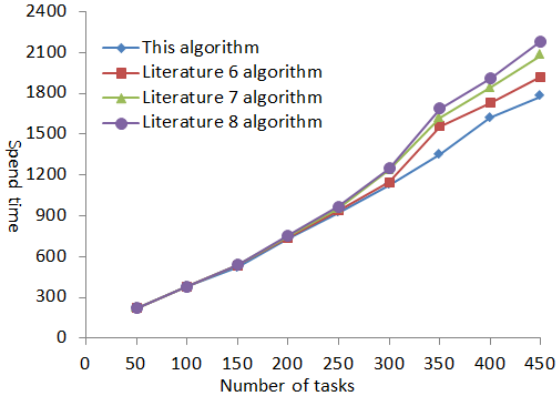

Figure 5. Comparison of time cost.

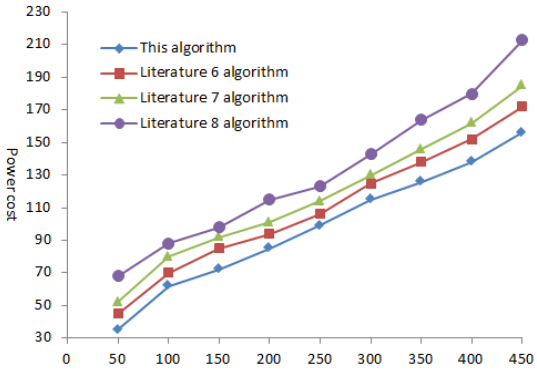

Figure 6. Power consumption cost comparison.

6. CONCLUDING REMARKS

Based on the research of resource scheduling in cloud computing, this paper proposes a genetic ant colony algorithm with load balancing, iteration times, time cost and power consumption cost as the research objectives. This algorithm has certain advantages in time cost, power consumption cost and load balancing. It provides a reference for resource scheduling in cloud computing.

FUNDING

This research was funded by Scientific research project of Guangdong Provincial Department of Education, grant number 2020KTSCX166, and Key scientific research project of Guangdong University of science and technology, grant number GKY-2020KYZDK-10, GKY-2021KYZDK-8.

REFERENCES

[1] Xiao, Y. T., "Cloud Computing resource scheduling based on improved ant colony optimization algorithm," Microcomputer Applications (2), 160-164 (2022).

[2] Ye, C., Yuan, X. P. and Cang, X. H., "Two hypotheses and test assumptions based on Quantum-behaved Particle Swarm Optimization (QPSO)," Cluster Computing (6), 88-96 (2019).

[3] Jain R. E., "An enhanced ant colony optimization algorithm for task scheduling in cloud computing Int. J. Secur. Appl. 13 (4), 91-100 (2020).

[4] Liu, F., Li, B. and Yang, J., "An improved genetic algorithm for cloud computing resource scheduling," Computer Measurement & Control (5), 202-206 (2016).

[5] Liu, K. and Jin, H., "A compromised-time-cost scheduling algorithm for cost-constrained workflows on cloud computing platform," International Conference System Modeling (6), 303-308 (2014).

[6] Zheng, Y., Cai, L. and Huang, S., "Cloud testing scheduling based on improved," ACO International Symposium on Compuers & Informatics (3), 569-578 (2015).

[7] Wei, S. W. and Deng, W., "Cloud resource scheduling based on multi-elite coevolutionary genetic algorithm," Computer Applications and Software (5), 274-280 (2021).

[8] Gao, R. and Juebo, W., "Dynamic load balancing strategy for cloud computing with ant colony optimization," Future Internet (4), 465-483 (2015).

[9] Binsack, Y. and Ralf, W., "The intelligent task scheduling algorithm in cloud computing with multistage optimization," International Journal of Grid and Distributed Computing 9 (4), 313-323 (2016).

[10] Ren, J. and Liu, M., "Task scheduling strategy for cloud computing based on improved GA," Journal of Shenyang University of Technology (5), 320-325 (2019).