A Study into Cloud Computing Task Scheduling Based on BIAS Algorithm

Kun Li, Liwei Jia, Xiaoming Shi

Computer Teaching and Research Section Department of Public Infrastructure, Henan Medical College, China
sunlik_1982@126.com, zzujialiwei@126.com, hnshixiaoming@yeah.net

Abstract

Aiming at the low efficiency of cloud computing resource task scheduling and uneven resource allocation, this paper proposes a cloud computing task scheduling strategy that integrates the Berger model into the improved Ant clony and SFLA-BIAS (Berger-Improve Ant Clony Optimization-Shuffled Frog Leaping Algorithm). Firstly, a cloud computing task scheduling model based on time and cost is constructed; secondly, the general balance function of Berger model is used in combination with the virtual machine for probability selection, and the feedback factor is used to optimize the path. Finally, in each individual iteration of ACO, the improved SFLA is introduced to update the individual. In the simulation experiment, BIAS can effectively improve the efficiency of cloud computing task allocation by comparing with the ACO and SFLA algorithms in the virtual machine load, execution time and consumption cost indicators.

Keywords: Cloud computing, Berger model, ACO, SFLA

1 Introduction

With the development of the express delivery of Internet technology and its applications, industrial production and various areas of people's lives will generate massive amounts of data, which requires continuous increases in data processing speeds. However, improving the physical performance and quantity of hardware devices for processing can no longer meet current data processing requirements. The concept of cloud computing meets the needs of the current era. It provides resources to users through the form of services such as basic resource facilities, applications and software platforms by using "charge on demand" as the computing standard [1].

Cloud computing takes the resource pool composed of computers as the carrier. Users obtain the task resources they need on demand without having to care about the specific implementation mechanism and process. Task scheduling allocates virtual resources to more tasks as reasonably as possible through virtual machines via scheduling algorithms. This is the key to cloud computing task scheduling. Task scheduling in cloud computing is generally divided into two parts: mapping the tasks submitted by the user to a set of available virtual machine resources and mapping the virtual machine and host to virtualize host machine creation or migration. Obviously, virtual machines are an important part of task scheduling, and efficient scheduling will directly affect the efficiency of cloud computing systems. Therefore, the use of high-performance scheduling algorithms plays a vital role in task scheduling.

This paper studies the time and cost of cloud computing, proposes a resource scheduling model based on time and cost, and uses the Ant colony algorithm and shuffled frog leaping algorithm to solve resource scheduling. Simulation experiments show that the algorithm has good resource scheduling effects in both small tasks and large tasks.

2 Related Knowledge

Effective allocation of virtual resources to tasks under constrained conditions is the main goal of task scheduling in current cloud computing. This is because the bandwidth, storage, resources, cost and time requirements of each task's own characteristics are different, and the heterogeneity and dynamics of the cloud computing environment further complicate the processing of the problem. Therefore, task scheduling is essentially an NP problem.

For task scheduling under cloud computing, many scholars use metaheuristic algorithms to improve scheduling effects. These include Genetic Algorithm (GA), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Shuffled Frog Leaping Algorithm (SFLA), Bat Algorithm (BA), and Whale Optimization Algorithm (WOA). Due to space limitations, this article only gives examples of the above algorithms. In terms of GA, [2-4] used GA for cloud computing task scheduling, The results showed that the use of GA in cloud computing tasks can reduce task completion time

*Corresponding Author: Kun Li; E-mail: sunlik_1982@126.com
DOI: 10.53106/160792642021112206014

and improve resource utilization. When [5-7] used ACO for task scheduling in cloud computing, the results showed that it effectively reduces the completion time and improves efficiency.

References [8-11] used the PSO algorithm, which achieved certain effects in cloud computing task scheduling. The simulation experiment shows that the improved PSO algorithm can obtain better cloud computing resource scheduling effects, In terms of improved ABC, the results of [12-15] showed that it can effectively reduce energy consumption and save user costs. References [16-18] used improved SFLA in the cloud computing workflow and showed that it has obvious effects on virtual machine scheduling. In addition, [19-22] used improved BA, which optimized cloud computing scheduling and improved efficiency. Reference [23] used WOA, and it further improved the effect of cloud computing task scheduling.

From the above research, it can be found that the metaheuristic algorithm can effectively solve the task scheduling in cloud computing, especially a merged metaheuristic algorithm, which has better scheduling effects in certain aspects of cloud computing. On the basis of the above research, this article constructs a cloud computing scheduling model and fuses different metaheuristic algorithms to further improve the effect of cloud computing scheduling.

3 Cloud Computing Task Scheduling Model

3.1 Virtual Machine Load

With the continuous allocation of resources, the processing capacity of virtual machines and the load of system resources will also change. Obviously, the performance parameters of a single virtual machine cannot represent the dynamic monitoring of the system. To formally describe changes, the following forms are used to record each virtual machine:

$$Load(VM_j) = [id_j, VM_j, vmTime_j] \quad \textbf{(1)}$$

Herein, id_j is used to indicate the number of the j virtual machine, which is unique; VM_j mainly includes the j virtual machine's CPU performance (VM_{j_cpu}), storage capacity (VM_{j_mem}), network bandwidth (VM_{j-bw}) and cost ratio ($VM_{j\_\cos t}$); and $vmTime_j$ refers to the execution time of the j virtual machine to complete the task assigned to it by the system.

3.2 Time and Cost

Suppose I refers to a resource allocation scheme for tasks submitted by users in cloud computing. Based on the virtual machine's operating load model, the completion time of each virtual machine's execution of system delivery tasks can be monitored, $vmTime(VM_j)$ refers to the time for the virtual machine to complete the task, and $finishTime(I)$ refers to the resource allocation time. For the I allocation strategy, the required system completion time should be the maximum completion time of all virtual machines, namely,

$$finishTime(I) = \max(vmTime(VM_j)) \quad (2)$$

In the same way, it is known that the operating cost of each virtual machine VM_j 's unit time is the complete execution cost that $VM_{j\_\cos t}$ can obtain for plan I :

$$finish\mathrm{Cos}\,t(I) = \sum_{j=1}^{m} vmTime(VM_j) \times VM_{j\_\cos t} \quad \textbf{(3)}$$

3.3 Task Scheduling Constraint Function

To better allocate resources, this paper constructs a task scheduling constraint function based on cost and time. By setting different weights, the value of the scheduling constraint function is maximized.

(1) Execution time constraint function:

$$resTime(I) = \frac{finishTime(I) - finishTime_{\min}}{finishTime_{\max} - finishTime_{\min}} \quad \textbf{(4)}$$

Herein, $finishTime_{\max}$ and $finishTime_{\min}$ express the predicted maximum execution time and minimum execution time of the task, $T_{i\_taskLtenth}$ and $T_{i\_InputFilesize}$ respectively represent the execution length of the task in the part of task T_i and the length of other input information, M represents the number of virtual machines, $\min(VM_{j\_cpu})$ and $\max(VM_{j\_cpu})$ represent the minimum and maximum computing power of a single virtual machine, $\min(VM_{j\_bw})$ and $\max(VM_{j\_bw})$ represent a single virtual machine Minimum and maximum communication capacity。respectively; that is, the concurrent execution time of all tasks deployed on the virtual machine with the worst performance and the best performance. The calculation formula is

$$finishTime_{\max} = \frac{\sum_{i=1}^{N} T_{i\_TaskLength}}{M \times \min(VM_{j\_cpu})} + \frac{\sum_{i=1}^{N} T_{i\_InputFileSize}}{M \times \min(VM_{j\_bw})} \quad \textbf{(5)}$$

$$finishTime_{\min} = \frac{\sum_{i=1}^{N} T_{i\_TaskLength}}{M \times \max(VM_{j\_cpu})} + \frac{\sum_{i=1}^{N} T_{i\_InputFileSize}}{M \times \max(VM_{j\_bw})} \quad \textbf{(6)}$$

(2) Constraint function of execution cost:

$$res\operatorname{Cos}t(I)=\frac{finish\operatorname{Cos}t(I)-finish\operatorname{Cos}t_{\min}}{finish\operatorname{Cos}t_{\max}-finish\operatorname{Cos}t_{\min}} \quad (7)$$

Herein, $finish\operatorname{Cos}t_{\max}$ and $finish\operatorname{Cos}t_{\min}$ represent the maximum and minimum execution costs predicted by the task, $Max(VM_{j\_\cos t})$ and $Min(VM_{j\_\cos t})$ respectively represent the maximum and minimum cost consumption of a single virtual machine. respectively; that is, the sum of the costs required to deploy all tasks on the virtual machine with the highest and lowest unit cost. The specific calculation formula is as follows:

$$finish\operatorname{Cos}t_{\max}=finishTime_{\max}\times Max(VM_{j\_\cos t}) \quad \textbf{(8)}$$

$$finish\operatorname{Cos}t_{\min}=finishTime_{\max}\times Min(VM_{j\_\cos t}) \quad \textbf{(9)}$$

Therefore, based on the above two constraint factors, the task scheduling constraint function can be constructed as follows:

$$F(I)=t\times resTime(I)+c\times res\operatorname{Cos}t(I) \quad \textbf{(10)}$$

In the formula, $F(I)$ represents the corresponding objective function of the I-th resource allocation plan, where， t and c are the influence weights of time and cost, respectively; the value range is [0, 1]; and $t+c=1$. That is, for tasks that require high time factors, the proportion of t can be increased. For tasks that are sensitive to cost factors, the weight of c can be increased. When t is the same as c, it not only satisfies the system's constraints on task execution time but also satisfies the user's requirements for cost reduction.

4 BIAS-Based Cloud Computing Task Scheduling

In this paper, the improved Ant colony-SFLA is used in cloud computing task scheduling. First, the individual ants are mapped to cloud computing tasks one by one. Second, the pheromone is updated and combined with the virtual machine using the general balance function of the Berger model for probability selection, and the feedback factor is used to select the path. Finally, in each individual iteration of ACO, the improved SFLA is introduced to update the individual, and the optimal cloud computing scheduling scheme is obtained.

4.1 ACO

ACO is a bionic optimization algorithm that simulates the foraging behavior of real ant colony bodies. It has positive feedback, distributed and heuristic search characteristics and has achieved good results in solving complex optimization problems. The essence of this algorithm is to use pheromones as a medium for communication among individuals in the population. Its formula is as follows:

$$\tau_{ij}=(1-\rho)\bullet\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \quad \textbf{(11)}$$

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/L_k\\0\end{cases} \quad \textbf{(12)}$$

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[w(i,j)][\tau_j(t)]^{\alpha}[\eta_j]^{\beta}}{\sum_{i=1}^{M}[w(i,j)][\tau_i(t)]^{\alpha}[\eta_i]^{\beta}}\\0 \qquad otherwise\end{cases} \quad \textbf{(13)}$$
$$if\ i,j\in[1,M],j\notin tabu_k$$

In the above formula, ρ refers to the pheromone volatile factor, m refers to the number of ants, τ_{ij}^{k} refers to the number of pheromones released by ant k on path (i,j), L_k refers to the path length covered by ant k, and p_{ij}^{k} refers to the probability for ant k to choose path (i,j).

4.1.1 Improved Pheromone

To improve the pheromone, the ACO pheromone update process is divided into two steps. This article limits the value of the pheromone to a specific interval [A, B]. The first step is the sequential update stage, which uses the optimal solution of the first n iterations to update the pheromone on the path. The second step is the optimal solution of the current iteration or the pheromone on the global optimal path that can be updated.

$$\tau_{ij}(t+1)=\left[(1-\rho_i)\tau_{ij}(t)+\sum_{w=1}^{r}\Delta\tau_{ij}^{w}(t)\right]_{\tau_{\min}}^{\tau_{\max}} \quad \textbf{(14)}$$

$$\Delta\tau_{ij}^{w}(t)=F(w)\rho_i\tau_{\max} \quad \textbf{(15)}$$

Herein, when $(1-\rho_i)\tau_{ij}(t)+\sum_{w=1}^{r}\Delta\tau_{ij}^{w}(t)$ is greater than A, the value of formula (14) is A. When $(1-\rho_i)\tau_{ij}(t)+\sum_{w=1}^{r}\Delta\tau_{ij}^{w}(t)$ is smaller than B, the value of formula (14) is B. Under other circumstances, the value of formula (15) is $(1-\rho_i)\tau_{ij}(t)+\sum_{w=1}^{r}\Delta\tau_{ij}^{w}(t)$. In the formula, ρ_i refers to the pheromone volatilization coefficient, and $\tau_{\max}$ and $\tau_{\min}$ refer to the maximum value and minimum values of the pheromone, respectively. r refers to the update according to the path length of the respective solutions obtained from small to large by ant r, w refers to the serial number, and $F(w)$ is the quality function with expression $F(w) = 1/w$.

4.1.2 Path Selection of Feedback Factors

The improvement of pheromones mainly considers the influence of heuristic information on the path. This makes it easy for ants to choose the shortest path in the initial iteration of the algorithm without considering the influence of the combination of paths selected as the solution on the final path length, which leads to blindness of the search. Through research, it is found that the overall length of the path where some short solution elements are located is often very large, which causes the algorithm to eventually converge to the local optimum instead of the global optimum.

To avoid the possibility of losing the original optimal solution at the initial stage of the algorithm, it is necessary to add a feedback factor to the path that the ants travel to consider the long-term impact of the selected path on the search process. In the initial stage, when the solution element tends to guide the ant to find a short path, it is given a higher weight. By contrast, when the solution element may lead the ant to a longer path, the weight of the element should be reduced. In this way, ants can choose different paths in the initial search stage, which increases the diversity of multiple paths and prevents the algorithm from falling into the local optimal solution.

$$L = L \times \sum_{a \in A(r,s)} \frac{c(a) - c_{\mathrm{mi}d}}{c_{\max} - c_{\min}} \tag{16}$$

In the formula, $c_{\min}$ and $c_{\max}$ represent the minimum and maximum path length between two points, respectively; $c_{\mathrm{mi}d}$ is the average value of the two; and $c(a)$ is the path length.

4.1.3 Probability Selection Based on Balance Constraints

When ant k selects the next resource node as the current task resource by probability, it obviously lacks the judgment of whether the resource is available. To make better use of virtual machine resources and improve resource utilization, this paper adopts the general expected balance constraint of the Berger model to set, i.e., to select the ratio of the load balance of the virtual machine that meets the task conditions and the similarity of the virtual machine resources to express probability. The balance constraints of virtual machine resources are expressed as follows:

$$\eta = \theta \ln \frac{AR}{ER} \tag{17}$$

In the formula, θ is the equilibrium coefficient whose value is between 0 and 1, and AR refers to the VM load balance. The virtual machine execution time is used mainly because the more time the user spends on tasks, the higher the virtual machine utilization, that is, the higher the load balance. Therefore, its value is $\frac{vmTime(VM_j)}{vmTime(VM_j) + bestTime}$ $bestTime$ refers to the average execution time of each virtual machine in the optimal search path so far, and ER refers to the similarity between the resource parameters expected by the normalized virtual machine and the general resources expected by the task and is usually expressed by Euler's formula. $D_i = \sqrt{\sum_{i=1}^{K} (X_i - Y_i)^2}$ (X_i is the virtual machine's parameter, and Y_i is the task parameter).

4.2 SFLA

SFLA is a heuristic optimization algorithm that executes heuristic searches by executing heuristic functions to obtain the global optimal solution. The idea is to decompose the frog group into different numbers of subgroups, search in the subgroups according to a certain strategy, and perform a global exchange.

4.2.1 Local Optimization

Local update using the differential evolution algorithm randomly selects a target individual to update. The update object of this subgroup is still the worst individual in the subgroup, and the update strategy is not static in the whole optimization process. To achieve the optimization effect of increasing the diversity of the population in the early stage of evolution and increasing the convergence speed of the algorithm in the later stage, the mutation method is used to optimize it.

(1) Mutation operation

In the early stage of the algorithm, to maintain the diversity of the population and improve the global search ability, the worst individual in the subgroup was updated according to formula (18), and three individuals were randomly selected. One of the individuals is used as the target individual, and the other two individuals are used to update the moving step length using the rand difference operator.

Herein, X_{r1} is the target individual, X_{r2} and X_{r3} are two other individuals randomly selected, and X'_w is an newly generated individual. $F_1 \in (0,1)$, $F_2 \in (0,1)$, and $F_1 + F_2 = 1$. In the later stage of the algorithm, to help converge to the best point, the best individual in the subgroup is used as the target individual, and the difference operator best mutation is introduced. The update strategy is formula (19), where X_{r2} and X_{r3}

are two randomly selected individuals, $X_w^{'}$ is a new individual, and X_b is the optimal individual. $F_2 \in (0,1)$ in the group.

$$X_w^{'} = X_{r1} + F_1 * (X_{r2} - X_{r3}) \quad \textbf{(18)}$$

$$X_w^{'} = X_b + F_2 * (X_{r2} - X_{r3}) \quad \textbf{(19)}$$

(2) Select operation

After an iteration, the new individual $X_w^{'}$ and the worst individual X_w of the subgroup are evaluated for fitness. According to the laws of nature, individuals with better fitness values are selected to enter the next generation population, as shown in formula (20).

$$X_w^{'} = X_w \ \ if \ f(X_w^{'}) >= f(X_w) \quad \textbf{(20)}$$

(3) Cross operation

To further provide the local search ability of the algorithm and maintain the diversity of the population, a crossover operation is introduced. The update strategy is shown in formula (21).

$$X_w^{'j} = \begin{cases} X_w^{'j} & rand_i^j \geq CR \\ rand(0,1) & otherwise \end{cases} \quad \textbf{(21)}$$

In the formula, $X_w^{'j}$ is the value of the current individual at the j th dimension, CR is the cross factor, and $rand(0,1)$ is the random factor.

4.2.2 Global Optimization

Suppose the number of all frog subgroups is m, and the optimal frog individuals in each subgroup are $P_b(1), P_b(2), ...P_b(m)$. The global optimal value is P_g. Choose two individuals among $P_b(1)$ and $P_b(m)$ as the father generation. Combine P_g to cross and generate offspring according to formula (22).

$$\begin{cases} P(i) = r_1 P_b(i) + r_2 P_b(j) + r_3 P_b g \\ P(j) = r_1 P_b(j) + r_2 P_b(i) + r_3 P_b g \end{cases} \quad \textbf{(22)}$$

In the formula, the values of $r_1 + r_2 + r_3 = 1$, $r_1, r_2, r_3 \in (0,1)$ and r_1, r_2, r_3 determine the size of the cross region. An elite retention strategy is adopted in the frog population to eliminate the poor individuals, but because P_b represents the internal optimal individuals in each subgroup, this leads to a local optimum. Therefore, crossover operations between the optimal individuals in different subgroups can avoid falling into the local optimum and achieve the global optimum.

4.3 Algorithm Flow

Step 1: Initialize ACO and SFLA parameter values, set relevant parameter values, and perform one-to-one correspondence between tasks in cloud computing task scheduling and ACO individuals;

Step 2: Use optimized pheromone and path update methods for processing;

Step 3: Each individual ant can select appropriate execution resources for the current task under the condition of satisfying the balance constraint;

Step 4: When an individual ant completes an algorithm iteration, the optimal individual is obtained, and a resource allocation plan for all tasks is obtained. Individuals are selected with the help of SFLA;

Step 5: During this iteration, all ants have completed their search process, and the optimal solution in all allocation strategies is calculated. All virtual machines on the path are updated with global pheromones. Skip to Step 6; otherwise, skip to Step 2;

Step 6: The number of iterations increases by 1;

Step 7: If the number of iterations meets the algorithm termination condition, then stop the iteration and output the optimal resource allocation plan; otherwise, go to Step 2.

5 Simulation Experiment

To further illustrate the effect of the BIAS algorithm, ACO and SFLA are selected as reference objects. The hardware platform selects the CPU as a Core i3, memory 4GDDR3, hard disk capacity 100 GB, software platform Windows 7 operating system and software MATLAB 2011. Under different iteration times, different task numbers illustrate the effects of the BIAS algorithm in cloud computing task scheduling, The number of algorithm iterations is set to 200, and the number of ants is set to 8. ρ is 0.01, α is 3, β is 2, θ is 1, w is 2, A is 1, B is 5, r_1 is 0.5, r_2 is 0.4, r is 0.1, F_1 is 0.5, and F_2 is 0.5. The experiment is divided into a comparison of indicators under different iterations, a comparison of indicators under small-scale tasks and a comparison of indicators under large-scale tasks. The related parameters of the small task set are shown in Table 1 to Table 2, and the related parameters of the large task set are shown in Table 3 to Table 4.

Table 1. Collection of small tasks

ID	type	Length	File size	Expect time	Expect bandwidth	Expect Cost
0	0	3000	2000	10		
1	0	5000	4000	15		
2	0	2500	1000	10		
3	1	2000	1000		100	
4	1	2500	1800		1500	
5	1	800	1200		1200	
6	2	3000	1200			16
7	2	1000	1500			6
8	2	1000	1000			10

Table 2. Small task set corresponds to virtual machine configuration

ID	Number of PE	PE speed (MIPS)	RAM (MB)	Bandwidth (bit)	price
0	4	250	2048	2000	7
1	2	128	1024	3000	5
2	2	112	1024	1200	3

Table 3. Large task set

Category	Scope
Number of tasks	[100, 500]
Task length (MI)	[5000, 15000]
General expectation type	0, 1, 2
Expected time (ms)	[50, 500]
Expected bandwidth (bit)	[300, 1500]
Expect cost	[10, 50]

Table 4. Large task set corresponds to virtual machine configuration

Data center		Virtual machine	
Number of data centers	10	Number of virtual machines	50
Number of hosts	[2, 6]	Number of PE	[2, 8]
Management type	Time sharing	PE speed (MIPS)	[250, 2000]
Unit time spent	[1, 15]	RAM (MB)	[512, 2048]
		Bandwidth (bit)	[500, 1500]
		Management type	Time sharing

5.1 Comparison of Indicators under Different Iterations

Figures 1 to Figure 3 show a comparison of the three algorithms at different iterations in terms of time spent, cost and virtual machine load. Figure 1 shows that as the number of iterations increases, the BIAS algorithm can save time compared to ACO. Figure 2 shows that the BIAS algorithm has been very stable in terms of cost, while the cost-consumption curves of the ACO and SFLA algorithms show some twists and turns, indicating that the BIAS algorithm has good stability in terms of cost. Figure 3 shows a comparison of the three algorithms in terms of virtual machine load. From the figure, it is found that the BIAS algorithm has better advantages in virtual machine loading and is obviously superior to the ACO and SFLA algorithms. This indicates that BIAS can effectively reduce the load and improve the processing power of the cloud server.

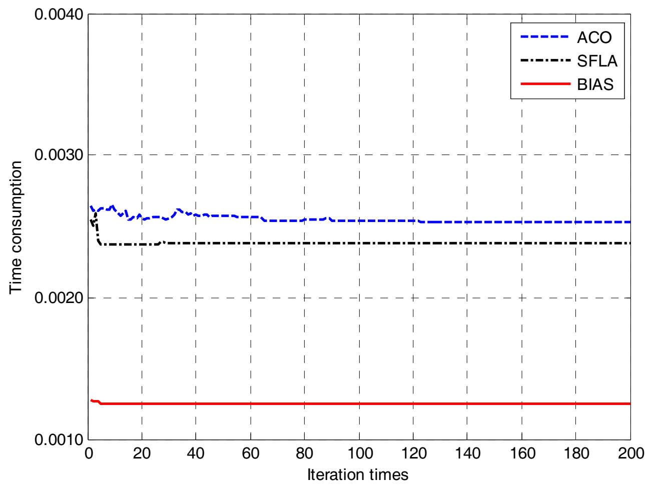

Figure 1. Comparison of time consumption under different iterations

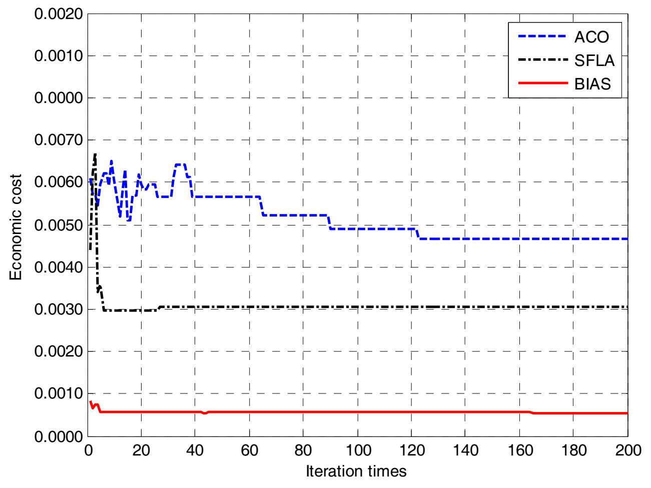

Figure 2. Comparison of economic costs under different iterations

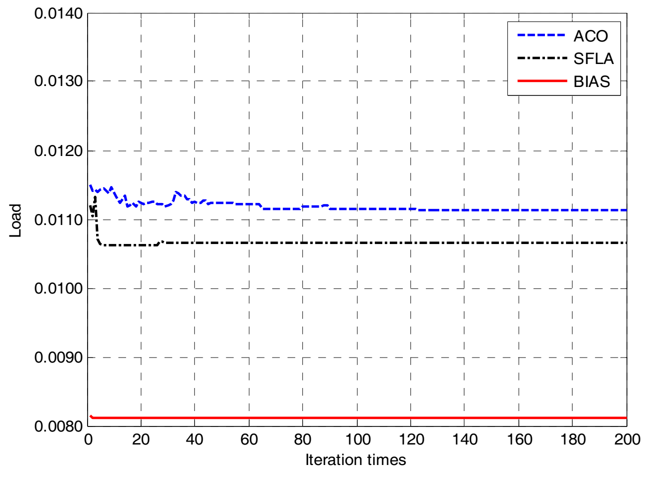

Figure 3. Comparison of virtual machine load under different iterations

5.2 Comparison of Indicators under Small-scale Tasks

The number of small-scale tasks is set from 100 to 1000, with 100 tasks incrementing each time. Figure 4 to Figure 6 show a comparison of the three algorithms under small-scale tasks in terms of time, cost and virtual machine load. From Figure 4, it is found that the time-consuming curves of the three algorithms all show different degrees of fluctuation, and the number of tasks increases with an increasing number of tasks. Overall, BIAS is better than the other two algorithms, with an average reduction of 16.7% and 46.7% compared to SFLA and ACO, respectively.

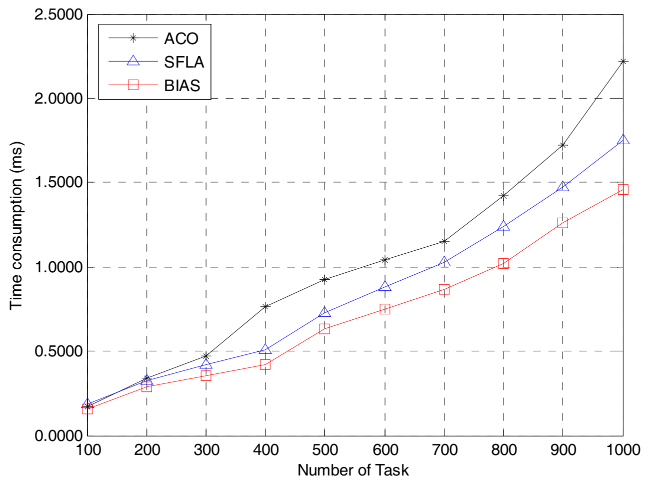

Figure 4. Comparison of time consumption under small-scale tasks

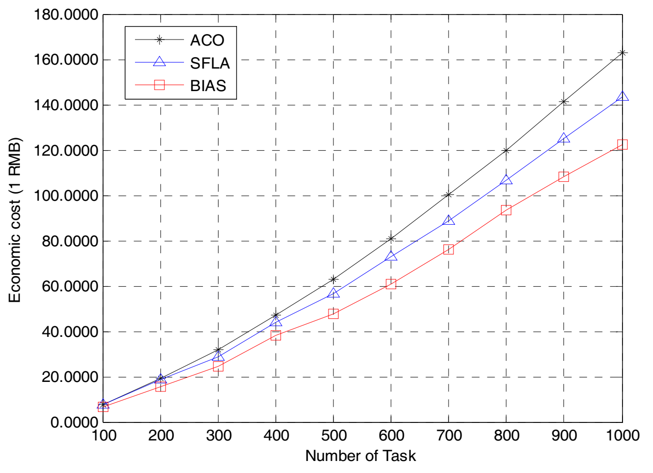

Figure 5. Comparison of cost consumption under small-scale tasks

Figure 5 shows a comparison of the cost of the three algorithms. From the curve, the cost of the BIAS algorithm is significantly lower than those of SFLA and ACO by 16.7% and 33.3%, respectively. This shows that the BIAS algorithm has certain advantages in terms of cost. Figure 6 shows a comparison of the three algorithms in terms of virtual machine load. As

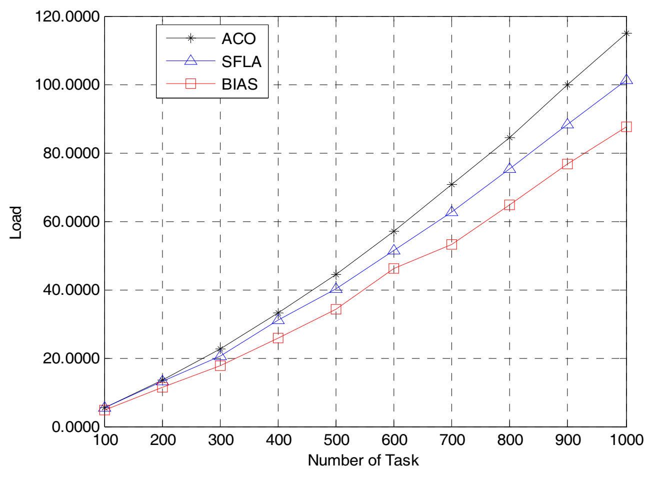

Figure 6. Comparison of virtual machine load under small-scale tasks

the number of tasks increases, the load of the three algorithms increases continuously. However, in terms of the overall effect, BIAS decreased by 11.1% and 27.8% compared with SFLA and ACO, respectively.

5.3 Comparison of Indicators under Large-scale Tasks

The number of large-scale tasks is set from 3000 to 10000, with 1000 tasks incrementing each time. Figure 7 to Figure 9 show a comparison of the three algorithms under large-scale tasks in terms of time, cost, and virtual machine load. From Figure 7, it is found that BIAS has obvious advantages in the comparison of completion time under large-scale tasks. As the number of tasks increases, the curves of the three algorithms show an upward trend. However, the BIAS curve has the least fluctuation and the slowest increase, which indicates that the algorithm performs well in terms of time spent. BIAS is reduced by 20.7% and 40.8% compared with SFLA and ACO, respectively.

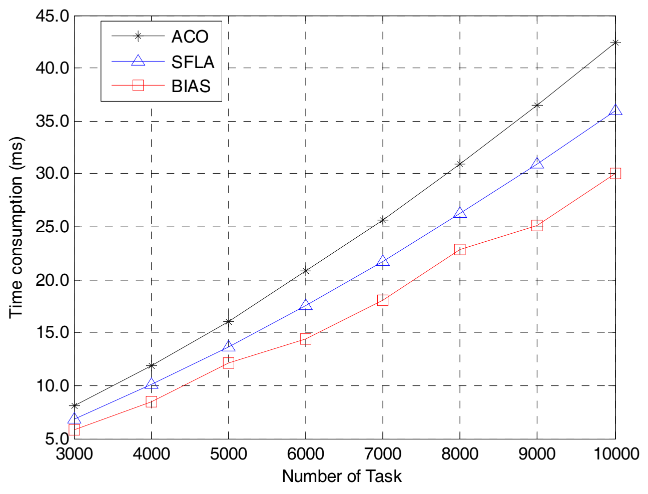

Figure 7. Comparison of time consumption under large-scale tasks

Figure 8 shows a comparison of the cost of the three algorithms. It is found from the figure that the cost of the BIAS algorithm is the lowest. This is mainly because the introduction of the task evaluation mechanism to determine the resources required by the task can effectively save the cost of the task, thus reducing the cost. Along with the increasing number of large-scale tasks, the corresponding curve of BIAS is relatively flat compared with the other two algorithms. This shows that BIAS can be adapted to task scheduling on a large scale in terms of cost, and compared with the SFLA and ACO algorithms, it is reduced by 14.2% and 37.1%, respectively.

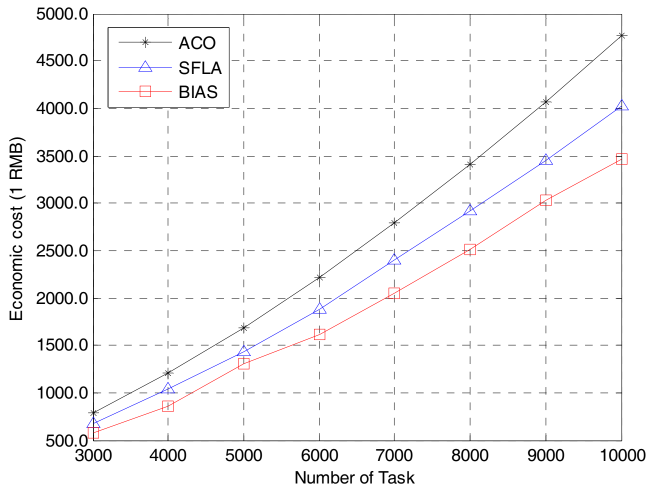

Figure 8. Comparison of cost consumption under large-scale tasks

Figure 9 shows the effect of the virtual machine load under large-scale tasks. The BIAS algorithm has certain advantages over the virtual machine load values of the other two algorithms, which are 18.75% and 37.5% lower than that of the SFLA and ACO algorithms, respectively. This shows that the BIAS algorithm can effectively reduce the virtual machine load value and can be suitable for cloud computing task scheduling on a large scale.

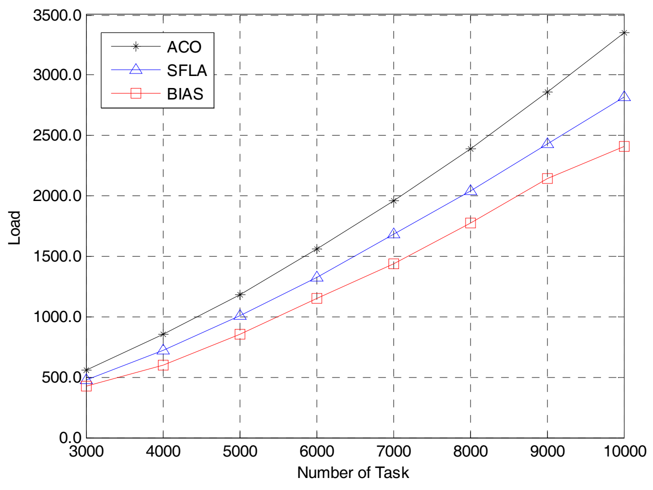

Figure 9. Comparison of virtual machine load under large-scale tasks

From Figure 7 to Figure 9, it can be found that the algorithm in this paper has better advantages in large tasks than the ACO and SFLA algorithms, which shows that the improvement strategy of the ACO algorithm is effective, especially the SFLA algorithm is used in the individual update., Provides a better individual quality for the next iteration. At the same time, the upward trend of the algorithm in this paper in the three figures shows the stability of the algorithm in this paper and the stability of the algorithm in this paper in terms of large tasks.

4 Conclusion

Aiming at the problem of long time and low cost of resource scheduling under cloud computing, this paper uses the fusion algorithm of ant colony and SFLA to solve the problem of cloud computing resource scheduling. First, build a model based on time and cost. Secondly, optimize the pheromone, probability selection and path selection of ACO. Use SFLA to update the individual of each iteration of ACO. Finally, simulation experiments show that the algorithm in this paper has advantages in time and cost. However, the algorithm in this paper does not consider the energy consumption problem in resource scheduling. In future research, a comprehensive analysis of time, cost, and energy consumption will be carried out.

References

[1] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computing-The business perspective, *Decision Support Systems*, Vol. 51, No. 1, pp. 176-189, April, 2011.

[2] J. F. Li, J. Peng, Task scheduling algorithm based on improved genetic algorithm in cloud computing environment, *Journal of Computer Applications*, Vol. 31, No. 1, pp. 184-186, January, 2011.

[3] S. Velliangiri, P. Karthikeyan, V. M. A. Xavier, D. Baswaraj, Hybrid electro search with genetic algorithm for task scheduling in cloud computing, *Ain Shams Engineering Journal*, Vol. 12, No. 1, pp. 631-639, March, 2021.

[4] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, M. U. Chowdhury, An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, *Neural Computing and Applications*, Vol. 32, No. 6, pp. 1531-1541, March, 2020.

[5] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, F. A. Torkey, Cloud task scheduling based on ant colony optimization, *2013 8th International Conference on Computer Engineering & Systems*, Cairo, Egypt, 2013, pp. 64-69.

[6] K. Li, G. Xu, G. Zhao, Y. Dong, D. Wang, Cloud task scheduling based on load balancing ant colony optimization, *2011 Sixth Annual China Grid Conference*, Dalian, China, 2011, pp. 3-9.

[7] A. Gupta, R. Garg, Load balancing based task scheduling

with ACO in cloud computing, *2017 International Conference on Computer and Applications*, Doha, Qatar, 2017, pp. 174-179.

[8] P. Wang, Y. Lei, P. R. Agbedanu, Z. Zhang, Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm, *IEEE Access*, Vol. 8, pp. 29281-29290, February, 2020.

[9] W. Zhong, Y. Zhuang, J. Sun, J. Gu, A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine, *Applied Intelligence*, Vol. 48, No. 11, pp. 4072-4083, November, 2018.

[10] H. Saleh, H. Nashaat, W. Saber, H. M. Harb, IPSO task scheduling algorithm for large scale data in cloud computing environment, *IEEE Access*, Vol. 7, pp. 5412-5420, December, 2018.

[11] Z. Wu, J. Xiong, A Novel Task-Scheduling Algorithm of Cloud Computing Based on Particle Swarm Optimization, *International Journal of Gaming and Computer-Mediated Simulations*, Vol. 13, No. 2, pp. 1-15, April-June, 2021.

[12] B. Hajimirzaei, N. J. Navimipour, Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm, *ICT Express*, Vol. 5, No. 1, pp. 56-59, March, 2019.

[13] M. R. Thanka, P. U. Maheswari, E. B. Edwin, An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment, *Cluster Computing*, Vol, 22, No. 5, pp. 10905-10913, September, 2019.

[14] R. Salem, M. A. Salam, H. Abdelkader, A. A. Mohamed, An artificial bee colony algorithm for data replication optimization in cloud environments, *IEEE Access*, Vol. 8, pp. 51841-51852, December, 2019.

[15] S. Janakiraman, M. D. Priya, Improved Artificial Bee Colony Using Monarchy Butterfly Optimization Algorithm for Load Balancing (IABC-MBOA-LB) in Cloud Environments, *Journal of Network and Systems Management*, Vol. 29, No. 4, pp. 1-38, October, 2021.

[16] J. Luo, X. Li, M. Chen, Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers, *Expert Systems with Applications*, Vol. 41, No. 13, pp. 5804-5816, October, 2014.

[17] P. Kaur, S. Mehta, Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm, *Journal of Parallel and Distributed Computing*, Vol. 101, pp. 41-50, March, 2017.

[18] S. Sharma, A. K. Luhach, S. S. Abdhullah, An optimal load balancing technique for cloud computing environment using bat algorithm, *Indian Journal of Science and Technology*, Vol. 9, No. 28, pp. 1-4, July, 2016.

[19] S. Raghavan, P. Sarwesh, C. Marimuthu, K. Chandrasekaran, Bat algorithm for scheduling workflow applications in cloud, *2015 International Conference on Electronic Design, Computer Networks & Automated Verification*, Shillong, India, 2015, pp. 139-144.

[20] L. Jacob, Bat algorithm for resource scheduling in cloud computing, *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, Vol. 2, No. 4, pp. 53-57, April, 2014.

[21] A. Ullah, N. M. Nawi, M. H. Khan, Bat algorithm used for load balancing purpose in cloud computing: an overview, *International Journal of High Performance Computing and Networking*, Vol. 16, No. 1, pp. 43-54, September, 2020.

[22] M. B. Shareh, S. H. Bargh, A. A. R. Hosseinabadi, A. Slowik, An improved bat optimization algorithm to solve the tasks scheduling problem in open shop, *Neural Computing and Applications*, Vol. 33, No. 5, pp. 1559-1573, March, 2021.

[23] X. Chen, L. Cheng, C. Liu, Q. Z. Liu, J. W. Liu, Y. Mao, J. Murphy, A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems, *IEEE Systems Journal*, Vol. 14, No. 3, pp. 3117-3128, September, 2020.

Biographies

Kun Li received the B.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2005 and received the M.S. degree in computer software and theory from Zhengzhou University, Zhengzhou, China, in 2008. His research interests include cloud computing and algorithm design.

Liwei Jia received the B.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2005 and received the M.S. degree in computer software and theory from Zhengzhou University, Zhengzhou, China, in 2008. His research interests include cloud computing and algorithm design.

Xiaoming Shi received the B.S. degree in computer science and technology from Zhengzhou University Of Light Industry, Zhengzhou, China, in 2005 and received the M.S. degree in computer software and theory from Zhengzhou University, Zhengzhou, China, in 2008. His research interests include algorithm design and multi-agent system.