Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 8, August 2019, pp. 2081-2090
DOI: 10.23940/ijpe.19.08.p7.20812090

Scheduling Algorithm for a Task under Cloud Computing

Yan Li[a,*] and Yao Yao[b]

[a]*Henan Technical College of Construction, Zhengzhou, 450064, China*
[b]*Zhengzhou Institute of Technology, Zhengzhou, 450044, China*

Abstract

Aiming at the problem of low efficiency of task scheduling in cloud computing, this paper first analyzes the task of cloud computing task scheduling. Secondly, the pheromone setting quality function of the ant colony algorithm and the empirical feedback factor of selection probability are improved. The new method is adopted for the colonial calculation method and boundary value processing in the imperial competition algorithm. Finally, the two algorithms are merged to obtain a cloud computing task scheduling algorithm based on the ant colony algorithm-empire competition algorithm. In the simulation experiment, the algorithm demonstrates certain advantages in terms of task execution time, execution cost, and load rate.

Keywords: cloud computing; task scheduling; ant colony algorithm; imperialist competitive algorithm

(Submitted on March 18, 2019; Revised on May 3, 2019; Accepted on June 12, 2019)

1. Introduction

The task scheduling algorithm in cloud computing is an important indicator for measuring the quality of cloud computing services. Selecting a task scheduling algorithm with excellent performance can allow cloud users to run stably and efficiently in the cloud computing platform, shorten the completion time required for tasks, and ensure the economic benefits of cloud computing service providers to the greatest extent [1]. References [2-7] proposed to use the basic ant colony algorithm, particle swarm optimization algorithm, and genetic algorithm to achieve certain effects in cloud computing resource scheduling. Ebadifard [8] proposed a static task scheduling method based on the particle swarm optimization (PSO) algorithm, where the tasks are assumed to be non-preemptive and independent. Yin [9] proposed a new scheduling algorithm based on double-fitness algorithm-load balancing and the task completion cost genetic algorithm (LCGA). The effectiveness of the scheduling algorithm and the availability of the optimization method were proven. Huang [10] proposed a multi-objective optimization algorithm combining the simulated annealing algorithm and genetic algorithm. Wang [11] proposed an ant colony algorithm with adaptive task assignment probability, and simulation experiments showed that the efficiency of the ant colony algorithm in cloud task scheduling was improved. Sun [12] proposed a task scheduling optimization algorithm based on Tabu search, and simulation experiments showed that the algorithm can save time on tasks compared with the basic scheduling algorithm. Ge [13] proposed an improved ant colony task scheduling algorithm with multi-objective optimization, and simulation experiment showed that the basic ant colony algorithm can achieve a good task scheduling effect. Wang [14] proposed a cloud computing task scheduling algorithm based on QoS-based cost fitness. Huang [15] proposed a multi-objective optimal scheduling model based on the fireworks algorithm (FWA), and the model achieved good results in the simulation task scheduling of cloud computing. Cheng [16] proposed a new method called DDR (duplication and direct redistribution), which aims to enable efficient small-large outer joins in cloud computing.
Based on the above research on cloud computing tasks, this paper proposes a cloud computing task scheduling method based on the ant colony algorithm (ACO) and imperial competition algorithm (ICA). Compared with the basic ant colony algorithm and the imperial competition algorithm in the simulation experiment, the algorithm can reduce the user execution time and reduce the user execution cost.

* Corresponding author.
E-mail address: lalei1984@aliyun.com

2. Task Scheduling Target of Cloud Computing

Task scheduling in cloud computing is related to the mapping between tasks and resources. Due to the size and requirements of cloud tasks, most cloud computing platforms can use the reasonable task scheduling method to assign tasks to corresponding resources according to the specific requirements of the actual tasks and the hardware processing resources of the integrated platform. Task scheduling in the cloud computing platform is able to quickly complete the task of user submission, and it has the following main objectives:

(1) Optimal task completion time: after the user submits the task in the cloud, the cloud user wants the cloud service provider to provide the resources corresponding to the task as soon as possible, and it can minimize the cost. The less time it takes for a task to complete, the more the cloud service system can handle the tasks submitted by other users. Therefore, higher economic returns can be obtained, and the optimal time for task completion is very important.

(2) System load balancing: load balancing in cloud services is very important because when the load cannot be balanced, it is easy to cause cloud services to fall into paralysis, or task allocation cannot have appropriate resource correspondence. Therefore, the task scheduling algorithm is very important. When the task scheduling algorithm has certain defects, it is easy to cause uneven resource allocation, which causes some resources to be allocated to most tasks. The remaining resources are not allocated or allocated to a small number of tasks. At the same time, it will cause some tasks to bear too many resources and exit the cloud platform. Too little resource allocation will reduce the system utilization.

(3) Economy: cloud computing task scheduling is not free, and operating costs are very important. This includes cost savings between different roles such as cloud service providers, service operators, and cloud users. Therefore, cloud computing services must consider the types of resources required between these different roles and the economics of service time.

CMDCWS [17] is a widely used workflow task scheduling model for cloud computing. The algorithm is used in the task scheduling of the model, which is a workflow task scheduling model with minimal cost limitation. Define $Sch(T,R,M,TEC,TET)$, and T refers to the collection of tasks. $R=\{r_1,r_2,\cdots,r_n\}$ refers to the collection of virtual machines that can be allocated. Each r_j corresponds to the attributes of virtual machines, M refers to the collection of tasks reflected on the virtual machine, TEC refers to the total implementation cost, and TET refers to the total implementation time. Therefore, the total implementation cost and implementation time are as follows:

$$TEC=\sum_{i=1}^{|R|}C_i\times\left[LET_i-LST_i\right] \tag{1}$$

$$TET=\max\{ET_t:t\in T\} \tag{2}$$

In Equation (1), the value of TEC comes from the total cost of all virtual machines, $|R|$ refers to the number of virtual machines, C_i refers to the unit cost of virtual machine, and LET_i and LST_i refer to the latest end time and earliest beginning time for the virtual machine. Therefore, the required running time is the difference between the two. In Equation (2), task t belongs to an element of the total task collection T. The maximum completion generated at the end of the task is the total execution time in this task schedule. Therefore, it is necessary to construct a threshold that constrains the completion time to ensure that the minimum scheduling scheme that does not exceed the final completion time of the task is the optimal solution.

3. Basic Algorithms

3.1. Ant Colony Optimization

The ant colony algorithm (ACO) is a bionics optimization algorithm that simulates the foraging behavior of real ant colonies. It has positive feedback and distributed and heuristic search characteristics, and it has achieved good results in solving complex optimization problems. It uses pheromones as the medium of communication for individuals in the population, and the equation is as follows:

$$\tau_{ij}=(1-\rho)\cdot\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{3}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/L_k\\0\end{cases} \tag{4}$$

$$p_{ij}^{k}=\frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{s\in allowd_k}\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}} \tag{5}$$

In the above equation, ρ is the pheromone volatile, m is the quantity of ants, τ_{ij}^{k} is the number of pheromones released by any k on the path (i, j), L_k is the length of the path passed by k, and p_{ij}^{k} refers to the possibility of ant k being on path (i, j).

3.2. Imperial Competition Algorithm

In 2007, Atashpaz-Gargari and Lucas proposed a new intelligent optimization algorithm, the imperial competition algorithm (ICA), based on the idea of the imperial colony. It is a group-based optimization algorithm that consists of several individuals for the country. The ICA divides the country into several subgroups. In each subgroup, the algorithm brings the non-optimal country (colonial) closer to the optimal state (empire) through the assimilation mechanism. The algorithm moves one or more colonies to other empire through the empire competition mechanism, allowing information exchange between the empire. The basic ICA algorithm consists of the following four parts:

(1) Generate an initial empire

ICA is a group optimization algorithm consisting of individual countries. For the optimization of N dimensions, generally use a form to represent a country, $country=[p_1, p_2, p_3,\cdots, p_N]$. Set a cost function to measure the size of the state power, $cost=f(country)=f(p_1, p_2, p_3,\cdots, p_N)$. For the minimization problem, the smaller the cost function value, the greater the state power.

(2) Assimilation mechanism

The so-called assimilation, as the name suggests, forces others to identify with others, and the empire adopts an assimilation mechanism for better jurisdiction over other colonies. The ICA algorithm is compared to the assimilation process by the colony to the moving process it belongs to.

The colony moving distance x is defined as follows: $x\leftrightarrow U(0,\beta\times d)$, in which $\beta>1$. When a colony moves close to imp, d means the distance between the colony, and $x\leftrightarrow U(0,\beta\times d)$ is the distribution of a random vector x in $(0,\beta\times d)$. After all the colonies have been updated, the value of the colonies is calculated. When the value of the cost function of a colony is greater than imp, the colony has a larger sphere of influence, so the colony is exchanged with imp. The colony becomes an empire, and the original imp becomes a new colony.

(3) Competitive mechanism

ICA algorithm simulation: in the empire competition, the strongest party tends to control and engulf the colonies of relatively weak empires, thus increasing the competition and making the powerful empire stronger and weaker. The empire may disappear or become another colony. ($r_1, r_2, r_3,\cdots, r_{N_{imp}}\leftrightarrow U(0,1)$). Subtract the vector P from the vector R to get the vector D, and $D=P-R=[D_1, D_2, D_3,\cdots, D_{N_{imp}}]$. The larger the value of the vector D, the greater the power of the corresponding empire, which easily leads to the creation of an empire that the colonies assign to the maximum of the vector D.

(4) Empire demise

The competition between empires leads to a certain degree of strong and weak differences in the empire. When there is no colony in an empire, the empire is deleted.

4. Cloud Computing Scheduling based on Ant Colony-Empire Competition Algorithm

This paper optimizes the shortcomings of the empire competition algorithm and the ant colony algorithm. Because the colony movement in the imperial competition algorithm lacks effective control, the ant colony algorithm and the empire competition algorithm are mixed. This makes the colony have the nature of individual ants through the adjustment of the ant individual algorithm and the optimization of the empire competition algorithm itself. The merged algorithms complement each other and can effectively improve the performance of the overall algorithm.

4.1. Improved Ant Colony Algorithm

4.1.1. Update of Pheromone

In view of the shortcomings of the pheromone in the ACO algorithm, the pheromone is improved in two aspects. The first aspect involves updating the information on the path by selecting the optimal solution obtained by the first few solutions. The second aspect is mainly focused on the current optimal solution path or the global optimal solution path.

$$\tau_{ij}(t+1) = \left[(1-\rho_i)\tau_{ij}(t) + \sum_{w=1}^{r} \Delta\tau_{ij}^{w}(t) \right]_{\tau_{\min}}^{\tau_{\max}} \tag{6}$$

$$\Delta\tau_{ij}^{w}(t) = e^{w}\rho_i\tau_{\max} \tag{7}$$

In the equation, ρ_i is the pheromone volatilization coefficient and $\tau_{\max}$ and $\tau_{\min}$ are the maximum and minimum values of the pheromone, respectively. It indicates that the ants r are updated in order according to the path length of the solution obtained by each, w represents the ordinal number, and e^{w} is a control function. The control is used here because it is able to reflect to some extent the robustness of the features of a good solution. At the same time, in the process of updating the pheromone cf, set the convergence factor, as shown in Equation (8):

$$cf = \frac{\sum_{j=1}^{m} \max\{\tau_{\max} - \tau_{ij}, \tau_{ij} - \tau_{\min}\}}{m \cdot (\tau_{\max} - \tau_{\min})} \tag{8}$$

When $cf < t$, the m symplectic sequence is used to update the pheromone on the path; otherwise, the optimal solution of the current iteration is performed to update the pheromone. t is a number between [0,1].

4.1.2. Set Experience Feedback Factor

The improvement in pheromone mainly involves the use of heuristic ideas, and this idea makes it easy for the ACO algorithm to select the shortest path as an integral part of the entire optimal path. However, the influence of such a selection path on the path of the overall solution cannot be considered as a whole, which leads to certain limitations of the search and the blindness of the algorithm. In order to effectively avoid the possibility of losing the optimal solution in the initial stage of the algorithm, a weight-experience feedback factor is added to the probability of the path of the algorithm to consider the influence of the selected path on the search process. At the initial stage, the ant individual sets a higher weight when searching for the short path. During the execution of the algorithm, when the solution element may guide the ant to the longer path, the weight value is set lower so that the selection can be made. Different paths increase the diversity of multiple path selections and prevent the algorithm from falling into local optimal solutions.

$$w(i,j) = 1 - (|A(i,j)|)^{-1} \sum_{a \in A(r,s)} \frac{c(a) - c_{\text{mid}}}{c_{\max} - c_{\min}} \tag{9}$$

In the equation, $c_{\min}$ and $c_{\max}$ refer to the minimum and maximum length of path between two points, respectively. c_{mid} refers to the average value of the two. $c(a)$ is the length of the path, and $A(i,j)$ is the collection of ants on (i,j).

4.2. Improved Empire Algorithm

4.2.1. Initial Stage of the Improved Empire Algorithm

In the initial phase of the empire algorithm, the initialization function counts the number of each empire colony. According to this calculation method, it is easy to cause the number of colonies in the weakest empire to be zero, which leads to great limitations of the algorithm and affects the generation of the optimal solution. In order to avoid this, the equation is changed to the following expression:

$$p_i = \begin{cases} \max_{i \le j \le N_{imp}} \{c_j\} - c_i, & \text{if } \max_{i \le j \le N_{imp}} \{c_j\} > 0 \\ 0.5(\max_{i \le j \le N_{imp}} \{c_j\}) - c_i, & \text{if } \max_{i \le j \le N_{imp}} \{c_j\} <= 0 \end{cases} \tag{10}$$

$$NC_i = round(|\frac{p_i}{\sum_{j=1}^{N_{imp}} p_j}| \times N_{col}) \tag{11}$$

In the equation, c_i and p_i refer to the cost function of strength of the i empire and NC_i is the number of colonies in the empire.

4.2.2. Update of Colonial Location

In order to avoid the problem of premature convergence in the algorithm, it is possible to move the colony away from the imperialist country and update the location of the colony to the following form:

$$p^c = p^c + (\beta \times \delta - 1). \times (p^i - p^c) \tag{12}$$

In the equation, p^c and p^i refer to the locations of the colony and the imperialist country, respectively. δ refers to the vector of N dimensions, and its value element is a random number between [0,1]. "$.\times$" refers to the product of the same element between two N dimensional vector positions.

4.2.3. Boundary Value Replacement

In the ICA algorithm, when a value that exceeds the search space is replaced with a simple boundary value, it is easy to cause a large number of candidate solutions to be aggregated in the search margin. This will undoubtedly lead to a reduction in search efficiency, so a new replacement strategy is adopted for the boundary value.

$$x_i = \begin{cases} x_i, & \text{if } L_i \le x_i \le U_i \\ 2U_i - (x_i - \left[\frac{x_i - U_i}{U_i - L_1}\right](U_i - L_i)), & \text{if } x_i \ge U_i \\ 2L_i - (x_i - \left[\frac{x_i - U_i}{U_i - L_1}\right](U_i - L_i)), & \text{if } x_i \le L_i \end{cases} \tag{13}$$

4.3. Algorithm Flow

Step 1 Initialize the population. Suppose there are m ants, t tasks, and r virtual machines in the ant colony algorithm. Set the maximum number of iterations. The pheromone vector of k ants is $X_k = (X_{k1}, X_{k2}, \cdots, X_{kt})$, and $X_{kj} = T$ indicates that the j tasks corresponding to the k ants are assigned to the T virtual machines. Set the ant individual to the individual optimal, and apply the empire algorithm. Each empire contains an imperial ant and multiple colonial ants.

Step 2 Pick imperial ants, and divide the empire. The i ant corresponding individual calculates the required cost using

Equation (1) as $Cost_i$ and selects the N_{imp} ant individual with the smallest adaptation value from all ant individuals to become an imperial ant. The individuals of the remaining N_{col} ants are turned into colonial ant individuals. Finally, according to the $Cost_i$ of each ant, the execution cost of the empire ant individual corresponds to the number of colony ants.

Step 3 Assimilation process. In the different small populations of each ant colony, the individuals in each ant colony coordinate with each other, and the pheromone values are adaptively adjusted.

Step 4 Reform mechanism. During each iteration of the ACO algorithm, there are always some individuals with relatively weak fitness who are replaced by new ants with strong adaptability. The reform mechanism can remove the ants with relatively poor fitness values, so that early maturity and local optima can be effectively avoided. The number of individuals in colonial ants depends entirely on the rate of reform. Through the reform mechanism, the latest ants can be compared with the optimal ants of the ICA group, and the individual ants can be found.

Step 5 Empire competition. The power of the empire usually uses the cost as a standard, mainly consisting of the sum of the execution cost of the empire and the average execution cost of the ant individual in the colony.

Step 6 The empire is dying. The weak empire gradually loses all the colonial ants and eventually goes to extinction.

Step 7 When the maximum number is reached, the algorithm ends and the optimal individual is found, that is, the optimal cloud computing task scheduling scheme; otherwise, go to step 3.

5. Experiment Simulation

5.1. Experimental Condition

In order to better illustrate the superiority of the algorithm, the number of tasks set in this paper is 500 and the number of virtual machines is 10. ρ_i is 0.5, and $\tau_{\max}$ and $\tau_{\min}$ are 10 and 1, respectively. The number of empire states is 100, the number of empires is 12, and in the empire algorithm, β is 2. The CPU is Core i3 2, 1GHZ, 4GDDR3, and the hard disk capacity is 1000G, which runs through the cloud platform Cloudsim.

5.2. Algorithm Performance

This paper selects four benchmark functions: the sphere function, Rosenbrock function, Schwefel function, and Rasrtigin function. It compares the algorithm with the empire algorithm and the ant colony algorithm. The maximum number of iterations is 1500, and the benchmark function is 100 dimensions. The sphere function and Rosenbrock function are variable single-mode functions used to verify the convergence speed and convergence accuracy of the algorithm. The Schwefel function and Rasrtigin function are used to verify the local optimal ability and global optimization ability of the algorithm, as shown in Table 1.

Table 1. Comparison results of the three algorithms

Function	Parameter	Ant colony algorithm	Empire algorithm	Algorithm in this paper
Sphere function	Optimal value	1.251E-10	1.93E-9	1.67E-10
	Worst value	5.46E-21	3.28E-16	9.13E-10
Rosenbrock function	Optimal value	1.24E+01	1.612+01	2.73E+01
	Worst value	9.06E+01	7.41E+02	9.42E+01
Schwefel function	Optimal value	1.38E+01	1.42E+01	0.67E+02
	Worst value	3.82E+05	9.36E+06	3.24E+04
Rasrtigin function	Optimal value	4.65E-01	1.72E-01	1.61E -01
	Worst value	2.93E+05	3.87E+07	1.32E+03

It is found from Table 3 that the proposed algorithm is better than the imperial algorithm and the ant colony algorithm. In the sphere function, the test results of this algorithm and ant colony algorithm are lower than those of the imperial algorithm, which shows that the convergence accuracy and convergence effect of the two are inadequate and that the algorithm is not satisfactory in terms of the convergence accuracy and convergence effect. The main reason is that the value range of the test function affects the convergence accuracy test of the algorithm. This shows that the algorithm has certain advantages in local search, primarily because of the improvement of the ant colony algorithm. The ability of the algorithm to improve local search is improved. In the Schwefel function, the results of the three algorithms show that the test function has a general effect on the local optimization and global optimization of the three algorithms. When the

whole result is seen, the algorithm has a good effect. For the Rasrtigin function, the test effect of this algorithm is obvious, which shows that the fusion algorithm can improve the global search ability of the algorithm. According to the comparison of the above four algorithms, the algorithm in this paper has good performance in terms of algorithm convergence and local and global optimization.

5.3. Experimental Results, Process, and Analysis

The purpose of this experiment is to solve the task scheduling problem of the CMDCWS model by the ACO-ICA algorithm. Under the condition limit, a minimum task scheduling strategy is found, and the workload of four different topologies is generated by the workflow generator: Montage (input/output type), Ligo (CPU and memory intensive), Sipht (CPU intensive), and Cybershake (data intensive). Set the number of tasks to 100 to ensure the topology of the task, and the number of instructions of the task are the same size as the task data. The working time is divided into five levels. The first level is the most urgent time, recorded as T1, the second level is more urgent, recorded as T2, and so on. The fifth level is the most relaxed, recorded as T5, and there are four kinds of work. Run the flow 20 times, and calculate the average value of the cost. The results are as follows.

5.4. Execution Cost

As shown in Figure 1, in the MONTAGE workflow, the ACO-ICA algorithm has the lowest cost at T1. IC can absorb the advantages of ACO, so the convergence accuracy is higher. Figure 2 shows that in the SIPHT workflow, the cost of ICA is lower than that of ACO in each completion period, which indicates that the ICA algorithm performs better for CPU-intensive workflows. ACO-ICA absorbs the optimization idea of ICA and optimizes iteration through pheromone updates, feedback factors, and other operations. As shown in Figure 3, in the CyberShake workflow, ICA uses more cost than TCO in T1 and T2. At T3, T4, and T5, it uses less cost than ACO. This shows that in the data-intensive workflow, as the final completion time increases, the performance of ICA begins to slowly exceed ACO, and ACO-ICA minimizes the cost due to the combination of the two advantages. As shown in Figure 4, in the LIGO workflow, ICA uses less cost than ACO for each completion period, which means that ICA's performance is superior to that of ACO for CPU and memory-intensive workflows. ACO-ICA combines the advantages of both algorithms, making the cost the least among the three algorithms. The data is shown in Tables 2-5.ACO has the highest cost. However, as the deadline for completion loosens, the cost of ICA is higher than that of ACO. This shows that when dealing with I/O-intensive workflows, the performance of ICA is worse than that of ACO with the completion of the final completion time. ACO-ICA is composed of several small ant colony individuals. The algorithm still uses the ant colony individual's optimization and movement method.

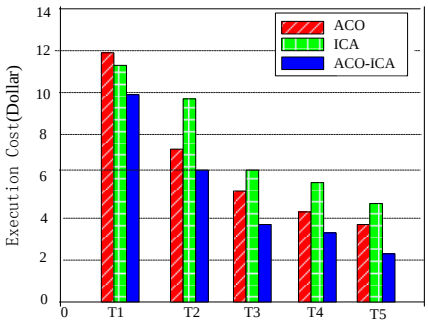

Figure 1. Execution cost of Montage

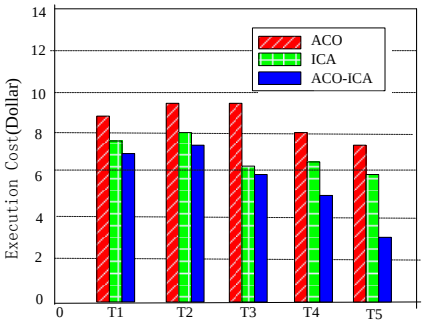

Figure 2. Execution cost of Sipht

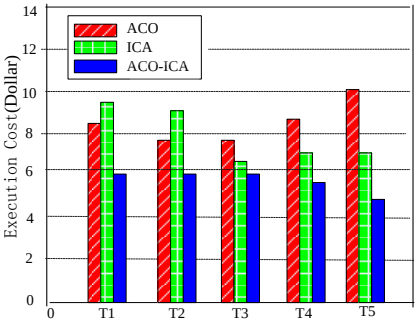

Figure 3. Execution cost of Cybershake

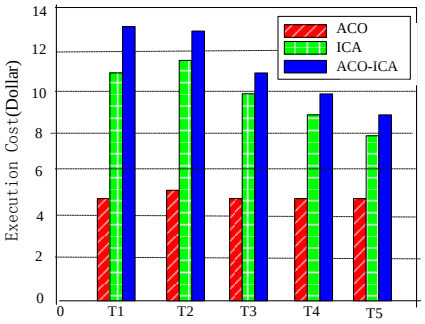

Figure 4. Execution cost of Ligo

Table 2. Execution cost of Montage

	ACO	ICA	ACO-ICA
T1	11.8	11	10
T2	7	9.2	6
T3	5	6	3.8
T4	4.1	5.5	3.8
T5	3.8	4.2	2.2

Table 3. Execution cost of Sipht

	ACO	ICA	ACO-ICA
T1	8.6	7.8	7.2
T2	9.5	8	7.3
T3	9.3	6	5.8
T4	8	6.2	5
T5	7.5	5.9	3.7

Table 4. Execution cost of Cybershake

	ACO	ICA	ACO-ICA
T1	8.2	9.6	5.9
T2	7.8	9	5.8
T3	7.7	6.2	5.8
T4	8.3	7	5.5
T5	10	6.9	4.3

Table 5. Execution cost of Ligo

	ACO	ICA	ACO-ICA
T1	4.2	10.6	13
T2	4.3	11.8	12.6
T3	4.2	9.9	11
T4	4.2	9	10
T5	4.3	7.9	8.8

5.5. Task Completion Rate

As shown in Figure 5, in the MONTAGE workflow, the ICA algorithm has a higher task completion rate than TCO at T1. In T2, ACO's mission completion rate is higher than ICA, and ACO-ICA's mission completion rate at T1 and T2 is higher than that of ACO and ICA. This shows that the ACO-ICA algorithm can guarantee that the completion rate of the task is higher than that of the other two algorithms even when the task completion time is the shortest, which indirectly indicates that the ACO-ICA algorithm has better search ability than ICA and ACO. The task completion rates of the three algorithms T3, T4, and T5 have reached 100%. As shown in Figure 6, in the SIPHT workflow, the ICA algorithm has a higher task completion rate than TCO at T1 and T2. For the CPU-intensive workflow, the three algorithms fail to complete the task in both the T1 and T2 intervals, but the task completion rate of the ACO-ICA algorithm is the highest at T1 and T2. At T3, the task completion rates of the T4 and T5 algorithms are both 100%. As shown in Figure 7, in the CyberShake workflow, the three algorithms fail to complete the task at T1, but the ACO-ICA algorithm has the highest task completion rate among the three algorithms. The three algorithms have a task completion rate of 100% in the four intervals of T2, T3, T4, and T5. As shown in Figure 8, in the LIGO workflow, the task completion rate of the ACO algorithm in T1 is higher than that of ICA, the task completion rate of the ICA algorithm in T2 is higher than that of ACO, and the task completion rate of the three algorithms in T3 is close to 100%. The task completion rate of the ICA algorithm is the highest at T3. At T4 and T5, the task completion rates of the three algorithms have reached 100%. The data is shown in Tables 6-9.

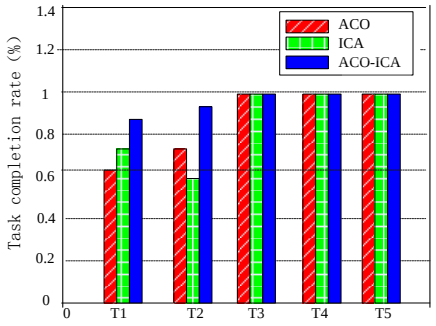

Figure 5. Montage type task completion rate

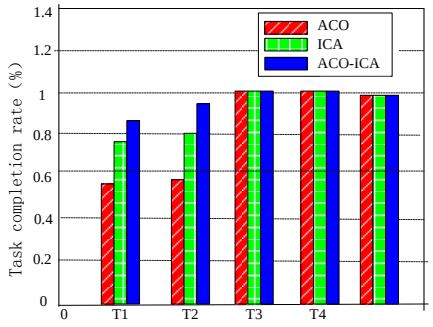

Figure 6. Sipht type task completion rate

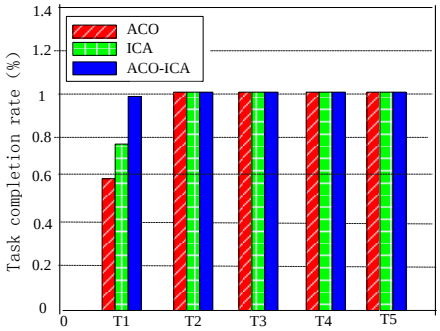

Figure 7. Cybershake type task completion rate

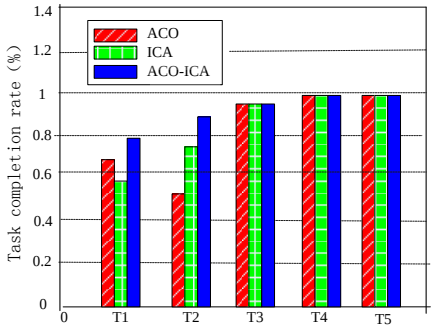

Figure 8. Ligo type task completion rate

Table 6. Montage type task completion rate

	ACO	ICA	ACO-ICA
T1	0.6	0.7	0.9
T2	0.7	0.58	0.95
T3	0.99	0.99	0.99
T4	0.99	0.99	0.99
T5	0.99	0.99	0.99

Table 7. Sipht type task completion rate

	ACO	ICA	ACO-ICA
T1	0.56	0.78	0.82
T2	0.58	0.8	0.9
T3	1	1	1
T4	1	1	1
T5	1	1	1

Table 8. Cybershake type task completion rate

	ACO	ICA	ACO-ICA
T1	0. 58	0. 78	1
T2	1	1	1
T3	1	1	1
T4	1	1	1
T5	1	1	1

Table 9. Ligo type task completion rate

	ACO	ICA	ACO-ICA
T1	0.62	0.58	0.79
T2	0.5	0.78	0.84
T3	0.92	0.92	0.92
T4	0.99	0.99	0.99
T5	0.99	0.99	0.99

6. Conclusions

Aiming at the shortcomings of the existing task scheduling algorithms in cloud computing, this paper improves the ACO and ICA algorithms and integrates the ACO-ICA algorithm. In the simulation experiment, the ACO-ICA algorithm is compared with the other two algorithms in terms of task execution time, execution cost, and system load. The experimental results show that the algorithm has certain advantages.

Acknowledgements

This work is supported by the Henan Higher Education School Young Backbone Teacher Training Program (No. 2016ggjs-201).

References

1. M. Armbrust, A. Fox, and R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, et al., "A View of Cloud Computing," *Communications of the ACM*, Vol. 53, No. 4, pp. 50-58, 2010
2. M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A Survey of PSO-based Scheduling Algorithms in Cloud Computing," *Journal of Network and Systems Management*, Vol. 25, No. 1, pp. 122-158, 2017
3. N. Kumar and P. Patel, "Resource Management using ANN-PSO Techniques in Cloud Environment," in *Proceedings of the 2016 International Congress on Information and Communication Technology*, pp. 419-428, AISC439, Springer, Singapore, 2016

4. V. S. Kushwah and S. K. Goyal, "A Basic Simulation of ACO Algorithm under Cloud Computing for Fault Tolerant," in *Proceedings of the International Conference on Data Engineering and Communication Technology*, Vol. 468, No. 46, pp. 465-472, 2017
5. A. Ragmani, A. E. Omri, N. Abghour, K. Moussaid, and M. Rida, "A Performed Load Balancing Algorithm for Public Cloud Computing using Ant Colony Optimization," *Recent Patents on Computer Science*, Vol. 11, No. 3, pp. 221-228, 2018
6. S. Shahdi-Pashaki, E. Trymourin, and R. Tavakkolmoghaddam, "New Approach based on Group Technology for the Consolidation Problem in Cloud Computing-Mathematical Model and Genetic Algorithm," *Computational and Applied Mathematics*, Vol. 37, No. 1, pp. 693-718, 2018
7. D. Kćo, A. Subasi, and J. Kevric, "Cloud Computing-based Parallel Genetic Algorithm for Gene Selection in Cancer Classification," *Neural Computing and Applications*, Vol. 30, No. 5, pp. 1601-1610, 2018
8. F. Ebadifard and S. M. Babamir, "A PSO-based Task Scheduling Algorithm Improved using a Load-Balancing Technique for the Cloud Computing Environment," *Concurrency and Computation*: *Practice and Experience*, Vol. 30, No. 12, pp. e4368, 2018
9. S. Yin, P. Ke, and L. Tao, "An Improved Genetic Algorithm for Task Scheduling in Cloud Computing," in *Proceedings of 2018 13th IEEE Conference on Industrial Electronics and Applications* (*ICIEA*), pp. 526-530, 2018
10. T. T. Huang, "An Improved Simulated Annealing Genetic Algorithm for Workflow Scheduling in Cloud Platform," *Microelectronics and Computer*, Vol. 33, No. 1, pp. 42-46, 2016
11. J. Y. Wang, "Task Scheduling Method based on Probability Adaptive Ant Colony Optimization in Cloud Computing," *Journal of Zhengzhou University* (*Engineering Science*), Vol. 38, No. 4, pp. 51-56, 2017
12. L. Y. Sun, M. Leng, and P. Zhu, "Load Balancing Task Scheduling Algorithm based on Tabu Search in Cloud Computing," *Mini-Micro Systems*, Vol. 36, No. 9, pp. 1948-1951, 2015
13. J. W. Ge, Q. Guo, and Y. Q. Fang, "A Multi-Objective Optimization Algorithm for Cloud Computing Task Scheduling based on Improved Ant Colony Algorithm," *Microelectronics and Computer*, Vol. 36, No. 9, pp. 1948-1952, 2015
14. J. Wang and S. P. Wang, "Cloud Computing Task Trade-off Scheduling Introduced in QoS Overhead Fitness Computing," *Bulletin of Science and Technology*, Vol. 31, No. 6, pp. 154-156, 2015
15. W. J. Huang and F. Guo, "Multi-Objective Task Scheduling based on Fireworks Algorithm in Cloud Computing," *Application Research of Computers*, Vol. 34, No. 6, pp. 1717-1720, 2017
16. L. Cheng, I. Tachmazidis, S. Kotoulas, and G. Antoniou, "Design and Evaluation of Small-Large Outer Joins in Cloud Computing Environments," *Journal of Parallel and Distributed Computing*, No. 110, pp. 2-15, 2017
17. H. J. Su, "Research on Cloud Computing Task Scheduling Strategy based on Particle Swarm Optimization and Imperial Competition Hybrid Algorithm," China Guang xi Nornal University, Gui Lin, pp. 17-18, 2017

Li Yan is currently a lecturer in the Department of Construction Information Engineering at Henan Technical College of Construction. She received her master's degree in computer software and theory from the Department of Information Engineering at Zhengzhou University in 2008. Her research interests include cloud computing and data mining.

Yao Yao is currently an associate professor in the School of Information Engineering at Zhengzhou Institute of Technology. She received her master's degree in computer applications from the Department of Information Engineering at the University of Zhengzhou in 2008. Her research interests include high-performance computing and web mining.