Mathematical Biosciences and Engineering

MBE, 19(9): 9147–9167.
DOI: 10.3934/mbe.2022425
Received: 29 April 2022
Revised: 29 May 2022
Accepted: 06 June 2022
Published: 22 June 2022

http://www.aimspress.com/journal/MBE

Research article

Resources allocation optimization algorithm based on the comprehensive utility in edge computing applications

Yanpei Liu*, Yunjing Zhu, Yanru Bin and Ningning Chen

School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China

* **Correspondence:** Email: liuyanpei@zzuli.edu.cn; Tel: +8618339810322.

Abstract: In the mobile edge computing environment, aiming at the problems of few classifications of resource nodes and low resource utilization in the process of multi-user and multi-server resource allocation, a resource optimization algorithm based on comprehensive utility is proposed. First, the algorithm improves the Naive Bayes algorithm, obtains the conditional probabilities of job types based on the established Naive Bayes formula and calculates the posterior probabilities of different job types under specific conditions. Second, the classification method of resource service nodes is designed. According to the resource utilization rate of the CPU and I/O, the resource service nodes are divided into CPU main resources and I/O main resources. Finally, the resource allocation based on comprehensive utility is considered. According to three factors, resource location, task priority and network transmission cost, the matching computing resource nodes are allocated to the job, and the optimal solution of matching job and resource nodes is obtained by the weighted bipartite graph method. The experimental results show that, compared with similar resource optimization algorithms, this method can effectively classify job types and resource service nodes, reduce resource occupancy rate and improve resource utilization rate.

Keywords: edge computing; resource allocation; improved Naive Bayes algorithm; resource service node classification; weighted bipartite graph

1. Introduction

According to Ericsson's "Mobile Market Report", it is predicted that by the end of 2025, there will be more than 2.8 billion 5G subscribers worldwide. 5G will cover nearly 65% of the world's population, and 5G networks will carry 45% of data traffic. Among that, smartphones will account for 86%, and the world will usher in the Internet era of the explosive growth of network data and information. The increasing popularity of 5G networks has witnessed the rapid development of the Internet of Things [1], which has stimulated the emergence of various forms of new applications, such as AR, VR, autonomous driving, smart cities, telemedicine and other technical fields. These application scenarios require a higher bandwidth rate and lower system energy consumption and computing delay [2,3], while the centralized processing mode adopted by the mobile cloud computing model is far from the terminal devices and cannot meet the daily needs of users [4,5]. Therefore, mobile edge computing (MEC) technology has emerged.

Mobile edge computing [6] is considered one of the core technologies in developing modern communication networks and 5G technology. MEC performs operations such as computing processing and resource allocation on edge nodes by introducing computing data to the edge of the mobile network. As a research hotspot of MEC, resource allocation mainly studies the problem of where to unload computing tasks [7]. In the MEC network scenario of computing-intensive applications, tasks need to be reasonably allocated to maximize the utilization rate of each computing node's storage, CPU and other resources. Therefore, effective resource allocation can not only avoid channel interference between tasks but also increase the calculation rate of tasks, effectively reducing resource occupancy rate and improving resource utilization rate [8,9]. Although a large number of excellent works are devoted to the related research of resource allocation in the MEC environment, numerous studies cover resource allocation based on joint computing offload, channel allocation, spectrum allocation, power allocation and other technologies; and most of the optimization objectives are offload delay, system energy consumption or user income. Thus, there is little research on dynamic resource allocation under multi-user and multiple MEC servers. Therefore, this paper studies the dynamic resource allocation problem in the mobile edge environment and proposes a resource allocation optimization method based on comprehensive utility in the MEC (RAOCU). The method mainly includes three parts: job classification based on an improved Naive Bayes algorithm, resource service node classification based on resource utilization and resource allocation based on comprehensive utility. The main, specific contributions of this work are summarized as follows.

1) Aiming at the problem of inaccurate job classification results caused by the underflow of the Naive Bayes algorithm, the Naive Bayes algorithm is optimized. The conditional probability of the job type is obtained according to the established Naive Bayes formula, and then the posterior probability that the running job is a CPU-intensive job and I/O-intensive job under certain conditions is obtained, which improves the Naive Bayes Classifier performance.
2) For resource allocation based on comprehensive utility, by considering the three factors of resource location, task priority and network transmission cost, matching computing resources are assigned to the particular types of jobs, and according to the weighted bipartite graph, the optimal solution of matching job and resource nodes is obtained.
3) The experimental results demonstrate that, compared with other algorithms, the algorithm of this paper can classify job types and resource service nodes more effectively, reduce resource occupancy rate and improve resource utilization rate.

The rest of the paper is organized as follows: Section 2 is the research status, and Section 3 expounds on and analyzes the design of resource optimization methods based on comprehensive utility in an MEC environment, including job classification based on an improved Naive Bayes algorithm, resource service node classification based on resource utilization rate and resource allocation based on comprehensive utility. Section 4 provides a detailed description of the proposed algorithm. Section 5 provides a comparison and analysis of experimental results. Finally, Section 6 summarizes the paper and discusses its development direction.

2. Related work

Many scholars have conducted relevant research on resource allocation in the MEC environment. Tran et al. [10] studied joint task offloading and resource allocation, aiming at maximizing the revenue of user task offloading, and the total revenue of the user is characterized as the weighted sum of task execution delay and energy consumption. Xu et al. [11] established a comprehensive model, Zenith, to capture the optimization problem of online resource allocation in the edge cloud. Based on the proposed model, an auction-based resource contract mechanism and a delay scheduling technology to maximize the utility of service providers were proposed. Dab et al. [12] proposed a joint task allocation and resource allocation method aiming at minimizing the energy consumption of mobile terminals. This scheme formulated the optimization problem based on integer programming and proposed an enhancement algorithm considering the waiting delay of mobile terminals and data transmission. Jinke et al. [13] studied the joint communication and computing resource allocation of multi-user multiple access communication systems, established a function with the delay and energy consumption of mobile devices as the optimization objectives and proposed a resource allocation scheme based on a subgradient algorithm. You et al. [14] discussed the resource allocation problem of minimum energy consumption in multi-user multiple access systems, defined the average unloading priority function and described the resource allocation as a convex optimization problem to minimize the weighted sum of system energy consumption under the constraint of calculation execution delay. Sardellitti et al. [15] considered a multi-cell mobile edge computing offload system, which jointly allocates radio and computing resources to minimize the total energy consumption of mobile terminals under the constraint of offload delay. Ketyko et al. [16] studied the offloading decision with the goal of maximizing the number of service applications and assigning computing nodes by priority. Wang et al. [17] used the method of deep learning to allocate resources and proposed a dynamic unloading scheme. Lemaréchal et al. [18] proposed a layered MEC deployment architecture when MEC computing resources are limited. Boyd et al. [19] proposed three different cloud selection strategies to optimize delay, total cluster energy consumption and energy consumption of each SCeNB in the cluster. Liu et al. [20] used the knapsack model to optimize the whole resource allocation and load balancing problem. Jabeen et al. [21] proposed an interference management scheme, which allocates communication resources and computing resources under the condition of minimum interference. Avgeris et al. [22] proposed an optimal resource allocation framework leveraging the amalgamation of the edge resources. A mechanism based on Markov Random Fields was introduced to allocate redundant workload. To speed up the learning and reduce the resource consumption of the network, Zuo et al. [23] formulated a problem of joint transmission time allocation, computing frequency control and user selection. Fan et al. [24] modeled the resource allocation and pricing of a cloud/edge computing service provider (CESP) as a mixed-integer programming problem (MIP) with the goal of optimizing the revenue of

the CESP, and an efficient resource allocation and pricing algorithm based on iterative greed and search was proposed. AlQerm et al. [25] proposed a novel resource allocation model that aimed to maximize the IoT applications' utilities considering multiple applications' priorities and various delay requirements and guaranteed resource allocation fairness. In order to solve the multi-user resource allocation problem, Huang et al. [26] adopted a performance-aware resource allocation (PARA) scheme based on a depth deterministic policy gradient (DDPG) to derive the optimal resource allocation strategy. Lin et al. [27] formulated the upstream resource allocation as a stratified multi-objective optimization model, adjusting the spectrum and storage allocation between latency-critical and delay-tolerant flows. Zhu et al. [28] formulated the virtual resource allocation strategy as an optimization problem that aims to maximize the revenue earned by mobile virtual network operators and proposed a distributed virtual resource allocation algorithm based on the alternating direction method of multipliers. Shabbir et al. [29] compared other commonly used algorithms against the health information security at the MCC environment in terms of better performance and auxiliary qualitative security-ensuring measures.

Although the literature mentioned above on resource allocation optimization has made some achievements, in the multi-user and multi-MEC server environment, the classification of resource nodes and the low utilization rate of resources in resource allocation are seldom considered. Therefore, this paper proposes a resource optimization algorithm based on comprehensive utility in the MEC environment, and the dynamic resource allocation problem in the mobile edge environment is studied.

3. Design of resource allocation optimization algorithm based on comprehensive utility in MEC environment

The resource allocation optimization method based on comprehensive utility in the MEC environment proposed in this paper mainly includes job classification based on an improved Naive Bayesian algorithm, classification of resource service nodes based on resource utilization rate and resource allocation based on comprehensive utility. The main parameters involved and their meanings are shown in Table 1.

3.1. Job classification based on improved Naive Bayesian algorithm

In the edge computing environment, by deploying edge computing nodes at the edge of the network, various mobile applications and new application scenarios can provide users with various services, increasing the number and types of jobs that edge computing nodes need to handle exponentially. Therefore, different types of jobs will generate different workloads on the cluster, including I/O-bound or CPU-bound workloads. In the research of job classification, jobs are mostly divided into I/O type or CPU type. I/O-intensive jobs are allocated to I/O type resources. CPU-intensive jobs are allocated to CPU type resources to achieve the load balance of cluster nodes and reduce the response delay of task allocation in the edge environment. Job classification based on job priority or resource requirements will also affect job resource allocation for most applications. The job controller uses the priority and the load indexes to establish the Bayes' theorem's conditional probability.

The jobs are divided into I/O type or CPU type according to the characteristics of the jobs. The ratio of input data (MID) to output data (MOD) is represented as $\rho=MOD / MID$, MCD denotes Map

processing data, SOD indicates the output result of Shuffle, RID shows the output data of Reduce, $MTCT$ represents the completion time of the Map task, $DIOR$ illustrates the I/O rate of the disk, and N illustrates the number of tasks. Therefore, I/O type and CPU type tasks can be defined as follows:

$$\frac{N(MID+MOD+\mathrm{SOD}+\mathrm{SID})}{MTCT}=\frac{N((1+2\rho)MID+SID)}{MTCT}\geq DIOR. \tag{1}$$

$$\frac{N(MID+MOD+SOD+SID)}{MTCT}=\frac{N((1+2\rho)MID+SID)}{MTCT}<DIOR. \tag{2}$$

Table 1. The meanings of main parameters.

Parameter	Definition
$MICT$	Completion time of Map task
$DIOR$	Disk I/O rate
N	Number of tasks
job_z	job z
T_{CPU}^{last}	Last cumulative CPU cycle
$T_{CPU}^{cumulative}$	The actual CPU cycle used to execute the job is the cumulative CPU cycle
$T_{CPU}^{current}$	Current CPU cycle
$T_{CPU}^{lastcurrent}$	Historical CPU cycle and CPU cycle at the last heartbeat
$U_{i/o}^{read}$	The amount of reads that the job accumulates in I/O
$U_{i/o}^{write}$	The cumulative output of the job in I/O
v_h^z	The amount of calculation required to complete the task h_z
s_h^z	The amount of memory required to complete the task h_z
J	Resource pool set
v_j	Computing power
s_j	Memory size of resource pool j
$y(z)$	Priority of job z
y_h^z	Priority of resources required for task h_z
l_e	Location of the local server
l_o	Location of required resources
δ	Quantity of content required
w_t	Size of the t-th required content
$\psi(h_z, j)$	Weight between task h_z and container u_r

If the sum of the resource utilization rates of the five stages of MapReduce (MID, MOD, MCD, SOD and RID) is greater than or equal to the disk I/O utilization rate, the job is a CPU type job; otherwise, it is an I/O type of job. According to Eqs (1) and (2), it can be seen that the categories that affect the job include input data (MID), output data (MOD), Shuffle output result (SOD), Reduce output data (RID) and ρ factors, and among them, the values of some factors need to be determined after the job is executed. This paper needs to classify jobs before they are executed, so Eqs (1) and (2) cannot be used to directly judge the types of work. Therefore, this paper uses the Bayesian classifier to classify jobs in Hadoop. According to job-related features and node characteristics, the Bayesian classifier is adopted, and its input data is expressed as job characteristics and node characteristics.

According to job characteristics and node characteristics, the Bayesian classifier divides jobs into I/O-intensive and CPU-intensive jobs.

Equations (3) and (4) calculate the conditional probability that the running job is I/O-intensive and CPU-intensive under certain specific conditions.

$$P\left(job_z = CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right) \tag{3}$$

$$P\left(job_z = I/O|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right) \tag{4}$$

job_z represents job z, and X_n indicates job characteristics and node characteristics, where it is assumed that each X_n is independent of each other. Taking CPU-intensive jobs as an example, $P(B|A)=P(AB)/P(A)$ can be acquired according to the Bayesian formula, as shown in Eq (5).

$$\rho\ P\left(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)=\frac{P\left(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n|job_z=CPU\right)P\left(job_z=CPU\right)}{P\left(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)} \tag{5}$$

Equation (6) can be derived from Eq (5).

$$P\left(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n|job_z{=}CPU\right)=\prod_1^n P\left(\mathrm{X}_n|job_z{=}CPU\right) \tag{6}$$

Since there is no correlation between $P(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)$ and job attributes, it can be ignored. Thus, Eq (6) can be converted to Eq (7).

$$P\left(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)=P\left(job_z{=}CPU\right)\prod_1^n P\left(\mathrm{X}_n|job_z{=}CPU\right) \tag{7}$$

As shown in Eqs (5)–(7), the posterior probability $P(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)$ can be obtained by being given $job_z{=}CPU$ under the premise of the conditional independence assumption, which is expressed as Eq (8).

$$P\left(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)=\frac{P\left(job_z{=}CPU\right)\prod_1^n P\left(\mathrm{X}_n|job_z{=}CPU\right)}{P\left(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)} \tag{8}$$

Owing to the value of $P(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)$ being constant, it is only necessary to calculate the relative size of the numerator in Eq (8), which can be expressed as Eq (9).

$$\left(job_z{=}CPU\right)_{max}=\mathrm{argmax}P\left(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)=arg\,max\,P\left(job_z{=}CPU\right)\prod_1^n P\left(\mathrm{X}_n|job_z{=}CPU\right) \tag{9}$$

Based on Eq (9), when the posterior probability $P(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)$ is contrasted, more conditional probabilities are required for multiplication calculation. At the same time, it is likely that underflow will occur, which will lead to the abnormal execution of the next command, thus leading to the uncertainty result of a posterior probability $P(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)$, thereby affecting the judgment of job type, influencing the feasibility of the algorithm and reducing the performance of the algorithm. Consequently, this paper optimizes according to the disadvantages of the Naive Bayes algorithm, and Eq (9) is transformed into the following:

$$\left(job_z{=}CPU\right)_{max}=\mathrm{argmax}P\left(job_z{=}CPU|\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n\right)=InP\left(job_z{=}CPU\right)+\sum_1^n InP\left(\mathrm{X}_n|job_z{=}CPU\right). \tag{10}$$

It can be seen from Eq (10) that there may be a value of 0 on the right side of the equation, which will have a certain impact on the job classification result, thus affecting the performance of the algorithm. The Laplace smoothing technique can effectively avoid the situation where the right side of the equation is 0, and we can add 1 to the right side of the equation. Therefore, the class conditional probability of I/O-intensive jobs and CPU-intensive jobs can be expressed as

$$P(\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n|job_z{=}CPU)=\frac{T_k+1}{\sum_k T_k+T_c}. \tag{11}$$

In Eq (11), T_k denotes the number of occurrences of feature items in I/O-intensive jobs and CPU-intensive jobs in the total workload, the total number of all feature items of a job type, and T_c illustrates the smoothing factor, the total number of feature items of all job types. Therefore, the posterior probability for the two types of operations can be formulated as

$$P(job_z{=}CPU\,|\,\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)=arg\,max\,P(job_z{=}CPU)\prod_1^n\frac{T_k+1}{\sum_k T_k+T_c}. \tag{12}$$

$$P(job_z{=}I/O\,|\,\mathrm{X}_1,\mathrm{X}_2,\mathrm{K},\mathrm{X}_n)=arg\,max\,P(job_z{=}I/O)\prod_1^n\frac{T_k+1}{\sum_k T_k+T_c}. \tag{13}$$

The job type is judged according to the established Naive Bayes posterior probability. Therefore, when $P(job_z=CPU|\mathrm{X}_1,\mathrm{K},\mathrm{X}_n)>P(job_z=I/O|\mathrm{X}_1,\mathrm{K},\mathrm{X}_n)$, the job is CPU-intensive, and when $P(job_z=I/O|\mathrm{X}_1,\mathrm{K},\mathrm{X}_n)>P(job_z=CPU|\mathrm{X}_1,\mathrm{K},\mathrm{X}_n)$, the job is I/O-intensive. The job classification algorithm based on improved Naive Bayes is shown in Algorithm 4.1.

3.2. Resource service nodes are classified based on resource utilization

In the problem of resource allocation in the edge cloud environment, not only the types of jobs and the arrival times and regularity of jobs should be considered, but also the requirements for resource types of jobs should be considered. Therefore, according to the resource utilization rates of I/O and CPU, the resource service nodes are divided into I/O main resource and CPU main resource to make full use of the system resources.

TaskTracker sends heartbeat information to JobTracker within a specific time to indicate that the node is running. Therefore, I/O and CPU usages on TaskTracker are captured by adding specific indicators. According to the heartbeat message received from TaskTracker, JobTracker obtains the resource utilization rate information of the node. There, I/O and CPU resource utilization rates in TaskTracker are achieved as follows:

$$CPU_{usage}=\frac{T_{CPU}^{cumulative}-T_{CPU}^{last}}{\left(T_{CPU}^{current}-T_{CPU}^{lastcurrent}\right)*Num\_of\_processors}. \tag{14}$$

$$I/O_{usage}=\frac{\left(U_{i/o}^{read}-U_{i/o}^{lastread}\right)+\left(U_{i/o}^{write}-U_{i/o}^{lastwrite}\right)}{\left(T_{CPU}^{current}-T_{CPU}^{lastcurrent}\right)}. \tag{15}$$

In Eq (14), $T_{CPU}^{cumulative}$ shows the actual CPU cycle used to execute the job, the cumulative CPU cycle, T_{CPU}^{last} indicates the last cumulative CPU cycle, $T_{CPU}^{current}$ denotes the current CPU cycle, and the value of $T_{CPU}^{lastcurrent}$ manifests the historical CPU cycle and the CPU cycle at the last heartbeat. $U_{i/o}^{read}$ indicates the cumulative read amount of the job in I/O, and $U_{i/o}^{write}$ shows the cumulative output of the job in I/O. When $CPU_{usage} > I/O_{usage}$, the resource node is the main CPU resource, and when $CPU_{usage} \le I/O_{usage}$, the resource node is the main I/O resource. The classification algorithm of resource service nodes is shown in Algorithm 4.2.

3.3. Resource allocation based on comprehensive utility

In this section, resource nodes are calculated for job allocation according to three factors: resource location, task priority and network transmission cost. The optimal solution of job and resource node matching is obtained according to the method of a weighted bipartite graph.

The job is denoted as z, each job $z(0 \le z \le f)$ is composed of multiple tasks l_z, and each task is indivisible, independent and non-preemptive. The task h_z is illustrated as $h_z=(v_h^z, s_h^z)(1 \le h_z \le l_z)$, v_h^z represents the amount of computation required to complete task h_z, and s_h^z illustrates the memory size required to complete task h_z. The resource pool set is defined as $j(0 \le j \le J)$, and each resource pool is indicated as $j@(v_j, s_j)$, where v_j is the computing power of the CPU, and s_j is the memory size of the resource pool j. Therefore, the similarity between task h_z and resource service node j can be defined as follows. $u_j^{'}$ is the transpose of vector u_j.

$$Sim(h_z, j) = \frac{h_z \cdot j^{'}}{|h_z||j|} \tag{16}$$

The priority $y(h_z)$ of task h_z is expressed as

$$y(h_z) = \eta y(z) - (1-\eta) y_h^z . \tag{17}$$

In Eq (17), $y(z)$ represents the priority of job z, which is determined by the scheduling priority in the data processing system, y_h^z denotes the priority of resources required by task h_z, and η indicates the weight coefficient. As the value of $y(h_z)$ increases, the priority of task h_z will also increase, and y_h^z is determined by the location of resources required by task h_z. When the required resources are stored in the memory on the local server, set $y_h^z=2$. When the demand resources are stored in the disk on the local server $y_h^z=1$, and otherwise, $y_h^z = \frac{1}{h(l_e, l_o)}$. l_e represents the location of the local server, l_o illustrates the location of the required resources, and $h(l_e, l_o)$ describes the minimum network distance between the local server and the required resources.

The network transmission cost is illustrated as

$$e(h_z, j) = \sum_{t}^{\delta} \frac{w_t h(l_e, w_t)}{band(l_e, w_t)} . \tag{18}$$

In Eq (18), δ represents the quantity of required content, w_t indicates the size of the t -th

required content, $h(l_e, w_t)$ manifests the minimum network distance between the local server and the needed content location, and $band(l_e, w_t)$ illustrates the bandwidth between the local server and the required content location. When the demand content is stored on the local server, $h(l_e, w_t)$=0 and $band(l_e, w_t)$=1. Therefore, the job resource matching optimization problem is described as

$$P1: max \sum_{z=1}^{f} \sum_{j=1}^{J} \sum_{h_z=1}^{l_z} \frac{\pi_{hj}^{z} Sim(h_z, j) y(h_z)}{\left[e(h_z, j)+1\right]}. \tag{19}$$

$$s.t. \begin{cases} \pi_{hj}^{z} \in \{0,1\} \\ \sum_{j=1}^{J} \pi_{hj}^{z} = 1 \\ j = 1,2,\mathrm{L}\ ,J \\ h = 1,2,\mathrm{L}\ ,l_z \end{cases}. \tag{20}$$

In Eq (19), for $\pi_{hj}^{z} \in \{0,1\}$, when the h-th task matches the j-th resource pool, π_{hj}^{z}=1; otherwise, π_{hj}^{z}=0. Since problem $P1$ is a shaping programming problem, that is, an NP hard problem, in order to solve this problem, problem $P1$ is transformed into an optimal matching problem under the condition of a weighted bipartite graph. Consequently, in the weighted bipartite graph, the weight of task h_z and resource pool j is composed of their similarities. The weight between task priority and network transmission cost can be depicted as

$$\psi(h_z, j) = \frac{\lambda_1 Sim(h_z, j)}{\overline{Sim(h_z, j)}} + \frac{\lambda_2 y(h_z)}{\overline{y(h_z)}} - \frac{\lambda_3 e(h_z, j)}{\overline{e(h_z, j)}}. \tag{21}$$

$\psi(h_z, j)$ represents the weight between task h_z and container u_r, λ_1 denotes the weight coefficient of similarity, λ_2 illustrates the weight coefficient of task priority, and λ_3 manifests the weight coefficient of network transmission cost.

$$\overline{Sim(h_z, j)} = \sum_{z=1}^{f} \sum_{j=1}^{J} \sum_{h_z=1}^{l_z} \frac{Sim(h_z, j)}{\sum_{z=1}^{f} \sum_{h_z=1}^{l_z} j} \tag{22}$$

$$\overline{y(h_z)} = \sum_{z=1}^{f} \sum_{h_z=1}^{l_z} \frac{y(h_z)}{\sum_{z=1}^{f} l_z} \tag{23}$$

$$\overline{e(h_z, j)} = \sum_{z=1}^{f} \sum_{j=1}^{J} \sum_{h_z=1}^{z'_z} \frac{e(h_z, j)}{\sum_{z=1}^{f} \sum_{h_z=1}^{l_z} j} \tag{24}$$

$P1$ can be reduced to the following programming problem.

$$P2: max \sum_{z=1}^{f} \sum_{j=1}^{J} \sum_{h_z=1}^{l_z} \pi_{hj}^{z} \psi(h_z, j) \tag{25}$$

$$s.t.\quad \begin{aligned} &\pi_{hj}^{z} \in \{0,1\} \\ &\sum_{j=1}^{J} \pi_{hj}^{z} = 1 \\ &j = 1,2,\mathrm{L}\ ,J \\ &h = 1,2,\mathrm{L}\ ,l_z \end{aligned} \tag{26}$$

When $\sum_{z=1}^{f} l_z > j$, $\sum_{z=1}^{f} l_z$-j virtual resource pools need to be added, so that there is a one-to-one relationship between each task and each resource pool. The weight between each task and the added virtual resource pool is zero. The resource allocation algorithm is based on comprehensive utility, as shown in Algorithm 4.3.

4. Implementation of resource allocation optimization algorithm based on comprehensive utility in MEC environment

4.1. Job classification based on improved Naive Bayesian algorithm

The first step is to establish the conditional probability of the job by using Naive Bayes and then calculate the posterior probability of each job type according to the formula. Aiming at the shortcomings of Naive Bayes, the Laplace Smoothing technique is used to optimize the Naive Bayes algorithm. Finally, the posterior probability of the job to be classified is computed, and the job type is judged according to the acquired posterior probability of the job type. The core pseudo-code of job classification based on the improved Naive Bayesian algorithm is shown in Algorithm 1.

Algorithm 1: Job classification based on improved Naive Bayesian algorithm

Input: Job volume ω, job [1,2,..., f] // job is defined as the job to be classified

1. for (ω=1 ; $\omega \le f$; ω++) do
2. Establish Bayesian conditional probability // according to Eq (3), (4)
3. Calculate the posterior probability of the operation type according to Eq (8)
4. Use Laplace smoothing technique to eliminate the situation where the right side of Eq (8) is 0 // Eq (11)
5. Obtain the posterior probability of CPU-intensive jobs and I/O-intensive jobs // Eq (12) and Eq (13)
6. call.Classifier(ω)
7. if $\omega \leftarrow P(job_z = CPU|\mathrm{X}_1,\mathrm{K}\ ,\mathrm{X}_n) > P(job_z = I/O|\mathrm{X}_1,\mathrm{K}\ ,\mathrm{X}_n)$ then
8. the job is CPU-intensive
9. else if $\omega \leftarrow P(job_z = I/O|\mathrm{X}_1,\mathrm{K}\ ,\mathrm{X}_n) > P(job_z = CPU|\mathrm{X}_1,\mathrm{K}\ ,\mathrm{X}_n)$
10. the job is I/O-intensive
11. end if
12. end for
13. return JobType[1,2,…,f]

4.2. Resource service nodes are classified based on resource utilization

According to the resource utilization rate, the resource pool is divided into CPU main resources and I/O main resources. The core pseudo-code for classifying resource service nodes is shown in Algorithm 2.

Algorithm 2: Resource classification algorithm based on status of resource service nodes

Input: Resource Service Node $j=\{1,2,L,J\}$ // The number of J resource nodes to be classified
Output: $Classify[1,2,K,J]$ // Classified resource nodes
1. For each j
2. CPU_{usage}, I/O_{usage} // Obtain the resource utilization of the CPU and I/O in TaskTracker according to Eqs (14) and (15)
3. if $CPU_{usage} > I/O_{usage}$
4. the resource node is the main CPU resource
5. else $CPU_{usage} \le I/O_{usage}$
6. the resource node is the main I/O resource
7. end if
8. end for
9. return $Classify[1,2,K,J]$

4.3. Resource allocation optimization based on comprehensive utility

It is necessary to divide corresponding types of jobs into corresponding resource pools. This paper formulates a resource allocation algorithm based on comprehensive utility by calculating the similarity between tasks and resource pools, task priority and network transmission overhead of required resources. The core pseudo-code is exhibited in Algorithm 3.

The flow chart of the resource allocation optimization algorithm based on comprehensive utility in the MEC environment is shown in Figure 1.

5. Performance evaluation

5.1. Experimental environment and configuration

5.1.1. Environment settings

1) Experimental environment

This experimental environment is ubuntu-18.10-desktop-amd64, JDK1.8.0_11. Use the virtual machine software VMware Workstation 15.0.4 and SSH tools OpenSSH-server and OpenSSH-client. Build cloud computing framework Hadoop-3.1.2. The Linux system cloning tool is Clonezilla. The integrated development environment is Linux Eclipse 4.5.0. The specific experimental test environment and cluster node configuration in this study are indicated in Figure 2 and Table 2.

In order to simulate the actual scene of the experiment, the edge server is built with nine mobile terminals with different configurations. The remote cloud uses the Aliyun instance to build the environment. Edge servers and remote cloud communicate via VPN. Table 2 shows the configuration of the cluster node of the Edge Server.

2) Experimental data

The experiment data comes from the dataset of Stanford Network [30] (SNAP), including the online social network set, communication network set, Amazon network set, and other large network data sets. The range is [2, 40] G, the size of the experimental data set is about 40 GB, the size of each segmented data is 128 MB, and the range of the map task is [14, 300]. Therefore, the scale of the dataset executed by the benchmark program in this experiment is shown in Table 3.

Algorithm 3: Resource allocation algorithm based on Comprehensive Utility

Input： Resource pool set $J=\{1,2,\mathrm{K},j\}$， job $Job[1,2,\ldots,f]$， task set $H=\{h_1,h_2,\mathrm{K},h_z;z=1,2,\mathrm{K},f\}$

Output： Resource allocation result HashMap

1. for each $h_z \in H$ do
2. Calculate the similarity $Sim(h_z,j)$ between task h_z and resource service node j // according to Eq (16)
3. Calculate the priority $y(h_z)$ of task h_z // according to Eq (17)
4. Calculate the network transmission cost $e(h_z,j)$ // according to Eq (18)
5. end for each
6. for each heartbeat information
7. if $\sum_{z=1}^{f} l_z > j$ then
8. add $\sum_{z=1}^{f} l_z - j$ virtual resource pools
9. Repeat
10. Use weighted bipartite graph matching method to select mutually matching job and resource service nodes
11. until obtaining the optimal solution matching the job and resource service node // according to the formula (19)
12. else
13. add $j-\sum_{z=1}^{f} l_z$ virtual resource pools
14. end if
15. update matching operation of job and resource service node
16. end for each
17. return obtained matching jobs and resource service nodes

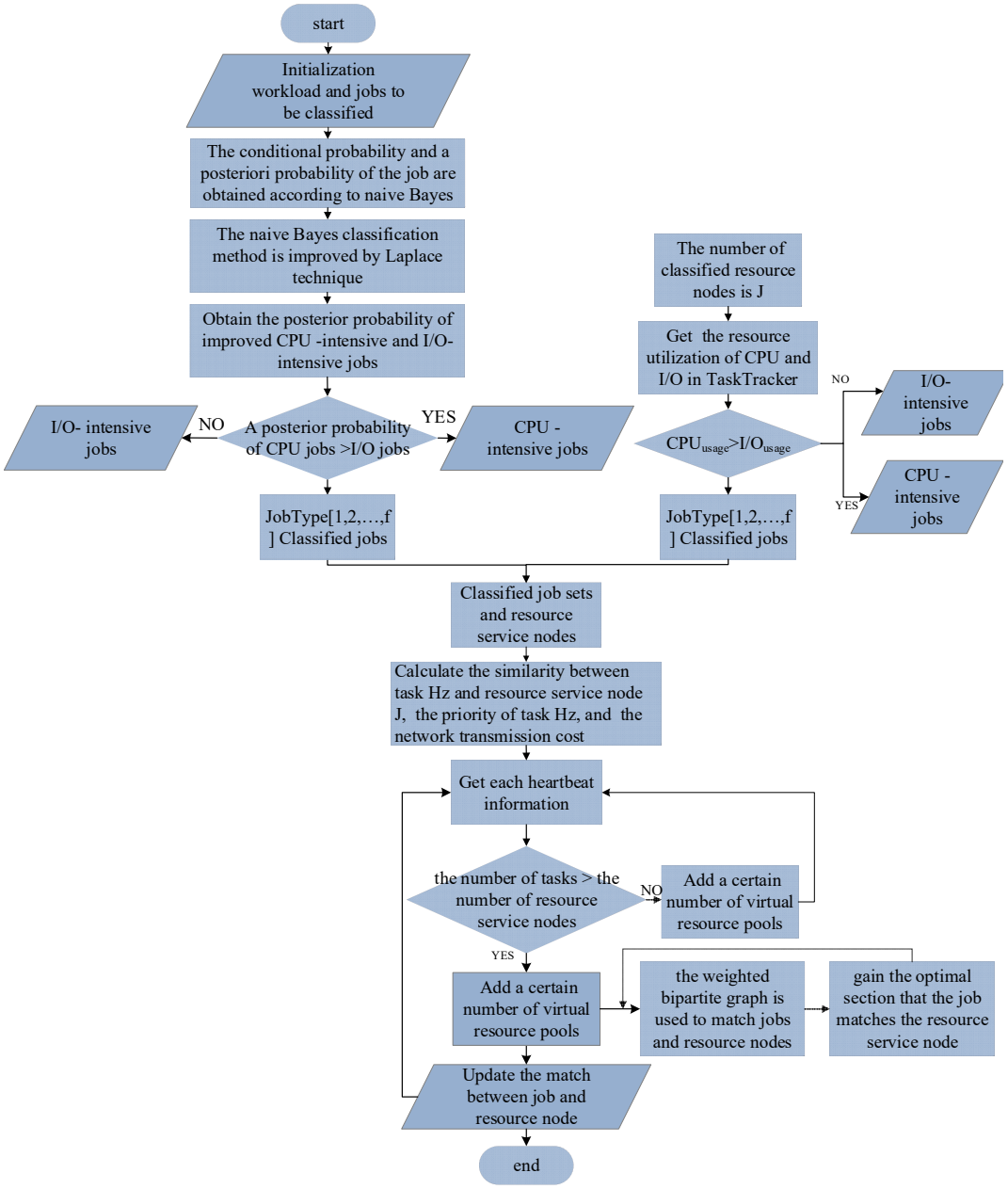

Figure 1. The flow chart of resource allocation optimization algorithm based on comprehensive utility in MEC environment.

Table 2. Edge server cluster node configuration.

Node type and name	IP address	CPU type	Running memory
Master (NameNode)	193.168.121.101	Inter Core i5 3.30 GHz, 2 cores, 500 G disk	8 GB
Slave (DateNode1)	193.168.121.102	Inter Core i5 3.30 GHz, 2 cores, 500 G disk	2 GB
Slave (DateNode2)	193.168.121.103	Inter Core i5 3.30 GHz, 2 cores, 500 G disk	8 GB
Slave (DateNode3)	193.168.121.104	Inter Core i5 3.30 GHz, 2 cores, 500 G disk	4 GB

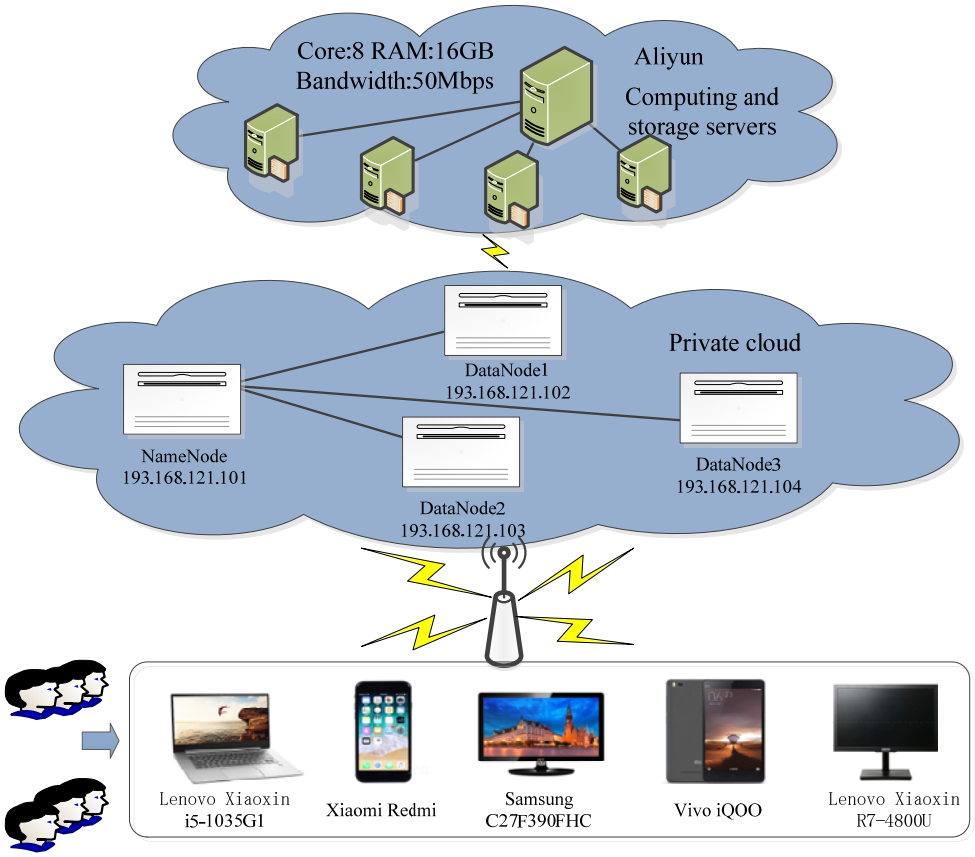

Figure 2. Experimental test environment.

Table 3. Data set size of benchmark program execution.

Data set	Data count	Map tasks
Online social network collection	4039 data of Facebook social circle, a total of 2 G data volume	16
Communication network set	36,692 data of e-mail communication network, a total of 8G data volume	32
Amazon Web collection	Purchase information and all comment data of various products, a total of 25 G data volume	190
Wikipedia Network	7115 data voted by Wiki, a total of 4 G data	25

5.1.2. Test cases and parameter settings

The experiment uses different job streams as the test cases [31], namely Wordcount, Kmeans, and Teragen. Since there are not many data calculation operations in the map and reduced phases of WordCount, this type of job can be classified as I/O-intensive. Kmeans involves more data computing operations in the Map and Reduce phases and does not have too many intermediate data read and write operations, so this type of job is classified as CPU-intensive. The data generated by Teragen is mostly used for subsequent programs, so this type of job can be classified as I/O intensive. The test case related data description of this experiment is shown in Table 4.

Table 4. Description of experimental test cases.

Test procedure	Description	Characteristic
WordCount	Total word count	I/O-intensive
Kmeans	Smaller dimension, numerical and continuous data set	CPU-intensive
Teragen	The generated data (3 GB) is used in subsequent procedures	I/O-intensive

5.2. Experimental results and analysis

In order to verify the effectiveness and stability of the algorithm proposed in this paper, the average job execution time and hit rate are used as the evaluation indicators of algorithm performance.

5.2.1. Influence of job number on algorithm performance

This group of experiments analyzes the average job execution time of the same job running under FIFO [32], FAIR [33], COSHH [34] and RAOCU. This experiment sets four job streams for each type of job, each job stream has 35 jobs, and a total of 140 jobs are submitted. Jobs of WorkCount, Kmeans and Teragen types are executed eight times each with job counts of 20, 60, 100 and 140, respectively. The average execution times of jobs under different numbers of jobs are shown in Figure 3. The job hit rates under different job numbers are shown in Figure 4.

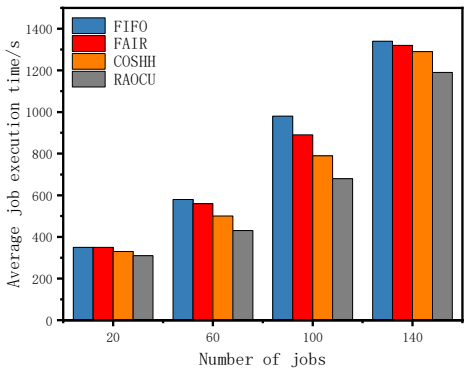

Figure 3. Comparison of average job execution times in different job quantities.

Figure 3 shows that when the number of jobs is 20, the average execution times of FIFO, FAIR, COSHH and RAOCU have little distinction, and they can all be stable at about 350 s. With increasing job data, COSHH can achieve a lower average job execution delay than FIFO and FAIR. The average job execution delay of the algorithm in this paper can show a lower trend than the other three algorithms. As the scale of job data increases, some jobs have priority implemented, while some jobs are only part of the execution jobs. For the jobs with priority execution, the Map task collects the job execution information and classifies the jobs, and then the remaining job allocates resources according to the corresponding labels. Because the RAOCU uses an improved Naive Bayes classification algorithm to classify the job, it can effectively speed up the classification of job data and reduce the average job execution time.

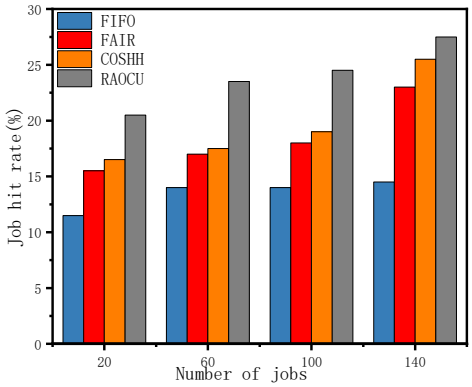

Figure 4. Comparison of job hit rates in different job numbers.

Figure 4 shows that with the increase in the number of tasks, the job hit rate of each algorithm also increases. For example, under the number of 140 jobs, the job hit rate of RAOCU is about 46.3% higher than that of 20 tasks, so the job hit rate of RAOCU is higher than those of the FIFO, FAIR and COSHH algorithms. In the case of 140 jobs, the RAOCU hit rate is about 25.9% higher than FIFO, 12.96% higher than FAIR and 11.1% higher than COSHH.

5.2.2. Influence of available storage space on algorithm performance under different resource service nodes

Figure 5 describes the average execution times of jobs with available storage space under different resource service nodes. Figure 6 illustrates the hit rates of job resources under different storage spaces.

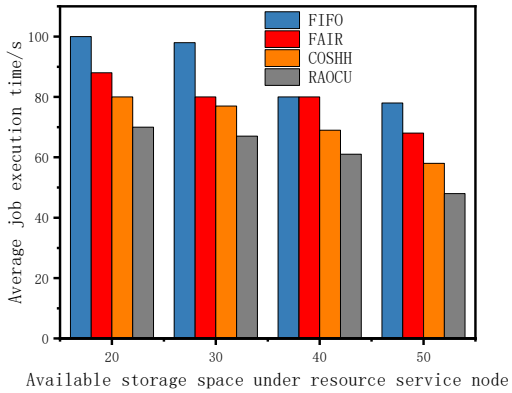

Figure 5. Comparison of average job execution time in different available storage spaces.

As shown in Figure 5, as the available storage space increases, the average job execution time for each algorithm decreases. When the size of the storage space under the resource service node increases from 20 to 50, the average job execution time of RAOCU is reduced by about 53.3%. Under the same available storage space, the average job execution time of the algorithm in this paper shows a lower trend than the other three algorithms. The resource location considered in RAOCU establishes the

relationship between the task and the resource pool according to the similarity, which can be matched by the task content and effectively reduce the average execution time of the job.

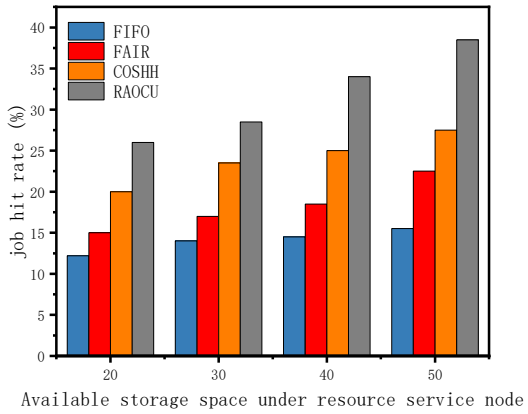

Figure 6. Comparison of job hit rates in different available storage spaces.

As shown in Figure 6, with the increase of storage space under the resource service node, the job hit rate of each algorithm also rises. When the size of the storage space under the resource service node changes from 20 to 50, the job hit rate of RAOCU increases by about 61.5%, which is higher than those of the FIFO, FAIR and COSHH algorithms. When the available storage space is 50, RAOCU's job hit rate is about 31.2% higher than FIFO's, 26% higher than FAIR's and 14.3% higher than COSHH's. With the same available storage space, the job hit rates of the other three algorithms are lower than that of the RAOCU algorithm. As the available storage space under the resource service node increases, the edge nodes contain more content. In resource matching, RAOCU considers three factors: resource location, task priority and network transmission cost, thus improving the hit rate of each algorithm.

5.2.3. Influence of the number of resource service nodes on algorithm performance

Figure 7 depicts the influence of the number of resource service nodes on the average job execution times. Figure 8 manifests the influence of the number of resource service nodes on the job hit rates.

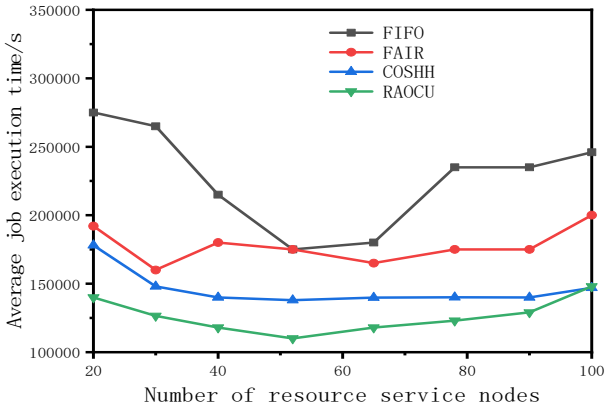

Figure 7. Comparison of average job execution times in different numbers of resource service nodes.

As shown in Figure 7, when the number of resource service nodes is between 20 and 100, the FIFO and FAIR algorithms fluctuate greatly. The average task execution delay of FIFO decreases first and then increases with the increase of resource service nodes. FAIR's average task execution delay first decreases and gradually increases with the number of resource service nodes. COSHH's performance shows a trend of first declining and then stabilizing. When it is stable, RAOCU shows a lower average job response time than COSHH. When the number of resource service nodes is about 52, the average job response time of RAOCU is the lowest.

It can be seen from Figure 8 that with the increasing number of resource service nodes, the job hit rates of FIFO and FAIR algorithms are in a relatively stable trend. Compared with FIFO and FAIR, COSHH and RAOCU show a convex trend with the increase in the number of resource service nodes. With the increase of resource service nodes, the job hit rates first show an upward trend and then gradually decrease. RAOCU considers the classification of jobs and the classification of resource service nodes. When the number of resource service nodes is too low, the matching between jobs and corresponding resource service nodes cannot be well completed, so the job hit rate is low. When the number of resource service nodes exceeds a specific number, the job will be matched to resource service nodes of the same type. Hence, RAOCU improves the utilization rate of resource nodes to a certain extent and has higher optimization performance than FIFO, FAIR and COSHH.

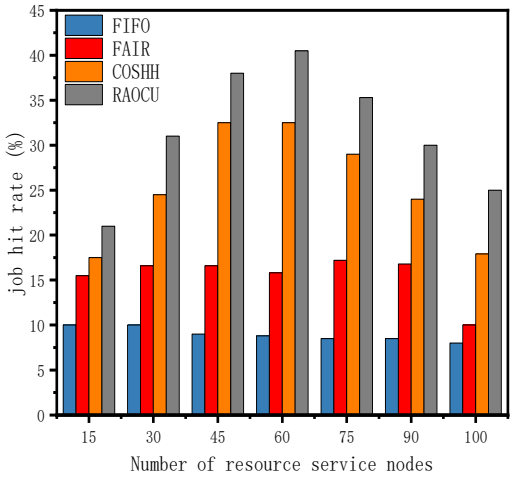

Figure 8. Comparison of job hit rates in different numbers of resource service nodes.

5.3. Experiment summary

By analyzing the above three sets of experiments, the following conclusions can be drawn: 1) The number of jobs impacts the resource optimization algorithm. As the number of jobs increases, the average task execution delay and the job hit rate also increase. 2) The available storage space under different resource service nodes will affect the resource allocation optimization algorithm. As the available storage space increases, the average task execution delay decreases, and the job hit rate gradually increases. 3) The number of resource service nodes influences the resource optimization algorithm. As the number of resource service nodes increases, the average task execution time increases steadily, while the job hit rate increases first and then gradually decreases.

6. Conclusions and future work

This paper studies the resource allocation optimization method based on comprehensive utility in multi-user and multiple MEC environments. The algorithm mainly includes job classification based on an improved Naive Bayesian algorithm, resource service node classification based on resource utilization rate and resource allocation based on comprehensive utility. Finally, the simulation results show that the RAOCU algorithm can reduce the resource occupancy rate and improve the resource utilization rate. At the same time, it has good performance in the average job execution delay and job hit rate. However, when classifying the computing tasks requested by the mobile terminal, this work does not consider the deadline and computing cost of the job. In the classification method of resource service nodes, only the resource utilization rate of resource service nodes is considered for classification. The network load of resource service node classification after job classification is not considered, so the next key work will be about these aspects of research.

Acknowledgments

The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61802353), the Natural Science Foundation of Henan Province (No. 202300410505), the project of Science and Technology of Henan Province (NO. 192102210270) and the Henan Provincial Department of Science and Technology (NO. 192102210270).

Conflict of interest

The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. J. H. Anajemba, T. Yue, C. Iwendi, P. Chatterjee, D. Ngabo, W. S. Alnumay, A secure multi-user privacy technique for wireless IoT networks using stochastic privacy optimization, *IEEE Int. Things J.*, **9** (2021), 2566–2577. https://doi.org/10.1109/JIOT.2021.3050755
2. M. Othman, S. A. Madani, S. U. Khan, A survey of mobile cloud computing application models, *IEEE Commun. Surv. Tutorials*, **16** (2014), 393–413. https://doi.org/10.1109/SURV.2013.062613.00160
3. B. Panchali, Edge computing-background and overview, in *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, (2018), 580–582, https://doi.org/10.1109/ICSSIT.2018.8748352
4. Taleb T, Samdanis K, Mada B, H. Flinck, S. Dutta, D Sabella, On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration, *IEEE Commun. Surv. Tutorials*, **19** (2017), 1657–1681. https://doi.org/10.1109/COMST.2017.2705720
5. K. LeDoux, P. Visser, D. Hulin, H. Nguyen, Starting large synchronous motors in weak power systems, in *Industry Applications Society 60th Annual Petroleum and Chemical Industry Conference*, (2013), 1–8. https://doi.org/10.1109/PCICon.2013.6666022

6. N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, *IEEE Int. Things J.*, **5** (2017), 450–465. https://doi.org/10.1109/JIOT.2017.2750180
7. J. B. Wang, H. Yang, M. Cheng, J. Y. Wang, M. Lin, J. Wang, Joint optimization of offloading and resources allocation in secure mobile edge computing systems, *IEEE Trans. Veh. Technol.*, **69** (2020), 8843–8854. https://doi.org/10.1109/TVT.2020.2996254
8. M. Aljarah, M. M. Shurman, S. H. Alnabelsi, Cooperative-hierarchical based edge-computing approach for resources allocation of distributed mobile and IoT applications, *Int. J. Electr. Comput. Eng.*, **10** (2020), 296–307. https://doi.org/10.11591/ijece.v10i1.pp296-307
9. X. Li, X. Zhou, C. Sun, D. W. K. Ng, Online policies for throughput maximization of energy-constrained wireless-powered communication systems, *IEEE Trans. Wireless Commun.*, **18** (2019), 1463–1476. https://doi.org/10.1109/TWC.2018.2890030
10. T. X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, *IEEE Trans. Veh. Technol.*, **68** (2019), 856–868. https://doi.org/10.1109/TVT.2018.2881191
11. J. Xu, B. Palanisamy, H. Ludwig, Q. Wang, Zenith: Utility-aware resource allocation for edge computing, in *2017 IEEE International Conference on Edge Computing (EDGE)*, (2017), 47–54.
12. B. Dab, N. Aitsaadi, R. Langar, Joint Optimization of Offloading and Resource Allocation Scheme for Mobile Edge Computing, in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, (2019), 1–7. https://doi.org/10.1109/WCNC.2019.8885537
13. J. Ren, G. Yu, Y. Cai, Y. He, Latency optimization for resource allocation in mobile-edge computation offloading, *IEEE Trans. Wireless Commun.*, **17** (2018), 5506–5519. https://doi.org/10.1109/TWC.2018.2845360
14. C. You, K. Huang, H. Chae, B. H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading, *IEEE Trans. Wireless Commun.*, **16** (2017), 1397–1411. https://doi.org/10.1109/TWC.2016.2633522
15. S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, *IEEE Trans. Signal Inf. Process. Networks*, **1** (2015), 89–103. https://doi.org/10.1109/TSIPN.2015.2448520
16. I. Ketykó, L. Kecskés, C. Nemes, L. Farkas, Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing, in *European Conference on Networks and Communications*, (2016), 225–229. https://doi.org/10.1109/EuCNC.2016.7561037
17. C. Wang, F. R. Yu, C. Liang, Q. Chen, L. Tang, Joint computation offloading and interference management in wireless cellular networks with mobile edge computing, *IEEE Trans. Veh. Technol.*, **66** (2017), 7432–7445. https://doi.org/10.1109/TVT.2017.2672701
18. C. Lemaréchal, S. Boyd, L. Vandenberghe, Convex optimization, Cambridge University Press, 2004 hardback, *Eur. J. Oper. Res.*, **170** (2016), 326–327. https://doi.org/10.1016/j.ejor.2005.02.002
19. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Mach. Learn.*, **3** (2010), 1–122. https://doi.org/10.1561/2200000016
20. P. Liu, J. Li, H. Li, Y. Meng, Convex optimisation-based joint channel and power allocation scheme for orthogonal frequency division multiple access networks, *IET Commun.*, **9** (2014), 28–32. https://doi.org/10.1049/iet-com.2014.0409

21. S. Jabeen, P. H. Ho, A Benchmark for joint channel allocation and user scheduling in flexible heterogeneous networks, *IEEE Trans. Veh. Technol.*, **68** (2019), 9233–9244. https://doi.org/10.1109/TVT.2019.2930884
22. M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, S. Papavassiliou, ENERDGE: Distributed energy-aware resource allocation at the edge, *Sensors*, **22** (2022), 660. https://doi.org/10.3390/s22020660
23. Y. Zuo, Y. Liu, User selection aware joint radio-and-computing resource allocation for federated edge learning, in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, (2020), 292–297. https://doi.org/10.1109/WCSP49889.2020.9299802
24. Y. Fan, L. Wang, W. Wu, D. Du, Cloud/edge computing resource allocation and pricing for mobile blockchain: An iterative greedy and search approach, *IEEE Trans. Comput. Social Syst.*, **8** (2021), 451–463. https://doi.org/10.1109/TCSS.2021.3049152
25. I. AlQerm, J. Pan, Enhanced online Q-learning scheme for resource allocation with maximum utility and fairness in edge-IoT networks, *IEEE Trans. Network Sci. Eng.*, **7** (2020), 3074–3086. https://doi.org/10.1109/TNSE.2020.3015689
26. B. Huang, Z. Li, Y. Xu, L. Pan, S. Wang, H. Hu, et al., Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing, *Wireless Commun. Mob. Comput.*, (2020), 1–17. https://doi.org/10.1155/2020/2765491
27. X. Lin, J. Shao, R. Liu, W. Sun, W. Hu, Performance and cost of upstream resource allocation for inter-edge-datacenter bulk transfers, in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, (2020), 634–639. https://doi.org/10.1109/ICCC49849.2020.9238818
28. X. Zhu, L. Yang, Resource allocation for virtualized wireless networks with mobile edge computing, in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, (2020), 139–144. https://doi.org/10.1109/ICCCWorkshops49972.2020.9209941
29. M. Shabbir, A. Shabbir, C. Iwendi, A. R. Javed, M. Rizwan, N. Herencsar, et al., Enhancing security of health information using modular encryption standard in mobile cloud computing, *IEEE Access*, **9** (2021), 8820–8834. https://doi.org/10.1109/ACCESS.2021.3049564
30. J. Leskovec, *Stanford Large Network Dataset Collection*, 2022. Available from: http://snap.stanford.edu/data/index.html.
31. S. Huang, J. Huang, J. Dai, T. Xie, B. Huang, The HiBench benchmark suite: Characterization of the MapReduce-based data analysis, in *New Frontiers in Information and Software as Services*, Springer, (2011), 209–228. https://doi.org/10.1007/978-3-642-19294-4_9
32. B. T. Rao, L. S. S. Reddy, Survey on improved scheduling in Hadoop MapReduce in cloud environments, preprint, arXiv:1207.0780.
33. M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, I. Stoica, *Job Scheduling for Multi-user Mapreduce Clusters*, Technical Report UCB/EECS-2009-55, EECS Department, University of California, Berkeley, (2009), 213–217.
34. A. Rasooli, D. G. Down, COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems, *Future Gener. Comput. Syst.*, **36** (2014), 1–15. https://doi.org/10.1016/j.future.2014.01.002

AIMS Press

©2022 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0).