A Cloud-Side Task Scheduling Algorithm with Multiple Evaluation Metrics

Yong Ma[1], Xinghong Jiang[2], Chenyang Lv[2], Tao Tao[3], Liang Zhou[4(✉)], Qilin Xie[5], and Lingguo Zhen[6]

[1] School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
[2] School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
{jxh,cyang_lv}@jxnu.edu.cn
[3] School of Digital Industry, Jiangxi Normal University, Shangrao, China
[4] China Electric Power Research Institute, Beijing, China
zhouliang@epri.sgcc.com.cn
[5] School of Electronic and Information Engineering, Jinggangshan University, Jian, China
1809102045@jgsu.edu.cn
[6] YUFENG Technology Corporation Limited, Shenzhen, China

Abstract. With the popularity of intelligent terminal devices, edge computing has been fully developed. Power patrol robot is widely used in power grid information collection, and edge computing can effectively shorten response time, improve processing efficiency and reduce network pressure, so as to meet the real-time requirements. However, the following problem is how to realize the scheduling strategy of edge cloud and central cloud and optimize multi performance indicators. To solve this problem, this paper proposes a task scheduling model combining genetic algorithm with Docker container technology and taking cloud computing center and edge cloud into comprehensive consideration. Firstly, the task is classified by condition analysis. Assign tasks to cloud computing centers or edge nodes according to the task type; Genetic algorithm is used to assign tasks to edge nodes. Finally, the performance of the model is verified in the simulation environment. The experimental results show that this task allocation method greatly improves the resource utilization of edge server equipment on the basis of considering the needs of tasks, the limited resources of edge server, and meeting the needs of task proposers.

Keywords: Genetic algorithm · Edge cloud · Amulti-objective restriction · task scheduling

1 Introduction

Substation is one of the core hubs of the power system. Inspection of the equipment in the station is a basic measure to ensure the safety of the system and the

Y. Cao and X. Shao (Eds.): MONAMI 2022, LNICST 474, pp. 212–229, 2023.
https://doi.org/10.1007/978-3-031-32443-7_15

effective operation of the equipment. Domestic power grid enterprises are put forward to realize the new target of "machine patrol+ person patrol" exploring the application of intelligent operation, aims to develop flexibly carrying variety, high-performance and high-precision sensors of intelligent robot, push to "intelligent equipment and intelligent wisdom" transformation and upgrading, alleviate the pressure of the structural vacancies, improve operation quality and equipment health level.

With the boom in IoT, AI and the proliferation of mobile end devices, the scale of users, resources, tasks and workflows is increasing, along with the scale of edge clouds. Task execution and scheduling in the edge cloud in conjunction with the cloud computing centre is of paramount importance. It affects the overall task running efficiency, user quality of service, resource load balancing, etc. In order to improve the efficiency of resource use in the cloud computing environment, optimise performance indicators and meet the requests of multi-user and multi-computing tasks, the problem of resource allocation and task scheduling optimisation has received widespread attention.

Cloud computing environment is a typical distributed computing environment. The services provided by cloud computing include the following three categories: software as a service (SaaS), platform as a service (PAAS) and infrastructure as a service (IAAs) [1]. Task scheduling refers to the process of assigning tasks to appropriate resources to execute according to the actual situation of tasks and resources in IAAs layer [2]. The task scheduling mechanism of cloud computing system mainly focuses on multi performance indicators such as load balancing of edge cloud devices and quality of service parameters (QoS). In recent years, the academic research on task scheduling algorithm is mostly meta-heuristic algorithm, which is the combination of random algorithm and local search algorithm. [3] meta-heuristic algorithm does not depend on specific issues, but as a general heuristic strategy, it can be widely used in function combination optimization and function calculation. The high-precision solution and short overall scheduling cycle are its remarkable features, it can provide a search process for large-scale optimization problems. Therefore, meta-heuristic algorithm is always used to find the approximate optimal solution of NP hard problems.

At present, the research focus of meta-heuristic algorithm is how to balance local search and global search and avoid local optimal solution effectively. These algorithms can be roughly divided into two categories: swarm intelligence optimization and random search optimization [3]. Swarm intelligence optimization algorithm refers to the behavior taken by insects, herds and other groups to complete a certain goal. There is a task selection mechanism and division of labor in the group. Individuals constantly change the search direction according to local rules and the interaction between adjacent individuals, so as to ensure that the global group behavior can achieve a certain goal in the way of approximate optimal solution. On the other hand, the random search optimization algorithm is based on the traditional local search and global search, and further improves the

search mode and adds a specific random algorithm to avoid the local optimal, so as to find the global optimal.

This paper proposes a task scheduling model combining genetic algorithm with Docker container technology and taking cloud computing center and edge cloud into comprehensive consideration. Firstly, the task is classified by condition analysis. Assign tasks to cloud computing centers or edge nodes according to the task type; Genetic algorithm is used to assign tasks to edge nodes. Finally, the performance of the model is verified in the simulation environment.

2 Related Work

In recent years, a variety of methods of task scheduling based on random search optimization have been proposed in academia. Zhang MAO et al. [4] put forward an adaptive individual-assessment scheme based on evolutionary states, and adjust the evolutionary parameters accordingly to handle the constraints in multi-objective optimization problems, which can meet the quality of service needs of users and efficiently utilize cloud resources. [5] proposes a temporal task scheduling algorithm (TTSA) to effectively dispatch all arriving tasks to private CDC and public clouds. In each iteration of TTSA, the cost minimization problem is modeled as a mixed integer linear program and solved by a hybrid simulated-annealing particle-swarm-optimization. H. Krishnaveni et al. [6] proposes an efficient algorithm namely Execution Time Based Sufferage Algorithm (ETSA). By establishing the expected calculation schedule, the difference between the two minimum execution time and completion time is calculated to achieve the best performance in load balancing. [7] using DEA and adaptive optimization strategy, an algorithm is proposed to enhance the fitness by modifying the mutation crossover operation, so as to optimize the execution time and energy consumption. Although the above methods can meet the accuracy requirements of higher solutions, it is limited to numerical solutions, multiple iterations and slow convergence speed.

Many scholars have proposed the task scheduling method based on swarm intelligence optimization. B. Gomathi et al. [8] proposed the Epsilon-fuzzy dominance based composite discrete artificial bee colony (EDCABC) approach. The Epsilon-fuzzy dominance sort approach is used to choose the best solutions from the Pareto optimal solution set in the multi-objective domain. And EDCABC with composite mutation strategies and fast local search method are used to enrich the local searching behaviours which help to avoid the premature convergence. Mandeep Kaur et al. [9] proposed a multi-objective bacterial foraging optimization algorithm (MOBFOA). Based on the improvement of the bacterial foraging algorithm, the method selects the bacteria positions from both the dominant as well as non-dominant fronts to obtain diversity in the solutions obtained. By introducing adaptive step size in chemotactic step, the accuracy and speed of the convergence of the BFOA has been improved. Huang Weijian et al. [10] proposed a multi-objective task scheduling model based on chaotic cat swarm algorithm (CCSO), in which the scheduling algorithm is carried out

through search and tracking modes, and Logistic chaos mapping is used to process experimental data, thus obtaining the optimal task scheduling solution set. Li Hongwei [11] proposed a cloud computing task scheduling strategy based on the improved moth optimization algorithm, introduced the dynamic inertia weight, and used the lateral mutation strategy for the optimal moth position in the algorithm iteration process, so as to avoid the particles falling into local optimization in the later stage of the iteration and improve the multi-objective efficiency. However, the above schemes are complex to implement, highly dependent on parameters and prone to fall into local optimization, which is difficult to meet the requirements of multi-latitude task scheduling optimization objectives. Xingyan Chen [12] proposes a new stochastic approach to optimise the use of resource transfer and resource transcoding in systems that utilise cloud, edge and crowd technologies in collaboration. A joint resource allocation problem is proposed using stochastic optimisation arguments and an accelerated gradient optimisation (AGO) algorithm is designed to solve this optimisation problem in a scalable way that reduces system cost while having higher QOE performance.

Therefore, it is still a difficult problem to achieve the high-precision and multi-dimensional optimization goal of task scheduling and explore the scheduling algorithm with better optimization performance.

3 Preliminaries

This part mainly introduces Docker technology, Linux task scripting technology and genetic algorithm used in the task scheduling model.

3.1 Docker

Traditionally, when we receive a task and want to assign it to an idle edge server, we usually need the server to set up the corresponding runtime environment, configuration files, class libraries, etc. However, different cloud server resources have different environment configurations. In this case, tasks may fail to run due to complex running environment configurations and memory overload. At the same time, when cloud computing resources are dealing with multi-tasks, it is obviously unrealistic to constantly change configurations because different tasks need different configuration environment requirements. To solve this problem, Docker container technology is used in this paper.

Docker container technology provides us with a very standardized solution in cloud computing. Docker technology can be used to easily achieve system migration, provide the virtualization environment required by the task and so on. Traditional applications generally refer to "code as application", in order to run the application, if the environment does not meet the requirements of the application, the application must temporarily configure the environment. However, Docker technology uses the concept of "image is application", which packages all applications into an image starting from the kernel of the system to be run. It can run quickly in the new computer environment through Docker

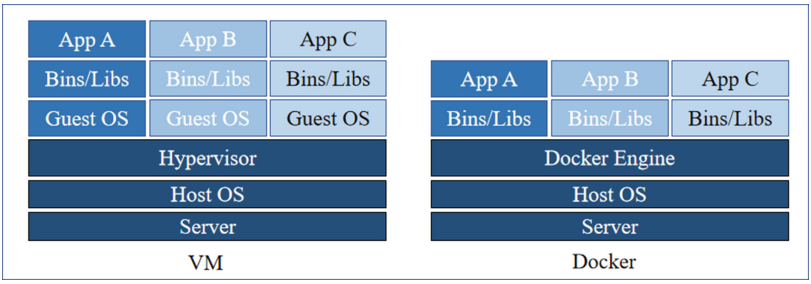

Fig. 1. Compare the traditional VM mode with the Docker container mode

engine. This not only solves the problem of task running environment configuration, but also enables the task scheduling and allocation calculation to get the most suitable cloud computing resources without considering the problem of running environment, thus improving the efficiency of task scheduling (Fig. 1).

3.2 Linux Task Scripting Technology

In actual scenarios, it is often necessary to schedule a large number of tasks, including the selection of edge services, configuration of the operating environment and data collection, etc. Obviously, it is unrealistic to configure each task manually (Fig. 2).

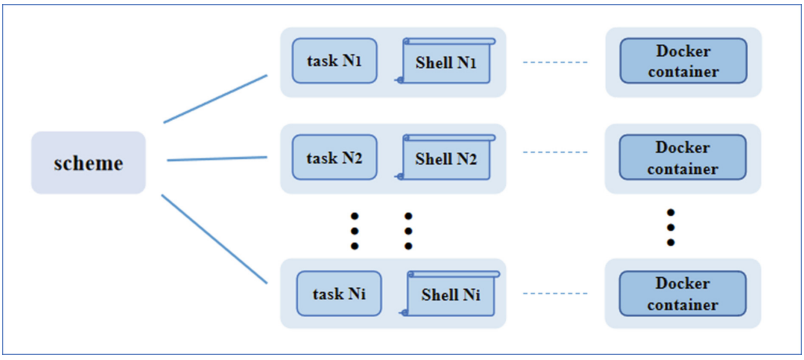

Fig. 2. Deployment principles of Shell task scripts

Therefore, in combination with the Docker technology introduced above, this paper installs the Docker environment for the edge server in advance. Based

on the Docker container and Linux scripting technology, the system generates specific task scripts for specific tasks. After finding the optimal allocation scheme, script automation is implemented to build the running environment of tasks and the execution operation of programs.

3.3 Genetic Algorithms

Multiple edge devices may be available when scheduling a task. The system hopes to improve resource utilization and reduce energy consumption as much as possible. Therefore, in the selection process, the link bandwidth of edge devices, the storage size of tasks, and the deployment and operation time after receiving tasks should be considered comprehensively. When there are multiple tasks to be assigned, we hope to take all factors into consideration and finally arrive at an optimal task assignment scheme. In order to solve this problem, genetic algorithm is used in this model.

Genetic algorithm (GA) is a global optimization adaptive probabilistic retrieval algorithm developed according to the mechanism of natural selection and genetic evolution, hoping to select the optimal population of individuals through many high-quality iterations. In this model, for a set of task assignment scheme, the optimal assignment scheme for each task to be assigned to each edge server should be given according to the type of task and parameters of edge server (Fig. 3).

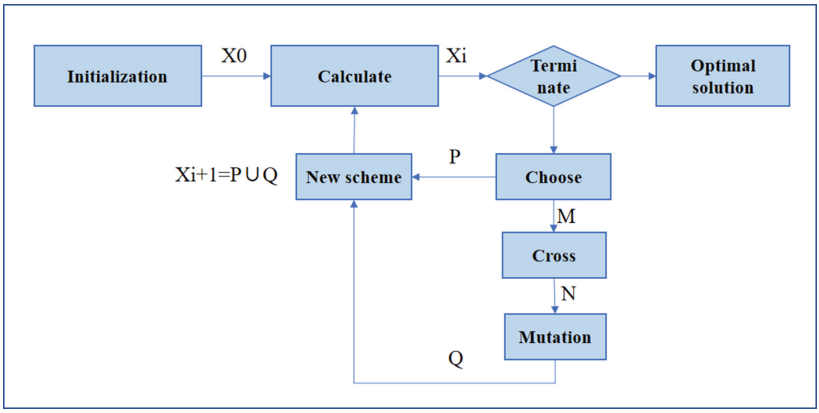

Fig. 3. Basic process diagram of genetic algorithm

4 A Cloud Side Task Scheduling Scheme Based on Task Deadline

This chapter first introduces the system model and application scenario, and then introduces the proposed comprehensive objective genetic algorithm scheme in detail.

4.1 System Model

The system model of this paper is shown in Fig. 4. The edge cloud in the figure refers to the computing and storage resources deployed around the decision server; The function of decision server is to collect the surrounding edge resource information, receive task information and generate task scheduling strategy. The decision server receives a certain range of task requests, and stores a certain number of tasks in the task queue within a period of time.

The tasks in the task queue may be executed in the cloud computing center or in the edge cloud. The task flow direction is mainly determined according to the specific requirements of the task.

The execution steps of the model task are as follows:

1. The decision server collects edge cloud device information.
2. The decision server receives the task information and generates a task queue.
3. Judge the task type according to the task information.
4. Generate a task scheduling scheme for the tasks assigned to the edge cloud.

The intelligent device gives the task execution conditions, task requirements and task execution script; After receiving the task, the decision server analyzes the task type, judges the task execution time, and finally determines whether the task is assigned to the cloud computing center or the edge cloud. If the task is judged to be assigned to the edge cloud for execution, the decision server schedules the task according to the task execution conditions and the current edge cloud state; After the task carries the execution script to the server, the server installs the task environment and executes the task according to the script. After completing the task, the edge cloud also returns the result through the decision server. And the cloud computing center can directly return the result to the intelligent device.

4.2 Task Classification Strategy Based on Task Information

After receiving the task information, the decision server simply classifies the task. See Table 1 for specific task types. In reality, due to the limited storage resources of mobile devices, large data contents are stored in cloud data centers. This paper defines such task as Scale Task. Considering the task execution deadline and measurement time, the task is divided into Immediate Task and Edge Timely Task. Immediate task is executed in the cloud computing center, while edge timely task is executed in the edge cloud. Measurement time refers

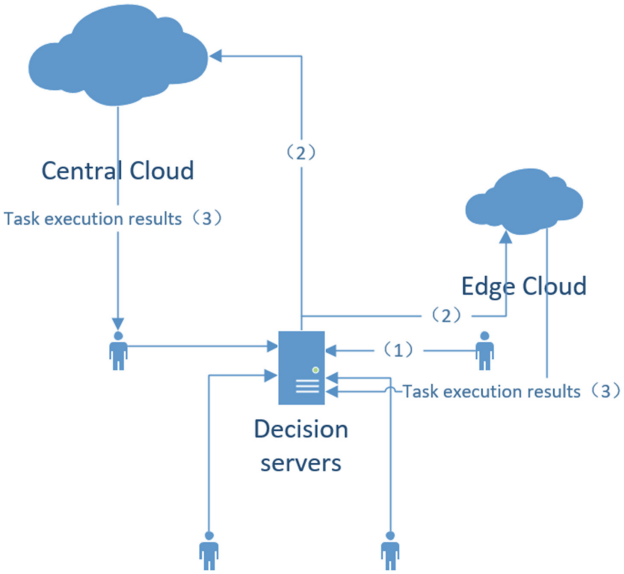

Fig. 4. Schematic diagram of system model

to the shortest time for the cloud computing center to perform tasks set by the model considering the network delay; The specific value is obtained by averaging the response speed of historical tasks under the LAN, and then making a fuzzy judgment incombination with the task size.

Table 1. Task classification

Task Name	Attribute
Scale task	The tasks consume a lot of storage and computing resources
Immediate task	The task deadline is greater than or equal to the measurement time
Edge timely task	The task deadline is less than measurement time

4.3 Edge Task Scheduling Scheme Based on Genetic Algorithm

The genetic algorithm considers the Task Size (TS), Task Deadline (TD), Task Number (TN) given by the edge timely task, and generates a set of task assignment scheme combined with the information of edge cloud devices at the current time. Edge cloud device information includes three parameters: CPU frequency, storage size and number of edge servers.

Construction of Multi-objective Task Scheduling Model
In this paper, multi-objective includes three parameters: task execution time, task execution energy consumption and storage utilization; Task execution time consists of task transmission time and task execution time; Task execution energy consumption is calculated by multiplying server power and task execution time, and storage utilization is the ratio of task volume to device storage space.

Edge Timely Task scheduling description: it is assumed that there are N edge timely tasks in the task queue of the decision server, represented by vector $P = \{P_1, P_2, P_3, \cdots, P_N\}$; There are M edge devices in the edge cloud, represented by vector $D = \{d_1, d_2, d_3, \cdots, d_M\}$. The scheduling scheme hopes to minimize the overall execution Time[P] and energy consumption Power[P] of N tasks, and maximize the utilization of edge device Resource[P] of N tasks. So the mathematical model of the objective function should be Eq. (1).

$$\text{Min Time [P](X)\&\& Min Power [P](X)\&\& Min Resource [P](X)} \tag{1}$$

X is the N × M matrix represents a task scheduling scheme, which is also a solution of the problem. The task L is executed on the edge server of M when element $x^{\mathrm{l}}_{\mathrm{m}} = 1$, and not executed on the edge server of M when element $x^{\mathrm{l}}_{\mathrm{m}} = 0$. In order to reduce the complexity, a task is only executed on one device by default in this paper, that is, only one element in a row of X matrix can have a value of 1 and the rest are 0.

$$\mathrm{X} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_M^1 \\ x_1^2 & x_2^2 & \cdots & x_M^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{\mathrm{N}} & x_2^{\mathrm{N}} & \cdots & x_M^{\mathrm{N}} \end{bmatrix} \tag{2}$$

The total execution time of a single task P_l on device d_m can be expressed as T^l_m. The total time consists of task transmission time and task execution time. The calculation formula is as follows:

$$T^{\mathrm{l}}_m = tt^{\mathrm{l}}_m + tc^{\mathrm{l}}_m \tag{3}$$

$$\mathrm{m} \in [0, \mathrm{M}] \tag{4}$$

$$l \in [0, N] \tag{5}$$

tt^1_mrepresents the time when task is uploaded to edge server m. TS_l represents the size of task l, BT^{l}_m represents the bandwidth of the upload link when task l is transmitted to edge server m, and d represents the interference of data during transmission. The calculation formula is as follows:

$$tt^1_m = \frac{TS_l * d}{BT^{\mathrm{l}}_m} \tag{6}$$

tc^{l}_mrepresents the running time of the task l on the edge server m, which is equal to the size of the task divided by the frequency of the edge server CPU, TS_l

represents the size of the task l, f_m represents the frequency of the edge server CPU, and the calculation formula is as follows:

$$tc_m^1 = \frac{TS_l}{f_m} \tag{7}$$

The calculation formula of T_m^l obtained from (3) (6) (7) is as follows:

$$T_m^l = \frac{TS_l * d}{BT_m^l} + \frac{TS_l}{f_m} \tag{8}$$

After calculating the execution time of each task on each edge server, it is necessary to limit T_m^l not to exceed the deadline $T_m^l$of task l. If it exceeds the deadline, it means that server does not have the ability to execute task l. If the execution time of the task l on all edge servers exceeds the deadline, the execution time of the task is deemed to have failed. Therefore, all task assignments need to meet the following constraints:

$$T_m^{\mathrm{l}} < TD_l \tag{9}$$

Finally, output the execution time of each task on each device. T matrix is used here. T is $\mathrm{N} \times \mathrm{M}$ matrix. The elements in the matrix comply with the constraints of Eq. (8), and the calculated time is filled in the specified position of the matrix. If not, the maximum value is filled in the corresponding position.

$$\mathrm{T} = \begin{bmatrix} T_1^1 & T_2^1 & \cdots & T_M^1 \\ T_1^2 & T_2^2 & \cdots & T_M^2 \\ \cdots & \cdots & \cdots & \cdots \\ T_1^{\mathrm{N}} & T_2^{\mathrm{N}} & \cdots & T_M^{\mathrm{N}} \end{bmatrix} \tag{10}$$

The resource utilization of a single task P_l on the device d_m can be expressed as M_m^{l}, and the calculation formula of the resource utilization is as follows, where D_m represents the overall storage space of the edge server m:

$$M_m^{\mathrm{l}} = \frac{TM_l}{D_m} \tag{11}$$

$$M_m^{\mathrm{l}} < 1 \tag{12}$$

After calculating the memory utilization of each edge server, the M matrix is used to store the utilization of each task on each edge server. M is $\mathrm{N} \times \mathrm{M}$ matrix. Each element in the M matrix needs to meet the condition that the value is less than 1 to ensure that the server can run this task.

$$M = \begin{bmatrix} M_1^1 & M_2^1 & \cdots & M_M^1 \\ M_1^2 & M_2^2 & \cdots & M_M^2 \\ \cdots & \cdots & \cdots & \cdots \\ M_1^{\mathrm{N}} & M_2^{\mathrm{N}} & \cdots & M_M^{\mathrm{N}} \end{bmatrix} \tag{13}$$

The task execution energy consumption C_m^{l} is related to the calculation power P_m and calculation time T_m^{l} of the edge server. The calculation formula of C_m^{l} is as follows:

$$C_m^{\mathrm{l}} = P_m * T_m^{\mathrm{l}} \tag{14}$$

The calculated power P_m of the edge server is related to the CPU frequency and k_m is a parameter related to the CPU structure. The calculated power P_m can be obtained from the following formula:

$$P_m = k_m * (f_m)^2 \tag{15}$$

After simplifying the task execution energy consumption in combination with formula (8) (14) (15), it is as follows:

$$C_m^{\mathrm{l}} = k_m * (f_m)^2 * \left(\frac{TS_{l^*} d}{BT_m^{\mathrm{l}}} + \frac{TS_l}{f_m} \right) \tag{16}$$

After calculating the execution energy consumption of each task by each edge server, the C matrix is used to store the corresponding execution energy consumption. C is N × M matrix.

$$C = \begin{bmatrix} C_1^1 & C_2^1 & \cdots & C_M^1 \\ C_1^2 & C_2^2 & \cdots & C_M^2 \\ \cdots & \cdots & \cdots & \cdots \\ C_1^{\mathrm{N}} & C_2^{\mathrm{N}} & \cdots & C_M^{\mathrm{N}} \end{bmatrix} \tag{17}$$

Multi-objective Conditional to Single Objective Scheduling Scheme
A good task scheduling scheme X has relatively low time cost, low energy consumption and high resource utilization. According to the above analysis, it can be found that the lowest time often brings high power consumption. In order to achieve a relatively good index for both of them and resource utilization, the following single-objective evaluation function is designed in this paper:

$$f(X) = C(X) * t_1 + \frac{1}{M(X)} * t_2 + T(X) * t_3 \tag{18}$$

t_1,t_2,t_3 respectively represent the weights of three different parameters. Different values can be set considering different scene requirements.$C(X)$,$M(X)$,$T(X)$ represent the total value of the three parameters under scheme X. According to the requirements of Eq. (1), the objective function of the system can be obtained:

$$\mathrm{Min} f(X) \tag{19}$$

The generation steps of single objective scheduling scheme are as follows:

1. The random algorithm is used to generate s sets of task scheduling schemes satisfying the constraints.
2. The evaluation values of all schemes are calculated according to the single objective evaluation function.
3. Sort the evaluation values from small to large. Combined with the objective function, we can know that the scheme with small evaluation value is more in line with the expectation of the model.
4. Select s schemes from the initial s schemes according to the evaluation value, and the same scheme can be selected repeatedly.

Algorithm 1. Crossover operator algorithm

Require: Father matrix,Mother matrix
Ensure: Child matrix
Calculate f(father),f(mother)
Father=Min(f(father),f(mother))
Mother=Max(f(father),f(mother))
i=0
while $i < task\_num/2 + 1$ **do**
    Child[i]=father[i]
    i=i+1
end while
while $i < task_num$ **do**
    Child[i]=mother[i]
end while
return Child;

Algorithm 2. Mutation operator algorithm

Require: Child matrix
Ensure: Child matrix
x =random(task)
y =random(device)
Child[x][y]= 1
i=0
while $i < device\_num$ **do**
    if $i == y$ **then**
        i=i+1
    end if
    Child[x][i]=0
    i=i+1
end while
return Child;

5. Cross judge the selected s set of schemes. If it fails, keep it unchanged and cross it successfully. Crossover operation, crossover operator is given in algorithm 1. After the population crossover is completed, s sets of schemes are also generated.
6. According to the mutation probability set and the mutation operator algorithm given in algorithm 2, the mutation operation is performed on the s schemes.
7. View evolutionary algebra
   (a) If the set evolutionary algebra is reached, the scheme with the minimum evaluation value is selected as the final scheme from s schemes.
   (b) If the set evolutionary algebra is not reached, go back to the first step and continue (Fig. 5).

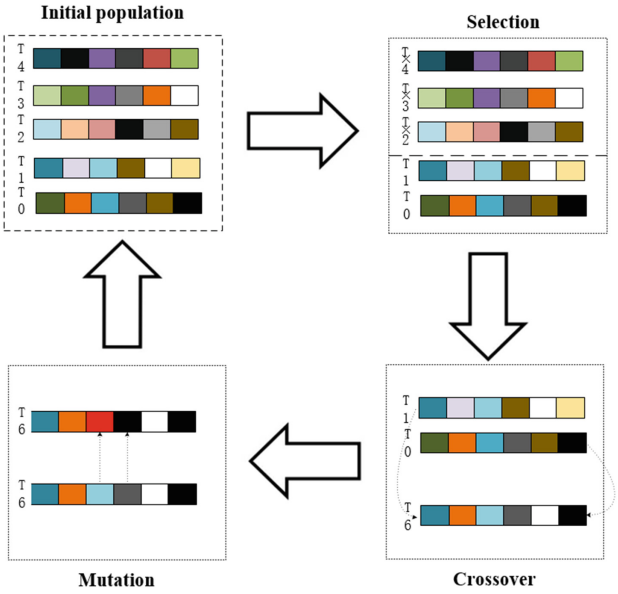

Fig. 5. Schematic diagram of population evolution

5 Simulation Experiment and Result Analysis

In this chapter, by comparing different evolutionary algebras, we find out the most suitable evolutionary algebra as the value of model evolution algebra. On the other hand, the performance differences between single target and comprehensive target in time, energy consumption and resource utilization are compared to prove the advantages of this model.

5.1 Experimental Setup

All experiments in this section are completed on laptops running windows10 system, and the specific configuration is as follows:

Intel (R) Core (TM) i7-8550UCPU@1.80 GHz processor, NVIDIA GeForce 940MX core graphics card with 8 GB of running memory. All the experimental code is written in Python language. Manual generation of 100 edge servers, a dataset containing 10 tasks and a dataset containing 20 tasks. The edge server parameters contain server cpu frequency, server storage space size. Task data contains task volume size, task deadline. By processing data to find out the edge and timely tasks, the task scheduling scheme is obtained by using genetic algorithms. Comparison algorithm:

1. Genetic algorithms of different evolutionary algebras: In this paper, evolutionary algebras were set as 10, 50, 100 and 150 respectively.

2. Genetic algorithms with different evaluation functions: Time first genetic algorithm, power consumption first genetic algorithm, storage resource utilization first genetic algorithm, comprehensive objective genetic algorithm.
3. Min-min algorithm: find the server with the smallest running time in the edge cloud to execute based on the order of the tasks in the task queue.

Comparison parameters:

1. Time: the average cost time of each task in the task scheduling scheme.
2. Energy consumption: the average energy consumption of each task in task scheduling scheme.
3. Storage usage: the average storage usage of each task in a task scheduling scheme (the amount of task computation compared to the storage size of the server).

5.2 Experimental Results and Analysis

As shown in Fig. 6, 7 and 8, For the same number of evolutionary generations, the task scheduling solution generated by the time-first genetic algorithm has the lowest time cost and the task scheduling solution generated by the power-first genetic algorithm has the lowest energy cost. The storage utilisation-first genetic algorithm generates the best utilisation solution. This is because time, energy consumption and storage utilisation are evaluated as separate measures of population evolution, so that the population always moves towards the optimal one for each of them.

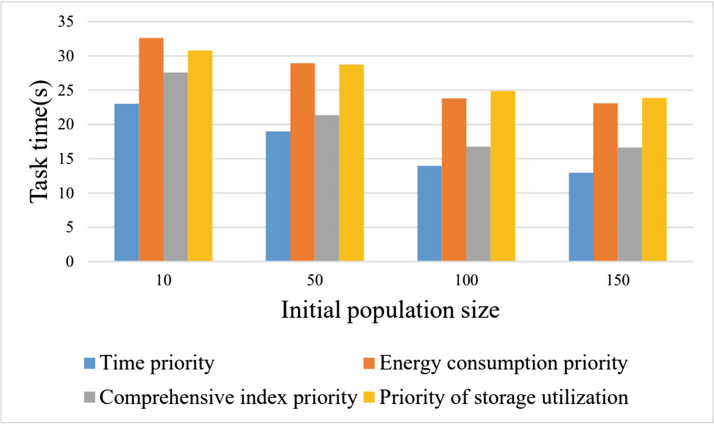

Fig. 6. The time cost of four genetic algorithms

It can be found from the information in the analysis figure that the performance of the comprehensive index cannot reach the optimal value in any evaluation direction, but it is always the sub-optimal value in all evaluation directions.

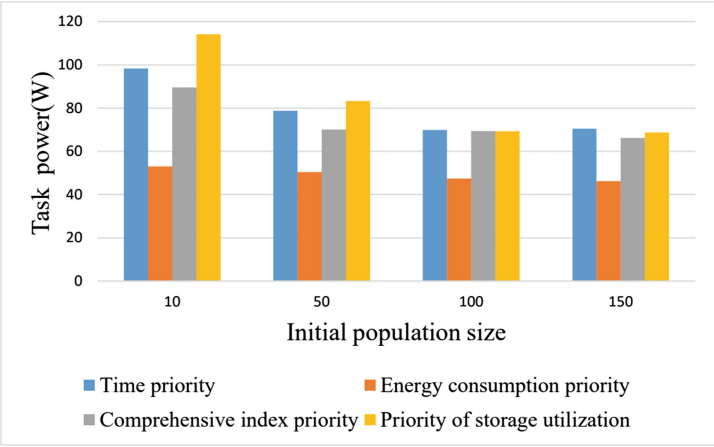

Fig. 7. Task power consumption of four genetic algorithms

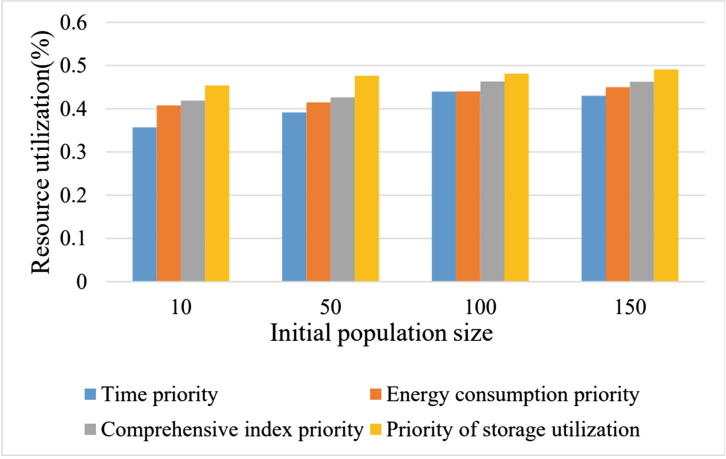

Fig. 8. Storage utilization of four genetic algorithms

For example, from the perspective of time, the comprehensive objective genetic algorithm is only smaller than the time-first algorithm but larger than the other two genetic algorithms. This is because the comprehensive index is the result of making a relative choice in three index directions.

For the whole edge cloud, the comprehensive evaluation index is often more helpful to the performance of the whole cloud.

Analysis of the information in Fig. 9 shows that as the number of evolutionary generations increases, the final solution performs better and better in time, while the decline in time decreases as the number of evolutionary generations increases. When the evolutionary generation is set to 100 and 150 respectively,

the final solution no longer changes much in time. With the same task scenario and the same evaluation index, there must be a perfect final individual, and the increase in evolutionary generation can help the population to approach the final individual indefinitely. When the initial test population evolves to 100 generations, the population is already close to the final solution. Therefore, the population will not change much if it continues to evolve.

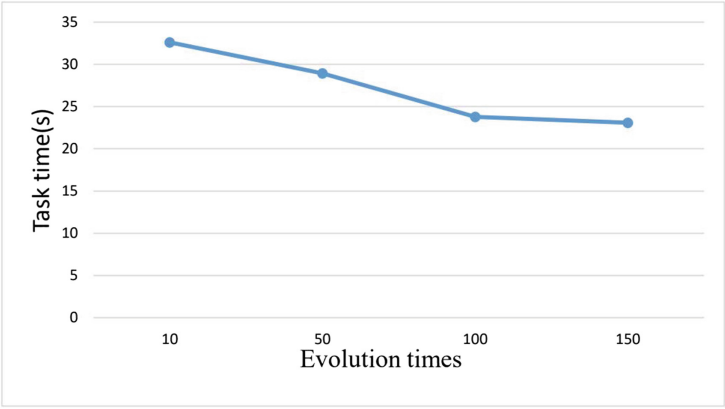

Fig. 9. Time cost of synthetic objective genetic algorithm

The analysis in Fig. 9 reveals that the genetic algorithm can deduce excellent task allocation schemes when the evolutionary generation is set to 100. Therefore, when comparing with the min-min algorithm, the evolutionary generation of the genetic algorithm is set to 100. In order to discover the advantages of the integrated genetic algorithm in dealing with complex situations. In this paper, two task sets, named task1 and task2, are set up, containing 10 and 20 tasks respectively.

Analysis of the information in Fig. 10 shows that the time-first genetic algorithm outperforms the min-min algorithm for either a task count of 10 or 20. For a task count of 10, the min-min algorithm outperforms the integrated goal genetic algorithm, and for a task count of 20, the results are reversed. The time-first genetic algorithm performs best as it finds the task allocation solution in the context of the whole population. When the number of tasks is 10, the min-min algorithm outperforms the integrated goal genetic algorithm because the number of tasks is much larger than the number of devices. As the number of tasks increases, the min-min algorithm's effectiveness drops and the integrated goal genetic algorithm outperforms the min-min algorithm.

In summary, the integrated goal genetic algorithm outperforms the power-first genetic algorithm and the storage resource utilisation-first genetic algorithm in terms of time; it outperforms the time-first and storage resource utilisation-first genetic algorithm in terms of energy consumption; and it outperforms the

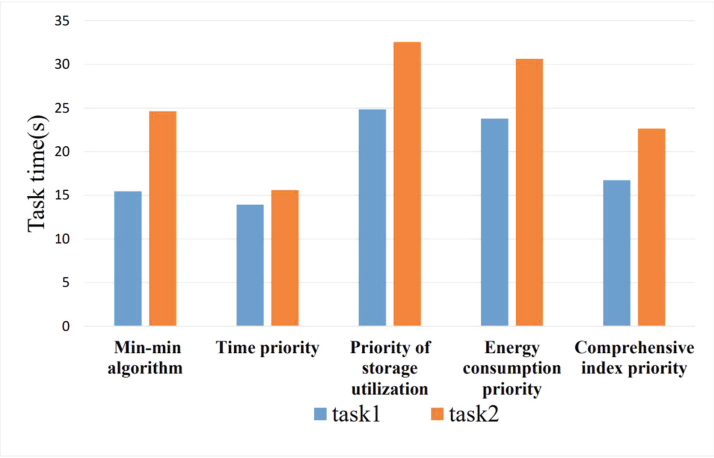

Fig. 10. Time cost of the mim-mim algorithm and the four genetic algorithms

time-first and power-first algorithm in terms of storage resource utilisation; and the value of a reasonable evolutionary algebra is found for the experimental data. In a comparison with the min-min algorithm, it was found that the integrated goal genetic algorithm could handle more complex situations.

6 Conclusion

In order to improve the efficiency of task scheduling and the overall resource utilization rate of edge cloud, a task scheduling model based on comprehensive objective genetic algorithm is proposed in this paper. Firstly, simple tasks are performed according to the task information, and then the edge cloud task scheduling scheme is generated by using comprehensive objective genetic algorithm for edge timely tasks. Other types of tasks are assigned to cloud computing centers. In order to improve the execution efficiency of the task, docker container technology and Linux script technology are combined to deploy and execute the task environment. Finally, the performance of the comprehensive objective genetic algorithm is verified in the simulation environment. The experimental results show that the comprehensive objective genetic algorithm can achieve good performance in the three dimensions of time, energy consumption and storage utilization.

References

1. He, J., Sun, G.: Multi-objective task scheduling based on cuckoo particle swarm optimization algorithm. Inf. Technol. **44**(5), 37–40 (2020)
2. Tian, Y., Huang, Z., Zhang, Y.: A survey of task scheduling methods in cloud computing environment. Comput. Eng. Appl. **57**(2), 1–11 (2021)

3. Wang, Q.: Application of meta-heuristic algorithm in discrete location selection. Nanjing University of Aeronautics and Astronautics (2010)
4. Zhang, M., Li, H., Liu, L., et al.: An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in clouds. Distrib. Parallel Databases **36**(2), 339–368 (2018)
5. Yuan, H., Bi, J., Tan, W., et al.: TTSA: an effective scheduling approach for delay bounded tasks in hybrid clouds. IEEE Trans. Cybern. **47**(11), 3658–3668 (2016)
6. Krishnaveni, H., Sinthu Janita Prakash, V.: Execution time based sufferage algorithm for static task scheduling in cloud. In: Peter, J.D., Alavi, A.H., Javadi, B. (eds.) Advances in Big Data and Cloud Computing. AISC, vol. 750, pp. 61–70. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-1882-5_5
7. Shishido, H.Y., Estrella, J.C., Toledo, C.F.M., et al.: Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. Comput. Electr. Eng. **69**, 378–394 (2018)
8. Gomathi, B., Krishnasamy, K., Balaji, B.S.: Epsilon-fuzzy dominance sort-based composite discrete artificial bee colony optimisation for multi-objective cloud task scheduling problem. Int. J. Bus. Intell. Data Min. **13**(1–3), 247–266 (2018)
9. Kaur, M., Kadam, S.: A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling. Appl. Soft Comput. **66**, 183–195 (2018)
10. Huang, W., Xin, F., Huang, Y.: Multi-objective task scheduling in cloud computing based on chaotic cat swarm algorithm. Microelectron. Comput. **36**(6), 55–59 (2019)
11. Li, H.: Cloud computing task scheduling strategy based on improved moth optimization algorithm. J. Taiyuan Univ. (Nat. Sci. Ed.) **38**(1), 61–67 (2020)
12. Chen, X., et al.: Augmented queue-based transmission and transcoding optimization for livecast services based on cloud-edge-crowd integration. IEEE Trans. Circuits Syst. Video Technol. **31**(11), 4470–4484 (2020)
13. He, J.-Y., Sun, Q.-K.: Cuckoo particle swarm optimization algorithm for multi-objective task scheduling. Inf. Technol. **44**(5), 37–40 (2020)
14. Wang, L., Wu, C., Fan, W.: A review of resource allocation and task scheduling optimization for edge computing. J. Syst. Simul. **33**(3), 509 (2021)
15. Zhao, X., Zhao, Y., Li, B., et al.: A delay- and energy-aware approach to edge server placement. Comput. Eng. (2021)
16. Tian, J.J., Huang, Z., Zhang, Y.: A review of task scheduling methods for cloud computing environments. Comput. Eng. Appl. **57**(2), 1–11 (2021)
17. Nardini, G., Stea, G., Virdis, A.: A low-latency and reliable multihop D2D transmissions scheduling algorithm for guaranteed message dissemination. Ad Hoc Netw. **126**, 102755 (2022)
18. Priya, V., Kumar, C.S., Kannan, R.: Resource scheduling algorithm with load balancing for cloud service provisioning. Appl. Soft Comput. **76**, 416–424 (2019)
19. Lin, Y., Song, H., Ke, F., et al.: Optimal caching scheme in D2D networks with multiple robot helpers. Comput. Commun. **181**, 132–142 (2022)
20. Zhao, H., Bai, K., Cui, B., Han, L., Ma, Y.: Research on the key path of enterprise-level data warehouse construction based on DAMT. J. Jiangxi Normal Univ. (Nat. Sci. Ed.) **42**(06), 634–638 (2018). https://doi.org/10.16357/j.cnki.issn1000-5862.2018.06.15