Multi-dimensional QoS Cloud Computing Task Scheduling Strategy Based on Improved Ant Colony Algorithm

Junwei Ge¹, Dehua Yu¹* and Yiqiou Fang²

¹ School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China
² School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China

*Corresponding author’s e-mail: cquptydh@163.com

Abstract. This paper proposed a multi-dimensional QoS cloud computing task scheduling algorithm based on improved ant colony algorithm, considering QoS demand of users and load balancing of cloud platform comprehensively. First, this paper defines a QoS model composed of the completion time and execution cost of tasks, and defines the cloud platform load balancing constraint function. Secondly, in view of the shortcomings of ant colony algorithm such as slow convergence speed and easy to fall into local optimum, the pheromone update method and expected heuristic function are improved, and the pheromone strength is dynamically changed. Finally, the simulation is carried out in CloudSim and compared with the ACS algorithm and the MMAS algorithm. Experimental results show that the algorithm in this paper is better than these two algorithms in terms of user satisfaction and cloud platform load.

1. Introduction
Cloud computing is further developed on the basis of grid computing, parallel computing and distributed computing. The goal of cloud computing is to turn the computing resources and storage resources of physical machines into public resources like water and electricity. Users can enjoy high-quality computing and storage services by paying on demand. The task scheduling of cloud computing directly affects the performance of the whole cloud platform, so task scheduling is one of the most important research directions in the field of cloud computing[1-2].

2. Related Work
Literature [3] proposed the optimization scheme of energy consumption cost of data center based on dynamic pricing strategy, so as to reduce the cost of cloud computing center as far as possible on the premise of ensuring the quality of service. Literature [4] proposes to dynamically adjust the resource allocation of applications on this node according to the prediction results of resource demand, and give priority to ensuring the service quality of high-priority applications, so as to improve the overall service quality. Literature [5] rearranges resources through fuzzy clustering of resources and tasks, directs the assignment of tasks with similar attributes, reduces the search scope of resources, and thus reduces scheduling costs and improves user satisfaction. Literature [6] introduced the concept of payoff ratio into the colony algorithm, which effectively improved user satisfaction and resource utilization. Literature [7] introduced the method of copying the parent task in the idle period in the scheduling process, which effectively improved the utilization of resources and user satisfaction.
These algorithms above have improved the service quality of the cloud platform to users to different degrees. However, the QoS demand of users and the load balancing of cloud platform are not considered comprehensively. Therefore, based on the basic ant colony algorithm, this paper establishes the user's QoS evaluation system by considering the task completion time and task execution cost, builds the load balancing constraint function of the cloud platform, improves its heuristic function, pheromone update rules, and dynamically changes the pheromone intensity. Effectively improve the user's service quality, and ensure the cloud platform load balance.

3. Cloud computing task scheduling problem definition

3.1. Cloud task scheduling description

The essence of cloud computing task scheduling is the process of assigning M tasks to N virtual machines according to the corresponding optimization rules. The specific process is shown in Figure 1.

\[ \text{SingleTaskCompleteTime}_{ij} = \text{tt}_{ij} + \text{wt}_{ij} + \text{et}_{ij} \]  

\[ \text{vm}_{j, \text{complete time}} = \max \{ \text{SingleTaskCompleteTime}_{ij} \mid i \in \text{TaskQueue}_j \} \]  

\[ \text{AllTaskFinishTime} = \max \{ \text{vm}_{j, \text{complete time}} \mid j \in \text{Vm} \} \]

3.2. The completion time of the task

The completion time of a task refers to the whole process from the submission of the cloud task to the completion of the cloud task on the virtual machine. In other words, the completion time of a cloud task is composed of the transmission time, the waiting time and the execution time on the virtual machine.

The completion time of a single task is expressed by the following formula:

\[ \text{SingleCompleteTime}_{ij} = \text{tt}_{ij} + \text{wt}_{ij} + \text{et}_{ij} \]  

\[ \text{vm}_{j, \text{complete time}} = \max \{ \text{SingleTaskCompleteTime}_{ij} \mid i \in \text{TaskQueue}_j \} \]  

\[ \text{AllTaskFinishTime} = \max \{ \text{vm}_{j, \text{complete time}} \mid j \in \text{Vm} \} \]

3.3. Cost of the mission

In a real environment, the cost of a task includes multiple aspects. In order to simplify the calculation, the cost of this paper mainly considers three parts: (1) computational cost of task execution in the virtual machine, (2) the bandwidth costs, (3) the storage cost. The cost of a single task, defined as:
\[ SingleTaskCost_{ij} = \begin{cases} 0 & r_{ij} = 0 \\ CostExecute_{ij} + CostBW_{ij} + CostStorage_{ij} & r_{ij} = 1 \end{cases} \]  

\[ SingleTaskCost_{ij} \] represents the total cost of a single task. \( CostExecute_{ij} \) represents the execution cost of a single task. \( CostBW_{ij} \) represents the bandwidth cost of a single task. \( CostStorage_{ij} \) represents the storage cost of a single task. \( r_{ij} = 1 \) represents task \( i \) executed on the virtual machine \( j \). \( r_{ij} = 0 \) indicates that task \( i \) is not executing on the virtual machine \( j \).

The total cost of a single virtual machine to complete all tasks performed on it, defined as:

\[ \text{vm}_{j, \text{cost}} = \sum_{i=1}^{m} \text{SingleTaskCost}_{ij} \]  

(5)

The total cost of all tasks is equal to the sum of the costs of each virtual machine, defined as:

\[ \text{TotalCost} = \sum_{j=1}^{n} \text{vm}_{j, \text{cost}} \]  

(6)

Where \( m \) represents the number of tasks. Where \( n \) represents the number of virtual machines.

### 3.4. Time demand fitness function

\[ Time(I) = \frac{\text{AllTaskFinishTime} - Time_{\text{min}}}{Time_{\text{max}} - Time_{\text{min}}} \]  

(7)

\[ Time_{\text{min}} = \frac{\sum_{i=1}^{m} \text{task}_{i, \text{storage size}}}{n \times \max \{| \text{vm}_{i, \text{compute}} \mid j \in \text{Vm} \}} + \frac{\sum_{i=1}^{m} \text{task}_{i, \text{input size}}}{n \times \max \{| \text{vm}_{i, \text{bw}} \mid j \in \text{Vm} \}} \]  

(8)

\[ Time_{\text{max}} = \frac{\sum_{i=1}^{m} \text{task}_{i, \text{storage size}}}{n \times \max \{| \text{vm}_{i, \text{compute}} \mid j \in \text{Vm} \}} + \frac{\sum_{i=1}^{m} \text{task}_{i, \text{input size}}}{n \times \max \{| \text{vm}_{i, \text{bw}} \mid j \in \text{Vm} \}} \]  

(9)

\( Time(I) \) represents time demand fitness function. \( \text{AllTaskFinishTime} \) represents the total completion time of all tasks. \( Time_{\text{min}} \) represents the minimum completion time for all tasks to run on the virtual machine with the worst overall performance. \( Time_{\text{max}} \) represents the maximum completion time for all tasks to run on the virtual machine with the best overall performance. \( \text{task}_{i, \text{storage size}} \) represents the amount of stored data of task \( i \). \( \text{task}_{i, \text{input size}} \) represents the amount of input data of the task \( i \).

### 3.5. Cost demand fitness function

\[ Cost(I) = \frac{\text{TotalCost} - \text{TotalCost}_{\text{min}}}{\text{TotalCost}_{\text{max}} - \text{TotalCost}_{\text{min}}} \]  

(10)

\[ \text{TotalCost}_{\text{min}} = n \times \min \{| \text{vm}_{i, \text{cost}} \mid j \in \text{Vm} \} \]  

(11)

\[ \text{TotalCost}_{\text{max}} = n \times \max \{| \text{vm}_{i, \text{cost}} \mid j \in \text{Vm} \} \]  

(12)

\( Cost(I) \) represents the cost fitness function. \( n \) represents the total number of virtual machines. \( \text{TotalCost}_{\text{min}} \) represents that all tasks are performed in the virtual machine with the least overall cost, that is the least cost. \( \text{TotalCost}_{\text{max}} \) represents that all tasks are performed on the virtual machine with the highest overall cost, that is the maximum cost. \( \text{Vm} \) represents a collection of virtual machines.

### 3.6. QoS comprehensive fitness function

Through the analysis of users’ QoS demands, the comprehensive fitness function of cloud computing task scheduling can be obtained according to the fitness function of time demand and cost demand obtained, which is defined as:
\[ L = w_1 \times \text{Time}(I) + w_2 \times \text{Cost}(I) \]  

(13)

\( w_1 \) represents time weight factor. \( w_2 \) represents cost weight factor. The weight factor represents the degree of user preference, which can be adjusted according to the actual situation.

3.7. Load balancing constraint function

The standard deviation of the completion time of all tasks of the virtual machine is used to evaluate the load situation of the system. The smaller this value is, the load balance of the whole system is indicated by the following formula:

\[ LB = \sqrt{\frac{\sum_{j=1}^{n} (\text{vm}_{j,\text{complete \_time}} - \text{vm}_{\text{average \_complete \_time}})^2}{n}} \]  

(14)

\( \text{vm}_{j,\text{average \_complete \_time}} \) represents the average execution time of the virtual machine, as follows:

\[ \text{vm}_{\text{average \_complete \_time}} = \frac{\sum_{j=1}^{n} \text{vm}_{j,\text{complete \_time}}}{n} \]  

(15)

In the formula, \( \text{vm}_{j,\text{complete \_time}} \) obtained from (2), \( n \) represents the number of virtual machines.

4. QoS Cloud Computing Task Scheduling Based on Improved Ant Colony Algorithm

Standard ant colony optimization algorithms generally have the shortcomings of slow convergence and easy to fall into local optimum. In order to solve the above deficiencies, the ACS model is used to dynamically change the pheromone intensity, adjust the updating mode of pheromone, and limit the pheromone within the appropriate range by improving the ant colony algorithm.

4.1. Pheromone intensity function

In the early stage of the algorithm, in order to expand the solution space, the influence of previous experience on the algorithm search should be reduced as much as possible, the pheromone on the path should be reduced, thereby expanding the scope of the search. In the middle of the algorithm, the search speed should be increased by increasing the pheromone intensity. In order to avoid too strong pheromone and make the algorithm converge prematurely, please limit the number of pheromone on the path to an appropriate range. In the later stage of the algorithm, expanding the search space has little effect on the results. In order to accelerate the convergence speed of the algorithm, the value of pheromone strength can be further expanded. The function of pheromone intensity \( Q(t) \) over time is in seconds /s, as follows:

\[ Q(t) = \begin{cases} 1 & 0 < t \leq 300 \\ 3 & 300 < t \leq 800 \\ 5 & t > 800 \end{cases} \]  

(16)

4.2. Path selection

A pseudo-random method is added to the standard ant colony algorithm for path selection by adopting the formula of ant colony system[8]. Set a parameter \( q_0 \), and each time the ant chooses a path, it will generate a random number \( q \). If \( q \leq q_0 \), choose a path that maximizes the product of pheromone function and heuristic function. If \( q > q_0 \), makes path selection according to the path selection formula of standard ant colony algorithm.

\[ S = \begin{cases} \text{arg max}_{j \in \text{allowed}} \{ r^\alpha_{ij}(t) \times \eta^\beta_{ij}(t) \} & \text{if } q \leq q_0 \\ r^\alpha_{ij}(t) & \text{otherwise} \end{cases} \]  

(17)
\[ P_{ij}^k(t) = \begin{cases} \frac{t_{ij}^0(t) \times \eta_{ij}^0(t)}{\sum_{j \in \text{allowed}_k} t_{ij}^0(t) \times \eta_{ij}^0(t)} & j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \]  

(18)

Where, \( q \) is a uniformly distributed random value in the interval \([0,1]\). The parameter \( q_0 \) is a preset number \((0 < q_0 \leq 1)\) whose size determines the relative importance of experience and exploration. \( \text{allowed}_k \) represents a collection of optional virtual machines. \( \alpha \) represents pheromone heuristic factor. \( \beta \) represents the expected heuristic factor. \( \tau(t) \) represents the pheromone function. \( \eta(t) \) represents the expected heuristic function, it indicates the expected strength of the task selection to be executed on the virtual machine, as follows:

\[ \eta_{ij}(t) = \frac{1}{wt_{ij} + et_{ij}} \]  

(19)

\( et_{ij} \) represents the execution time of task \( i \) on virtual machine \( j \). \( wt_{ij} \) represents the waiting time of task \( i \) on virtual machine \( j \), and describes the current load of virtual machine \( j \). The smaller the value of \( wt_{ij} \), the less the current virtual machine load.

### 4.3. Pheromone partial update

\[ \tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \Delta \tau_{ij}(t) \]  

(20)

\[ \Delta \tau_{ij}(t) = \sum_{k=1}^{m} \Delta \tau_{ij}^k(t) \]  

(21)

\[ \Delta \tau_{ij}^k(t) \begin{cases} 0 & \text{The kth ant passes through (i, j) in this cycle} \\ \frac{Q}{L} & \text{otherwise} \end{cases} \]  

(22)

When an ant completes the optimization, it updates the pheromone locally. Where \( \tau_{ij} \), represents the pheromone content on the path \((i, j)\) at time \( t \). \( \rho \) is pheromone volatilization factor, representing the degree of pheromone volatilization per unit time, \( \rho \in (0,1) \). The larger \( \rho \) indicates the faster the pheromone volatilizes, that is, the smaller the influence of the past search path on the current process. The smaller \( \rho \) indicates the slower the pheromone volatilization, that is, the greater the impact of the past search path on the current process. \( 1 - \rho \) represents the degree of pheromone residue. \( \Delta \tau_{ij}(t) \) represents the pheromone increment on the mapping path between task \( i \) and virtual machine \( j \) at time \( t \). \( m \) is the total number of ants. Where \( L \) represents the value of the comprehensive fitness function obtained by formula (13).

### 4.4. Pheromone Global Update

\[ \tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \Delta \tau(t)_{ij}^{best} \]  

(23)

\[ \Delta \tau(t)_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{the optimal solution for this iteration} \\ 0 & \text{otherwise} \end{cases} \]  

(24)

\( L_{best} \) is the optimal value of the fitness function so far. When all ants complete a path search, only the global pheromone update is performed on the optimal path. Due to the existence of the volatilization mechanism, the pheromone on the optimal solution path is more than that on other paths, so that it can be play a guiding role in the next search of the ants, forming a positive feedback mechanism, so that the algorithm can find the global optimal solution faster. In order to avoid the pheromone growth on the path corresponding to the global optimal solution, the difference between the pheromone and other paths.
is too large, so that the algorithm is premature, and the pheromone on the path is limited to an appropriate range[9]. The algorithm in this paper is defined as BQ-ACS.

5. Simulation experiment
This article conducts simulation experiments on CloudSim platform. This paper compares the experimental results with the ACS and MMAS algorithms.

ACS: The ant colony system, the ant colony algorithm is greatly changed, and the path construction and pheromone update are optimized. MMAS: Maximum and minimum ant system. The ant colony algorithm has been greatly improved. It only allows the iterative optimal ant or the optimal ant so far to release pheromone, and the amount of path pheromone is restricted to a suitable interval.

5.1. Experimental parameter settings
Setting the number of cloud tasks and virtual machines: The instruction size of the cloud task [5000, 10000]. The space size of the task [300, 1500], in MB. The number of virtual machines is 10, the computing power of a single processor [512, 2048], the unit is Mips. The number of processors per virtual machine [1, 4]. The range of memory [512, 1500], the unit is MB. Virtual The bandwidth range of the machine [500, 1500], the unit is Mbps. The range of the virtual machine hard disk [10, 50], the unit is GB. The processor in the virtual machine is 0.005 yuan/mips, bandwidth is 0.003 yuan/mbps, and memory is 0.003 yuan/MB, hard disk 0.05 yuan/GB.

Parameter setting in the algorithm: The parameter setting in the algorithm refers to the literature [10], and related experiments are carried out. The specific settings are: pheromone heuristic factor $\alpha = 1$, expected heuristic factor $\beta = 5$, pheromone volatilization factor $\rho = 0.6$, $\text{antNum} = 15$, $\text{iter}_{\text{max}} = 200$. Set the same environmental parameters for BQ-ACS, ACS, MMAS, these algorithms in this article.

5.2. Comparison of no preference experiment results ($w_1 = 0.5$, $w_2 = 0.5$)

![Fig 2 No preference-comparison chart of task completion time](image)

![Fig 3 No preference-task cost comparison chart](image)
In the case of no preference, the task completion time of BQ-ACS algorithm in this paper has some improvement compared with ACS and MMAS algorithm. At this time, the cost is lower than ACS, MMAS. At this time, the load balancing situation is also slightly better than ACS and MMAS. This is because the task is executed in full consideration of the user's preference, so it is better than the other two algorithms in terms of overall performance. This shows that, compared with the two algorithms of ACS and MMAS, the algorithm in this paper can better satisfy the user's service quality, and can better realize the load balance of the cloud platform.

6. Conclusion
In view of the previous cloud computing task scheduling algorithms that did not comprehensively consider the user's QoS requirements and the load balance of the cloud platform, this paper proposes a cloud computing task scheduling strategy based on the improved ant colony algorithm QoS constraints.

Task completion time requirement fitness function, cost fitness function, and system load balancing constraint function was established. By changing the task completion time fitness function and cost fitness function weighting factor, the user's preference is expressed. This paper improve the heuristic function and pheromone update method of the ant colony algorithm, limit the pheromone on the path to an appropriate range, and dynamically change the pheromone intensity. Finally, a simulation experiment is performed on the CloudSim platform to verify the effectiveness of the algorithm in this paper. In actual situations, cloud computing scheduling is more complicated. Future research will consider the dependencies between tasks and increase the number of tasks in experiments.

Acknowledgments
Fund Name: Application and Demonstration of Transmedia Big Data in Omnimedia Industry. Chongqing key industry generic key technology innovation project. (Major Theme Project).
Fund number: cstc2017zdcy-zdzx0046.

References
[1] Wang Peng. Cloud computing and big data technology [M]. Beijing: People Post Press, 2014.
[2] M.G. Avram. Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective[J]. Procedia Technology,2014,12.
[3] Wang Wei, Luo Junzhou, Song Aibo. Dynamic pricing based energy cost optimization in data center environment [J]. Journal of Computer, 2013,36(03):599-612.
[4] Li Qing, Li Yong, Tu Bibo, et al. QoS-Guaranteed dynamic resource provision in internet data centers [J]. Journal of Computer, 2014,37(12):2395-2407.
[5] Li Kunlun, Guan Liwei, Guo Changlong. Cloud task scheduling strategy based on clustering and improved symbiotic organisms search algorithm [J]. Computer application, 2018,38(03):707-714.
[6] Yan Liyan, Zhang Guizhu. Multi-dimensional QoS cloud task scheduling based on colony
algorithm [J]. Computer Engineering and Science, 2016,38(04):648-655.

[7] Zhang Qiaolong, Zhang Guizhu, Wu Delong. Multi dimensional QoS cloud task scheduling algorithm based on task replication [J]. Computer Application, 2014, 34(09): 2727-2731.

[8] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53-66.

[9] Stutzle T, Hoos H. MAX–MIN ant system [J]. Future Generation Computer Systems, 2000, 16(8): 889-914.

[10] Duan HaiBin, Ma Guanjun, Liu Senqi. Experimental study of the adjustable parameters in basic ant colony optimization algorithm [C]. //Evolutionary Computation, IEEE, 2007: 149-156.