Research on Multi-Agent Task Optimization and Scheduling Based on Improved Ant Colony Algorithm

Y Wang¹, R R Yang¹, Y X Xu¹, X Li¹ and J L Shi¹

¹ School of Mechanical Engineering,
Inner Mongolia University of Technology,
Hohhot, Inner Mongolia, 010051, China

E-mail:iris.yingwang@outlook.com

Abstract. As a key problem of cloud computing, the performance of task scheduling strategy may seriously affect the efficiency and service quality of the system. With the purpose of achieving balanced task scheduling with optimal completion time, ant colony optimization (ACO) algorithm is adopted as task scheduling strategy in this paper. Considering the problem of premature convergence to local optimal solution of ACO, an improved Max-Min Ant System (MMAS) is introduced to find the global optimal solution. Based on the study of pheromone’s updating mechanism of MMAS, MMAS is applied as task scheduling strategy with reasonable mapping from the objective of shortest path to shortest completion time. The task scheduling strategy based on MMAS has been simulated from views of tasks’ number, size and load balance, and the results shows that MMAS task scheduling strategy is with a better performance on completion time and load balance than ACO based strategy.

1. Introduction
As a business computing model, the cloud computing is developed from distributed computing, parallel computing and grid computing. All the computing capacities, which are provided by a large number of computers, are distributed as a resource pool. The users can dynamically obtain the assigned resource under a guidance of specific scheduling algorithm. Since the cloud computing process is handling huge number of tasks, the task scheduling algorithm, which plays a crucial role of the computing efficiency, can seriously affect the performance of the cloud computing system. How to reasonably allocate computing resources and effectively schedule all the tasks with the purpose of shorter completion time and lower cost is an important issue to study.

The task assignment algorithms mainly include genetic algorithm, ant colony optimization algorithm (ACO), simulated annealing algorithm and so on [1]. This paper mainly focuses the application of ant colony algorithm on task assignment. The earliest generation of improved ACO was called the Ant System with elitist strategy (AS) with the idea of finding the shortest path in each iteration. This strategy allows the ant system to find better a solution as quickly as possible. However, if too many elite ants are used for iteration, the search will quickly converge on a local optimal solution instead of a global one. Therefore, it is necessary to properly decide the number of elite ants.

In 1996, Gambardella and Dorigo proposed Ant Colony System (ACS) [2] with the improvement of local pheromone updating in the process of problem establishment and global updating only with optimal ant path. ACS shows a better performance to find global optimal solution, but when encounters with large-scale optimization problems, the searching time with be relatively long. Aiming
at solving the problems of low efficiency and poor quality of ant system searching, Zhang [3] proposed an algorithm based on the pheromone updating of best-worst ant. The basic idea of this algorithm is to gather more and more ants in the vicinity of the optimal solution as well as avoid premature suspension of searching by controlling the pheromone concentration difference between optimal path and worst path. T. Stuetzle and H. Hoos [4] propose an Max-Min Ant System (MMAS) with high searching efficiency. Compared with the basic ant colony algorithm, MMAS only update the pheromone once of every iteration cycle with the optimal ant or the best ant in the current cycle with the limitation of maximal and minimal pheromone concentrations. These improvements can make MMAS algorithm with the advantages of fast convergence, enhanced searching ability, and it has been widely used.

Over the years, scholars have worked on solving the issues of cloud computing task scheduling. In order to speed up the execution time of cloud computing tasks and improve the utilization of cloud computing resources, the basic and improved ant colony algorithms have been adopted [5]. In [6] ACS is adopted as laxity-based priority algorithm to get an approximate optimal cloud-fog-task scheduling sequence for with reasonable priority. ACS-based scheduling methods are designed to deal with dynamic events with the goals of minimizing the total makespan and the delivery time, respectively in [7]. A multi-objective optimization scheduling method is proposed in [8] considering the trade-off of the performance and budget cost. [9] improves ACS into a scheduling algorithm for time-triggered flows in time-sensitive network. In this paper, the basic ACS and MMAS algorithms are carried out and compared for task scheduling of cloud computing. To accelerate the cloud computing time and increase the computing efficiency, ACS and MMAS algorithms are employed based on the characteristics of cloud computing to solve the task scheduling strategy in the cloud computing environment through in-depth study of the task scheduling strategy of the cloud environment with more complex constraints.

The remainder of this article is organized as follows. Section 2 reviews the preliminary knowledge of the ACS algorithm. Section 3 describes the MMAS method in detail. The application of MMAS ACS algorithm on task scheduling is introduced in Section 4. Section 5 compares MMAS with classic ACO algorithms with different task scales and analyzes the performances from the view of resource node load balance. Finally, the conclusion is drawn in Section 6.

2. Basic Introduction of ACO

2.1. Basic ACO principle
ACO is a probabilistic algorithm for imitation of biological behavior based on the assumption that ants will select path randomly when they are looking for food at the initial time [10, 11]. With the accumulation of pheromone that is left by the ants over iteration searching, most ants are tend to choose the path with high concentrate of pheromone. In the end, the shortest and most optimal path is settled under current conditions by the study of ants’ behaviors.

2.2. Algorithm of basic ACO
To describe the algorithm clearly, we state two prerequisites for the basic ACO algorithm in advance as follows.

Prerequisite 1: When a new corner is encountered, an ant will choose a random path and leave pheromone on the path.

Prerequisite 2: The concentrate of pheromone is inversely proportional to the length of path.

The necessary notation is listed in Table 1 and the basic ACO algorithm is described in Table 2. All involved equations are given in the following of Table 2.
Table 1. Notations.

| Symbols | Parameters |
|---------|------------|
| \( \tau_{ij} \) | pheromone concentrate of path from \( i \)-th to \( j \)-th city |
| \( \alpha \) | parameter indicating importance of pheromone |
| \( \beta \) | parameter indicating importance of heuristic factor |
| \( s \) | any city hasn’t been visited |
| \( allowed_k \) | city set that \( k \)-th ant hasn’t visited |
| \( d_{ij} \) | distance between \( i \)-th and \( j \)-th city defined in (3) |
| \( \eta_{ij} \) | heuristic information defined in (2) |

Table 2. Basic ACO algorithm.

Steps

1 Initialization
   Initialize the variables of heuristic function \( \eta_{ij} \), pheromones \( \tau_{ij} \), \( \alpha \), \( \beta \), maximum iteration number \( N_{\text{max}} \), number of the ants \( m \) and coordinates of all cities. Set the initial pheromone of each path to be zero.
   Locate \( M \) ants randomly to \( n \) cities. Choose next path with probability given in (1). Update passed path to the taboo table and ignore these passed paths in next round.

2 Roulette Strategy
   Calculate pheromone increment of \( k \)-th ant after it walks through any path according to (4); calculate total pheromone increment of each path using (5) and update the pheromone table once of each iteration.

3 Pheromone Updating
   Update optimal paths’ length and clear taboo table after every iteration.

4 Optimal Path Updating
   Terminate iteration if \( N_{\text{max}} \) is reached or the length holds constant.
   The current optimal length is output; otherwise, add the number of iterations by one, and skip to step 2 for the next iteration.

5 Output

The probability that the \( k \)-th ant selects from the \( i \)-th city to the \( j \)-th city is

\[
P_{ij}^{k} (t) = \begin{cases} \frac{\tau_{ij}^{a}(t)\eta_{ij}^{\beta}(t)}{\sum_{j \in allowed_k } \tau_{ij}^{a}(t)\eta_{ij}^{\beta}(t)} & , j \in allowed_k \\ 0 & \text{else} \end{cases} \tag{1}
\]

where

\[
\eta_{ij} = 1/d_{ij} \tag{2}
\]

\[
d_{ij} = \sqrt{(x_{i} - x_{j})^2 + (y_{i} - y_{j})^2} \tag{3}
\]

with the coordinate of \( i \)-th city \((x_{i}, y_{i})\) and coordinate of \( j \)-th city \((x_{j}, y_{j})\).

The pheromone update formula of the path from the \( i \)-th city to the \( j \)-th city assumed at the time \( t+1 \) is

\[
\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \Delta \tau_{ij}^{k}(t) \tag{4}
\]

and the total pheromone increment by all ants on this path is

\[
\Delta \tau_{ij}(t) = \sum_{k=1}^{m} \Delta \tau_{ij}^{k}(t) \tag{5}
\]
where $\rho$ is pheromone attenuation factor; $\Delta \tau^k(t)$ is the amount of pheromone released by the $k$-th ant on this path at time $t$ as follows

$$\Delta \tau^k(t) = Q / L_k$$

(6)

where $Q$ is a constant that indicates the pheromone mass coefficient; $L_k$ is the total walking length of the $k$-th ant in the current round of iteration.

3. **Max-Min Ant System**

Aiming at solving the problems of slow searching at the beginning of iteration and premature falling into local optimization, MMAS algorithm is introduced with the following improvements from the view of pheromone’s updating mechanism.

3.1. **Elite policy for pheromone updating**

Instead of updating pheromone according to contribution of every ant, only the pheromone of optimal path is quantified to be updated for MMAS at the end of each iteration round. The optimal path is discovered by elite ants either in the current round of iteration or in a previous round with better solution [12,13].

After modification, the pheromone update rule on the path from the $i$-th city to the $j$-th city at time $t+1$ is shown in the following

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta \tau_{ij}^{\text{best}}(t)$$

(7)

where $\Delta \tau_{ij}^{\text{best}}$ is the amount of pheromones released by the elite ant, which is defined as,

$$\Delta \tau_{ij}^{\text{best}} = 1 / L(S^{\text{best}})$$

(8)

where $L(S^{\text{best}})$ is the distance of the shortest path found by the elite ant of every iteration.

3.2. **Limitation of pheromones range**

After several times of iteration, some paths would be with high concentrate of pheromone during short time period. It will lead to premature convergence because ants would select this high concentrate path with greater probability. As a result, a better path instead of global optimal one will be regarded as output [14]. To solve this problem, the interval $[\tau_{\text{min}}, \tau_{\text{max}}]$ with maximum and minimum value would be settled to prevent premature stagnation caused by the overlarge difference between the minimal and maximum value. We should take the value of $\tau_{ij}(t)$ cautiously: if $\tau_{ij}(t)$ is more than $\tau_{\text{max}}$, set $\tau_{ij}(t)$ as $\tau_{\text{max}}$; and if $\tau_{ij}(t)$ is less than $\tau_{\text{min}}$, set $\tau_{ij}(t)$ as $\tau_{\text{min}}$.

![Figure 1. Output of ACO optimization.](image-url)
The simulation experiments of ACO and MMAS algorithms are carried out and compared with the same parameters and number of iterations. Figure 1 and Figure 2 show the optimization results of ACO and MMAS respectively. The modified parameters are set as $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $Q = 100$. It can be seen from the results that ACO obtains the optimal solution from 95-th iteration, while the optimized length of MMAS keeps stable after 50-th iteration. Therefore, MMAS is faster than ACO to find the optimal solution, as well as with a higher quality.

It shows that MMAS algorithm has better optimal searching ability and faster searching speed than ACO. For further research, the task scheduling optimization problem will be focus on the application of MMAS.

4. Application of MMAS in Task Scheduling

The performance of task scheduling strategy directly affects the resource utilization. Advanced dispatching strategies are of great significance to improve the utilization efficiency of resources, as well as to reduce operating costs for schools, government agencies, research institutes, and so on [15,16]. The total completion time of tasks and the load balance of resource nodes are two key performance indices for task scheduling strategy.

4.1. Cloud computing task scheduling

![Figure 3. Cloud computing task scheduling structure.](image-url)
By dividing task submitted by users into several tasks of different size, task scheduling strategy can allocate reasonably all the tasks to virtual machine, which acts as resource node. A competent scheduling algorithm can improve the performance of the system by shorten the completion time of tasks, balance the resource load and guarantee the quality of service for users [17]. The optimization results of the scheduling algorithm in this paper are mainly reflected by two indicators: task completion time and load balance.

The structure of cloud computing task scheduling is shown in Figure 3. A large number of tasks requested by users will be divided into n1 subtasks of different sizes and assigned to n2 resource nodes with different processing abilities (speed) in the data center. Based on the length of tasks and the computation speed of resource nodes, the scheduling algorithm acts as a dispatcher to assign proper tasks to the corresponding optimal resource nodes with the purpose of energy-efficient utilization of resources.

4.2. Task scheduling process based on MMAS

The problem of task scheduling using MMAS algorithm is described in Fig. 4. Ant is regarded as a remover carrying subtask to the scheduled resource node. The scheduling algorithm is under the assumptions as follows.

Prerequisite 3: Each subtask can be assigned to only one resource node, and every resource node can handle multiple subtasks.

Prerequisite 4: Only until the current subtask is completed, the subsequent subtask gets the chance to be executed, otherwise the subtask will be in the waiting state.

The idea is that each subtask is randomly distributed to an ant, which will select a reasonable resource node according to MMAS. In solving of MMAS problem, the shorter the path is, the higher possibility of being chosen. While in task scheduling, the better performance (computation processing capability) of the resource node is, with the higher probability to be selected is. The concentrate of pheromone, which is decided by the path length in the form of heuristic information, is of great importance of MMAS algorithm. While for task scheduling, the heuristic information representing the concentrate of pheromone is related with subtask size and processing ability of resource node. The optimal completion time, which is regarded as output of task scheduling algorithm, is defined by ratio of subtask size and processing rate. Differences between MMAS and task scheduling algorithms are summarized in table 3 as follows.

![Figure 4. Task scheduling policy based on MMAS.](image)

| Algorithm | MMAS | Task scheduling |
|-----------|------|-----------------|
| Role of ant | pedestrian | remover |
| Selection principle | shorter path with priority | higher performance with priority |
| Selection factors | concentrate of pheromone | subtask size and processing ability |
| Output | total optimal length | optimal completion time |

Table 3. Compare MMAS with task scheduling algorithm.
4.3. Mathematical model of task scheduling

The application of task scheduling is based on improved MMAS algorithm with key parameters of new meaning as follows.

Assume the number of sub-tasks is $n_1$, number of resource nodes is $n_2$ and number of ants is $m$. The probability of $k$-th ant assigning $i$-th task to $j$-th resource node is given in the following formula:

$$P_k(i, j) = \frac{[\tau(i, j)]^\alpha [\eta(i, j)]^\beta}{\sum_{r \in A_k(i)} [\tau(i, r)]^\alpha [\eta(i, r)]^\beta}, \quad j \in A_k(i)$$ (9)

where $\tau(i, j)$ is the concentrate of pheromone indicating tendency of the choice of $k$-th ant with $i$-th subtask to $j$-th resource node, and it is formulated in (10); $\eta(i, j)$ is heuristic factor shown in (11); $A_k(i)$ is the available set of destination nodes that $k$-th ant can choose, and the occupied node will be removed from taboo list to prevent choice confliction. All relevant formulae are listed in the following:

$$\tau(i, j)(t+1) = (1 - \rho)\tau(i, j)(t) + \Delta \tau_{\text{best}}(i, j)(t)$$ (10)

where $\rho$ is pheromone attenuation factor, which represents the degree of congestion if the $j$-th resource node is occupied; $\Delta \tau_{\text{best}}(i, j)(t)$ is the pheromone released by elite ant until this cycle of scheduling at time $t$.

$$\eta(i, j) = 1/T(i, j)$$ (11)

where $T(i, j)$ is completion time of $i$-th task handled by $j$-th resource node, that is the ratio of the $i$-th task’s length to $j$-th node’s processing speed.

4.4. Algorithm of task scheduling based on MMAS

As formulated in the last subsection, the task scheduling algorithm is described in Table 4.

| Steps | Description |
|-------|-------------|
| 1 Initialization | Initialize the variables of heuristic function $\eta(i, j)$, pheromone concentrate $\tau(i, j)$, $\alpha$, $\beta$, maximum iteration number $N_{\text{max}}$, number of ants $m$, number of subtasks $n_1$ and number of resource nodes $n_2$. Subtask length and processor speed are generated randomly. Choose destination node according to the probability given in (9) with randomly selected subtask. Calculate the execution time of the distributed subtask and using roulette strategy to select the next resource node with updated taboo list. |
| 2 Roulette Strategy | Calculate pheromone increment of $k$-th ant after it assigned all subtasks according to (10). |
| 3 Pheromone Updating | Update the shortest calculation time and clear the taboo list every iteration. |
| 4 Optimal Path Updating | Terminate iteration if $N_{\text{max}}$ is reached or the completion time holds constant. The optimal completion time is the output; otherwise, add the number of iterations by one, and skip to step 2 for the next iteration. |
| 5 Output | |

5. Experimental Simulation and Result Analysis

The experimental simulation is realized and analysed employing task scheduling algorithm based on ACO and MMAS strategies from the views of completion time with different number of subtasks, completion time and resource nodes’ load balance under large-scale and small-scale tasks.

5.1. Completion time of two scheduling strategies with small-scale tasks
Generate subtasks sets with the numbers of 50, 75, 100, 150, 200 and 300 respectively, and the size of each subtask randomly sampled in range of [10,100]. Set 10 virtual machines as resource nodes with different processing speeds randomly sampled in range of [10,100]. The values of parameters are chosen to be $\alpha=2$, $\beta=3$, $\rho=0.4$, $Q=100$ based on expert experience. The experimental results are shown in Table 5 and Fig. 5 based on ACO and MMAS strategies respectively. The completion time is the sum of calculation time and waiting time of all subtasks.

Table 5. Summary of completion time over two scheduling strategies with small-scale tasks.

| Number of tasks | Completion time of ACO/s | Completion time of MMAS /s |
|-----------------|--------------------------|---------------------------|
| 50              | 19.16                    | 17.24                     |
| 75              | 34.37                    | 32.18                     |
| 100             | 48.51                    | 43.22                     |
| 150             | 71.36                    | 64.08                     |
| 200             | 101.56                   | 92.14                     |
| 250             | 128.09                   | 115.12                    |
| 300             | 146.65                   | 131.45                    |

Figure 5. Comparation of completion time over two scheduling strategies with small-scale tasks.

There is no obvious difference of completion time over MMAS and ACO strategies when the number of subtasks is relatively small. However, as the number of subtasks increases, the difference between the two algorithms in the task completion time becomes more and more large and The task completion time of MMAS is always much shorter than that of the ACO algorithm. For example, when the number of subtasks reaches 300, the completion time difference between these two algorithms is nearly 15s. Although MMAS and ACO almost have the same capability when facing small amount of tasks, ACO performs easier to get worse and fall into local convergence with the increasing of tasks’ number comparing with the performance of MMAS. In conclusion, the task scheduling based on MMAS reflects significant better performance than that of ACO algorithm with small-scale tasks.

5.2. Completion time of two scheduling strategies with large-scale tasks
Generate subtasks sets with the numbers of 1000, 1100, 1200, 1300, 1400, and 1500 respectively, and the size of each subtask randomly sampled in range of [100,1000]. Set 50 virtual machines as resource nodes with different processing speeds randomly sampled in range of [10,100]. The values of parameters are chosen to be $\alpha=2$, $\beta=3$, $\rho=0.4$, $Q=100$ based on expert experience. The experimental results are shown in Table 6 and Figure 6 based on ACO and MMAS strategies respectively. The completion time is the sum of calculation time and waiting time of all subtasks.
Table 6 Summary of completion time over two scheduling strategies with large-scale tasks.

| Number of tasks | Completion time of ACO/s | Completion time of MMAS /s |
|-----------------|--------------------------|----------------------------|
| 1000            | 534.15                   | 526.68                     |
| 1100            | 586.05                   | 576.33                     |
| 1200            | 652.00                   | 637.16                     |
| 1300            | 723.36                   | 699.37                     |
| 1400            | 789.95                   | 760.79                     |
| 1500            | 847.26                   | 813.69                     |

Figure 6. Comparison of completion time over two scheduling strategies with large-scale tasks

It can be seen from the results that MMAS algorithm still has obvious advantages comparing with the results of ACO algorithm under the condition of large-scale tasks. The difference between completion times of MMAS and ACO is getting larger and larger and this trend is continued. Combined with the analysis results of small-scale task scheduling, it is found that the task scheduling based on MMAS reflects significant better performance than that of ACO algorithm regardless of the scale of tasks.

5.3. Explore the resource node load balance of the two scheduling strategies

The total task processing time of each resource node is used to measure load balance situation. During the execution process, the recently incoming tasks will be assigned to the resource node with lower load. When a resource node is suffering overload, the task scheduling algorithm will reallocate waiting tasks to other nodes with lower load. Load balancing strategy can be used to measure the efficiency of resource utilization.

The load balance of resource node is quantized by the ratio of task execution time for single node to task execution time for all nodes. Considering the results in Section 5.2, when the number of tasks is set to be 200, the load balance of each node is shown in Fig. 7. It can be seen from the result that load situation of every resource node differs. The load of each resource node of MMAS algorithm is relatively better balanced since the waveform of ACO algorithm’s load balance severely fluctuated comparing with that of MMAS algorithm. So MMAS algorithm is also with better load balance performance comparing with ACO algorithm.
6. Conclusion

According to performance analysis of basic ACO algorithm, an improved MMAS algorithm is introduced and applied as the task scheduling strategy with the purpose of global optimal completion time and load balance in this paper. MMAS algorithm reveals a better performance of task completion time comparing with basic ACO algorithm, both in small-scale and in large-scale tasks. In addition, from the perspective of resource node load balancing, MMAS algorithm also performs better. Therefore, it is feasible for MMAS algorithm to solve task scheduling problems. The adoption of MMAS algorithm on task scheduling can improve the utilization efficiency of source, reduce task completion time and operating costs, and it will provide users with high-quality terminal services.

Acknowledgements

This work is supported by the School Fund of IMUT: Reliability Study of Wireless Networked Control System (ZD201503), the Education Department Fund of Inner Mongolia: Fault Diagnosis Research of the Networked Control System of High-Speed Train (NJZZ16085) and the Natural Science Foundation of Inner Mongolia: Fault Diagnosis Research based on Network Performance Analysis of Coupling Distributed System (2020MS05005).

References

[1] Zhang, S., Wong and T.N. 2018 Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. *J. Intell. Manuf.* 29 585–601
[2] M. Dorigo and L. M. 1997 Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem *IEEE Trans. on Evolutionary Computation* 1(1) 53-66
[3] Yan Zhang, Hao Wang, Yonghua Zhang and Yun Chen, 2011 BEST-WORST Ant System 3rd International Conference on Advanced Computer Control Harbin 392-5
[4] Thomas Stützle and Holger H. Hoos 2000 MAX–MIN Ant System *Future Generation Computer Systems* 16(8) 889-914
[5] L. Zuo, L. Shu, S. Dong, Y. Chen and L. Yan, 2017 A Multi-Objective Hybrid Cloud Resource Scheduling Method Based on Deadline and Cost Constraints *IEEE Access* 5 22067-80
[6] J. Xu, Z. Hao, R. Zhang and X. Sun 2019 A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling *IEEE Access* 7 116218-26
[7] Y. Wan, T. ZUO, L. Chen, W. Tang and J. Chen, 2020 Efficiency-Oriented Production Scheduling Scheme: An Ant Colony System Method *IEEE Access* 8 19286-96
[8] L. Zuo, L. Shu, S. Dong, C. Zhu and T. Hara 2015 A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing *IEEE Access* 3 2687-99
[9] Yang Wang, Jidong Chen, Wei Ning, Hao Yu, Shimei Lin, Zhidong Wang, Guanshi Pang and Chao Chen 2020 A time-sensitive network scheduling algorithm based on improved ant colony optimization *Alexandria Engineering Journal*
[10] Vázquez, K.R 2005 Ant Colony Optimization Genet Program Evolvable Mach 6 459–60
[11] Zhang, Z., Hu, F. & Zhang and N. 2018 Ant colony algorithm for satellite control resource scheduling problem Appl. Intell 48 3295–305
[12] Wei, X., Fan, J., Wang, T. and et al 2016 Efficient application scheduling in mobile cloud computing based on MAX–MIN ant system Soft Comput 20 2611–25
[13] Zheng Y., Yang Q., Jin L. and He L. 2018 A Parallel Pre-schedule Max-Min Ant System Cloud Computing and Security ICCCCS 2018. Lecture Notes in Computer Science 11063 Springer, Cham
[14] Basha M., Siva Kumar M., Pranav V.S. and Rehman B.K. 2018 Approach to Find Shortest Path Using Ant Colony Algorithm Intelligent Communication, Control and Devices. Advances in Intelligent Systems and Computing 624 Springer Singapore
[15] Shen, Y., Bao, Z., Qin, X. and et al 2017 Adaptive task scheduling strategy in cloud: when energy consumption meets performance guarantee World Wide Web 20 155–73
[16] Li, C., Wang, C. & Luo and Y. An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment J. Supercomput 76 6941–68
[17] Singh, P., Dutta, M. & Aggarwal and N. 2017 A review of task scheduling based on meta-heuristics approach in cloud computing Knowl. Inf. Syst. 52 1–51