Received April 30, 2020, accepted May 21, 2020, date of publication June 9, 2020, date of current version July 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3000985

Resource Constrained Profit Optimization Method for Task Scheduling in Edge Cloud

LIQIONG CHEN, (Member, IEEE), KUN GUO, GUOQING FAN, CAN WANG, AND SHILONG SONG
Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China

Corresponding author: Liqiong Chen (lqchen@sit.edu.cn)

This work was supported in part by the NSF of China under Grant 61702334, in part by the Shanghai Municipal Natural Science Foundation under Grant 17ZR1406900 and Grant 17ZR1429700, and in part by the Collaborative Innovation Foundation of Shanghai Institute of Technology under Grant XTCX2016-20.

ABSTRACT Edge cloud is a cloud computing system built on edge infrastructure. Task scheduling optimization is the key technology to ensure the quality of service in edge cloud. However, the openness of the edge cloud environment challenges the load balancing and profit optimization of task scheduling. In this paper, we analyze the business process and optimization factors of task scheduling in edge cloud. First, we propose a resource constrained task scheduling profit optimization algorithm (RCTSPO), which consists of clustering preprocessing, classification, profit matrix construction and optimal scheduling strategy calculation. Clustering preprocessing gathers similar tasks into one class and perform a classification on the clustered tasks. Then construct the profit matrix for resource constrained task scheduling, and the optimal task scheduling strategy is obtained based on the constructed profit matrix. Second, Petri nets are used to construct the different components of edge cloud, such as resource, task, user request and virtual machine, thus forming the task scheduling model of edge cloud. Third, the properties of task scheduling model are verified by using the related theory and tools of Petri nets. Finally, several experiments are done to evaluate the proposed method, the simulation results show that the algorithm not only achieves the maximum profit, but also performs well in terms of time, reliability and load balancing of task scheduling.

INDEX TERMS Edge cloud, Petri nets, profit, resource constraints, task scheduling.

I. INTRODUCTION

Edge computing brings the advantages of low latency, small network load and low data management cost to the Internet of Things (IOT) [1]. Edge cloud is essentially a cloud computing system built on the edge infrastructure, which brings stability to the connected devices in the IOT network. By delivering the cloud services to the edges of network in the proximity to the users, the latency of transmission time can be reduced and the heavy burden on the backhaul link is avoided [2]. With the application of edge cloud in key fields such as smart city, the number of tasks has increased rapidly. A reasonable and efficient task scheduling strategy can improve the performance of the edge cloud [3]. In view of the social benefits and economic value of cloud computing and edge computing, the research on task scheduling of edge cloud has become a hot research field [4].

Edge cloud scheduling is divided into computation offloading and task scheduling according to the operation level. Computation offloading is used to transfer some tasks from the edge to the cloud for processing based on the attributes of task, such as energy consumption and calculation volume, thus extending the life cycle of edge devices and improving the task response time [5]. Task scheduling is used to allocate tasks to the corresponding virtual machines for execution, so as to optimize the performance of cloud, such as load balancing, reliability, response time, and utility. With the increase of tasks in the edge cloud, how to design an effective task scheduling strategy under the limited virtual machines has become a challenging problem.

However, the characteristics of edge cloud applications are diversified, user requests are random, and virtual machine resources are limited. The operating environment is more

The associate editor coordinating the review of this manuscript and approving it for publication was Yuyu Yin.

open, dynamic and complex, which makes the optimization of task scheduling in edge cloud be more difficult. (1) In the task scheduling process, load balancing of virtual machine is a problem that cannot be ignored. Due to the heterogeneity of physical devices in the edge cloud and the difference of resource requests from a large number of users. Some virtual machines may load imbalance, which may cause the waste of resources and excessive energy consumption. (2) The profit of task scheduling is an important factor that determines the application of edge cloud. The workload and required resources of each task are different, and the same type resource may have different unit price. Therefore, the profit of edge cloud is different under different task scheduling strategies [6]. (3) The edge cloud is based on the edge virtualization resources, which often includes many servers or devices that work together. In addition to the diversity and uncertainty of user requests, it is necessary to construct the task scheduling model for describing the complex software structure of edge cloud. In order to solve these problems, this paper proposes a resource constrained task scheduling profit optimization algorithm (RCTSPO) to maximize the profit of edge cloud, and constructs the task scheduling model based on Petri nets. The contributions of this paper are as follows:

(1) We design a clustering preprocessing and classification to avoid the local optimization in task scheduling. It is used to classify the tasks to achieve batch processing, thus improving the efficiency of task scheduling.

(2) We propose a method to realize the load balancing and profit optimization of task scheduling. The equal subgraphs and their relationships are established between task and virtual machine under resource constraints. A profit matrix is got based on the profit of task scheduling. Finally, Kuhn-Munkres (KM) is used to get the optimal matching with the best profit, so as to maximize the profit and achieve the best load balance of task scheduling in edge cloud.

(3) We construct a task scheduling model in edge cloud based on Petri nets, which is used to model different components of edge cloud, the internal logic and time attribute are also considered. The theories of Petri nets are provided to validate the correctness of proposed method.

(4) Several simulation experiments are carried out to verify the effectiveness of the proposed method. The results demonstrate its correctness and promise.

The rest of this paper is organized as follows. Section II is the related work. Section III describes the task scheduling framework and requirements. The RCTSPO algorithm is proposed in Section IV. Section V constructs the task scheduling model. Section VI is the experimental simulation. While Section VII is the conclusion and future work.

II. RELATED WORK

Edge cloud is a cloud computing platform built on edge infrastructure. The "looking beyond the Internet" organized by NSF in 2016 discussed the development trend and demand of edge cloud [7]. AT & T released the "AT & T Edge Cloud (AEC)-White Paper", OpenStack [8] and other companies released the white papers related to edge cloud in 2018 to elaborate on the definition, architecture, application scenarios, main challenges and other issues of edge cloud computing. Gartner's top 10 strategic technology trends for 2020 points out that edge computing moves key applications and services closer to the people and devices that use them [9]. Reference [10] explored the dynamic configuration of service workflow for mobile e-commerce based on cloud edge framework. Although the research of edge cloud has got some achievements in architecture, model and other aspects, it is still facing many challenges, such as software structure design, edge cloud collaboration and migration [11].

Formal method has been applied in the field of computer hardware and software. It helps to increase the software developers' understanding of the system and to correct the errors in the design time. Many formal methods have been used to model edge software system, such as Petri net, Markov decision process. A formal model of fog computing is established based on price time Petri net, the algorithm and evaluation method of predicting task completion time are also proposed in Ref. [12]. In Ref [13], a digital alarm system based on fog calculation is designed to detect the displacement of Nasogastric tube, the elements and their attributes are described by using fuzzy Petri net. Reference [14] formally verified the protocol against basic security properties by using High Level Petri net (HLPN). Reference [15] formulated the service migration problem for mobile edge computing as a Markov Decision Process (MDP). The above literature mainly designs the business processes of fog computing and edge computing, without considering the profit optimization of task scheduling in edge cloud.

The cooperation and interaction between cloud and edge device can help reduce the energy consumption. Kaur K. *et al.* proposed a multi-objective evolutionary algorithm to balance the trade-off between energy efficiency and waiting time [16]. A load balancing technique which decreased the response time and processing time of fogs to consumers has been proposed in Ref [17]. A multi-objective evolutionary algorithm using Tchebycheff decomposition is proposed in Ref. [18], which is used to analyze the flow scheduling and routing in SDN. Reference [19] proposed a heuristic algorithm approximating the optimal solution to reduce brown energy consumption. In Ref. [20], ant colony algorithm is proposed to achieve load balancing and profit in cloud task scheduling. Aziza and Krichen [21] used Genetic Algorithm (GA) to estimate the time needed to run a group of tasks in task scheduling of cloud, so as to reduce the processing cost. Reference [22] proposed a data-intensive service edge deployment scheme to optimize response time for service deployment based on Genetic Algorithm. However, ant colony algorithm and genetic algorithm need to train the data first, and the results are uncertain.

The task scheduling in edge cloud often has a large number of tasks. Clustering algorithm can classify tasks and maximize the similarity between data samples within the same cluster [23]. Reference [24] proposed a new matrix factorization model to analyze the deep features of edge computing services and users based on deep features learning. Items with similar features are classified into one class by using K-means algorithm. Reference [25] proposed a dynamic grouping through K-implies which suits well for dynamic topology qualities of Vehicular ad-hoc Network. An algorithm for IoT devices' computation offloading decisions is proposed in Ref [26]. The initial clustering center of K-means clustering algorithm chooses the point under the high density as the initial clustering center to achieve better clustering effect [27]. When the number of tasks is far greater than the number of virtual machines, the polling mechanism is used to establish the equal subgraph of task and virtual machine. KM algorithm finds the matching strategy of maximum weights in complete matching. In Ref. [28], the task scheduling in the cloud is transformed into a bipartite graph matching problem, and the optimal bipartite graph search algorithm KM is used to calculate the task scheduling strategy with maximum profit. However, KM algorithm needs to establish equal subgraph, and the number of tasks in cloud is greater than the number of virtual machines. If polling mechanism is used to establish equal subgraph, it is easy to fall into the local optimization.

To sum up, ACO, GA and KM algorithms do not consider the characteristics of the large number of tasks in the edge cloud, and do not preprocess the tasks to improve the efficiency of task scheduling. The selection of initial clustering centers of traditional K-means clustering is random, and the clustering results are uncertain. This paper proposes a RCTSPO algorithm to solve the uncertainty of clustering results, which selects the initial clustering center in the high-density region to achieve K-means clustering. The classification avoids the local optimization of tasks in equal subgraph in task scheduling. In order to avoid being occupied for a long time due to the lack of resources provided by the virtual machine, resource constraints are added in constructing the equal subgraph. KM algorithm can maximize the profit of task scheduling. Furthermore, we construct a formal model to characterize the task scheduling process in edge cloud.

III. THE FRAMEWORK AND REQUIREMENTS OF TASK SCHEDULING IN EDGE CLOUD

A. SYSTEM FRAMEWORK

The edge cloud architecture is a three-tier network structure, as shown in Figure 1. The bottom layer is the terminal device of the end user, mainly composed of sensors, collectors, mobile phones, PC, smart watches, etc. The middle layer is the edge computing, mainly composed of edge devices (such as routers, gateways, small servers, etc.) with a certain computing capabilities. The top layer is the cloud computing center, mainly composed of virtual machines. The end user will initiate the application requirement of edge cloud, which consists of a series of tasks and their relationships. The edge layer has a certain amount of computing resources to handle simple tasks. The cloud computing center layer has virtual machine resources that can handle complex tasks. The implementation process of edge cloud application is to decide that tasks are executed by different layers (terminal, edge or cloud) according to task offloading strategy. The cloud dispatching center receives the edge cloud application requirements and the set of task to be processed, and assigns the tasks to the corresponding virtual machine for execution according to the task scheduling strategy.

The number of user requests submitted to the edge cloud is increasing gradually, it needs an effective task scheduling strategy to maximize the profit of edge cloud. Task scheduling is mainly allocating the tasks to be executed to virtual machines based on the optimization objectives. For optimizing and modeling resource constrained task scheduling in edge cloud, this paper intends to take the following two measures: (1) We propose a profit optimization algorithm for resource constrained task scheduling in edge cloud. (2)We construct a formal model of task, user request, virtual machine and scheduling center by using Petri nets, thus improving the model reusability.

The specific framework is shown in Figure 1, which is divided into three steps.

(1) Optimization algorithm: The tasks are clustered according to the related parameters (Memory, bandwidth, computing power) and classified to avoid local optimization. The resource constraints are considered to achieve load balancing of task scheduling, thus avoiding the idle resources. The equal subgraph is also established with scheduling profit as the weight, KM algorithm is used to calculate the optimal scheduling strategy.

(2) Model construction: Based on the execution results of the scheduling algorithm, the basic elements of edge cloud such as tasks, user requests, scheduling processes, and virtual machines are modeled by using Petri nets. Then, according to the actual requirements, the interface of the model is used to match the model of the basic elements to form the task scheduling model. Finally, the effectiveness and correctness of the constructed model are analyzed with the help of related tools of Petri nets.

(3) Simulation analysis: Aiming at the application scope and effectiveness of the proposed method, we design several simulation experiments to analyze and compare the RCTSPO algorithm with other scheduling methods, thus illustrating the effectiveness of the proposed method.

B. TASK SCHEDULING REQUIREMENT

The resource constrained task scheduling in edge cloud is given in the following, which includes the set of task, the set of virtual machine, the set of user request and its attributes. The user request consists of the set of task and relationships between tasks. Task has attributes such as length, computing power and memory capacity. The virtual machine has attributes such as computing power and memory capacity.

Definition 1: The user request is a 2-tuple $Rq_i = (TK_i, RL_i)$, $TK_{i,j}$ represents the jth task of Rq_i. RL_i: $TK_i \times TK_i \rightarrow \{>, +, ||, n\}$ is the relationship functions between

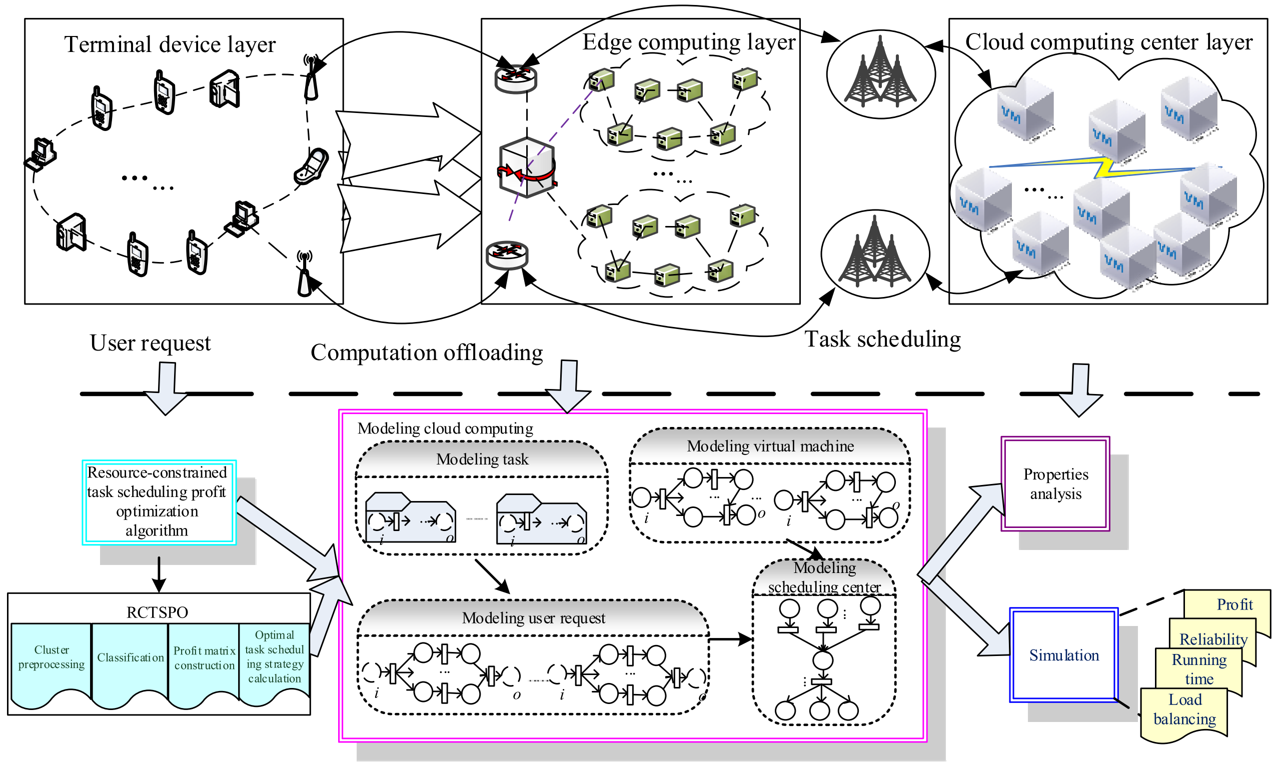


FIGURE 1. System framework.

tasks, $>$, $+$, $||$, n represent the sequence, selection, parallel and repeated relationships. Repeated execution means that the task is called multiple times, and n should be limited. Let Rq be the set of user requests.

Definition 2: The task is a 6-tuple $TK_{i,j} = \{T^{id}{}_{i,j}, T^{length}{}_{i,j}, T^{comp}{}_{i,j}, T^{ram}{}_{i,j}, T^{bw}{}_{i,j}, T^{deadline}{}_{i,j}\}$. $T^{id}{}_{i,j}$, $T^{length}{}_{i,j}$, $T^{com,}{}_{i,j}$, $T^{ram}{}_{i,j}$, $T^{bw}{}_{i,j}$, $T^{deadline}{}_{i,j}$ represent the number, length (millions of instructions), computing power (millions of instructions per second), memory capacity (MB), network bandwidth (MB/s) and the deadline of task (ms). TK is the set of all tasks.

Definition 3: The virtual machine is a 4-tuple $VM_i = \{V^{id}{}_i, V^{comp}{}_i, V^{ram}{}_i, V^{bw}{}_i\}$. $V^{id}{}_i$, $V^{comp}{}_i$, $V^{ram}{}_i$, $V^{bw}{}_i$ represent the number of virtual machine, computing power (Millions of instructions per second, $MIPS$), memory capacity (MB) and network bandwidth capacity (MB/s) of virtual machine VM_i. Let VM be the set of all virtual machines in the edge cloud.

C. PROBLEM DESCRIPTION

The profit of task scheduling is equal to the difference between the expense paid by the user and the cost. The cost of task scheduling is calculated by the resource size and time provided by the virtual machine. The value of a task is determined by the resources and the time it takes. It can be got by using Formula (1).

$$\text{value}_{i,\text{j}} = (T^{deadline*}_{\text{i,j}}(a^*T^{ram}_{i,j} + b^*T^{bw}_{i,j} + c^*T^{comp}_{i,j}) \tag{1}$$

a, b and c represent the price weight of memory, bandwidth and computing power. At the same time, the resources provided by the virtual machine are different, so the estimation value of the edge cloud for the virtual machine is shown in Formula (2).

$$\text{cost}_{i,j,k} = (\frac{T^{length}_{\text{i,j}}}{VM^{comp}_k})^*(a^*VM^{ram}_k + b^*VM^{bw}_k + c^*VM^{comp}_k) \tag{2}$$

In order to maximize the profit, the objective function of the resource constrained task scheduling in edge cloud is shown in Formula (3).

$$\text{MAX profit} = \sum_{\text{i}=0,j=0,k=0}^{|Rq|,|Rq_{\text{i}}|,|VM|} w(value_{i,j} - \cos t_{i,j,k})$$
$$s.t.T^{ram}_{\text{i,j}} < VM^{\text{ram}}_k \&\& T^{bw}_{i,j} < VM^{bw}_k \&\& T^{comp}_{i,j} < VM^{\text{comp}}_k$$
$$\text{i} \in [0, |Rq|),\ j \in [0, |Rq_{\text{i}}|),\ k \in [0, |VM|), w \in \{0, 1\} \tag{3}$$

When w is equal to 1, TK executes on the VM. When w is equal to 0, TK is not executed on the VM. The objective function is to maximize the profit of task scheduling under resource constraints, which can be modified by adjusting the profit weight.

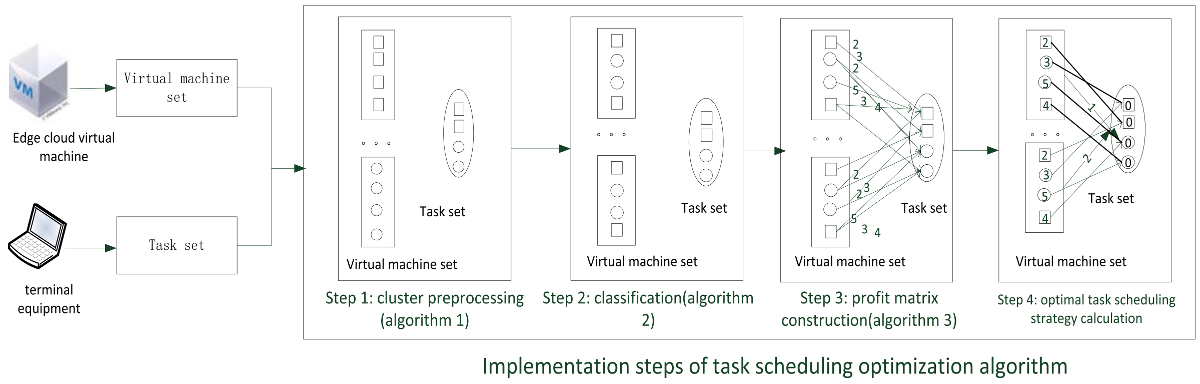

Edge cloud virtual machine
Virtual machine set
terminal equipment
Task set
Task set
Virtual machine set
Step 1: cluster preprocessing (algorithm 1)
Task set
Virtual machine set
Step 2: classification(algorithm 2)
Task set
Virtual machine set
Step 3: profit matrix construction(algorithm 3)
Task set
Virtual machine set
Step 4: optimal task scheduling strategy calculation
Implementation steps of task scheduling optimization algorithm

FIGURE 2. RCTSPO algorithm.

IV. TASK SCHEDULING ALGORITHM

A. TASK SCHEDULING PROCESS

This section will give the implementation process of the RCTSPO algorithm, which is shown in Figure 2. The input of RCTSPO algorithm is the task scheduling requirements of resource constrained edge cloud, and the output is the task scheduling strategy with optimal profit.

Step 1: Cluster preprocessing: K-means clustering is improved for tasks according to the parameters $T^{ram}{}_{i,j}$, $T^{bw}{}_{i,j}$ and $T^{comp}{}_{i,j}$. The whole virtual machine is divided into one class, and the task is divided into K class. The K initial value of clustering is determined by the number of tasks ($|TK|$) and the number of virtual machines ($|VM|$) ($K = |TK|/|VM|$). The value of k can be adjusted based on the simulation results.

Step 2: Classification: The clustering results are evenly distributed to K task sets according to the different task resource requirements. (K is the ratio of the number of tasks to the number of virtual machines.)

Step 3: Constructing the profit matrix: Establishing the matrix between task and virtual machine with $T^{id}{}_{i,j}$ as row coordinate and $VM^{id}{}_{k}$ as vertical coordinate. $TK_{i,j}$ is executed on the VM_k, the profit of this task scheduling is viewed as the value of the corresponding element in the profit matrix.

Step 4: Calculating the optimal task scheduling strategy: The profit matrix of Step 3 is taken as the input of *KM* algorithm, and find the matching with the best profit.

B. CLUSTER PREPROCESSING OF TASK SCHEDULING

In the first step of RCTSPO algorithm, K-means algorithm is used to dynamically find the initial clustering center. It calculates the distance between two data points in the task set by using Formula (4), and stores it in the matrix with the related row of task and the related column of virtual machine. The matrix is sorted in ascending order. The two data points with the smallest distance in the former K are selected as the initial clustering center $m_i (i = 0, \ldots, K)$. After selecting K initial clustering centers, the former n/K data points closest to the cluster center are classified into one cluster.

$$\mathrm{distance}_{(i,j),(e,f)} = \sqrt{(T^{ram}_{i,j} - T^{ram}_{e,f})^2 + (T^{bw}_{i,j} - T^{bw}_{e,f})^2 + (T^{comp}_{i,j} - T^{comp}_{e,f})^2} \tag{4}$$

Each cluster has its center, which is the cluster center. The cluster center is determined by Formula (5).

$$clusterCenter_j = \frac{1}{|TK|}\sum_{i=0}^{|TK|} \times \sqrt{(point^{ram}_{\mathrm{i}})^2 + (point^{\mathrm{bw}}_{i})^2 + (point^{comp}_{i})^2}, \quad j \in [0, \mathrm{K}) \tag{5}$$

$Point^{ram}{}_i$, $Point^{bw}{}_i$ and $Point^{comp}_i$ represent the memory attribute, bandwidth attribute and computing power attribute of the ith point to be clustered.

The pseudo-code of clustering preprocessing is illustrated in Algorithm 1.

Step 1 is used to determine the initial clustering center according to the above algorithm. Step 4 calculates the distance from the task set to the clustering center according to Formula (4), and selects the n/K nodes closest to the clustering center. Step 5 calculates the new clustering center by using Formula (5).

According to the criteria of |CenterOld-CenterNew| <0.1, (CenterOld represents the old cluster center, CenterNew represents the new cluster center). If the criterion is satisfied, the clustering process is completed. Otherwise, clustering continues to execute.

C. CLASSIFICATION

Tasks in the task clusters have the similar requirements for resources, but the resources provided by virtual machines are inconsistent. If a virtual machine with high performance is used to realize the tasks with light workload, virtual machine resources are wasted. The task with heavy workload is assigned to the virtual machine with low performance, which makes the task not be completed within the deadline. Therefore, it is necessary to improve this problem.

Algorithm 1 Cluster Preprocessing

```
Input: Task and virtual machine information
Output: Clustering information for tasks and virtual machines
1: Determining the initial cluster center according to the above algorithmic idea.
2: Set the Boolean flag to true.
3: while flag is true do
4:   Calculating the distance from the remaining nodes to the cluster center according to Formula (4), and select the nearest $n/k$ nodes to the cluster;
5:   Recalculating the average value of the data objects in each cluster according to Formula (5), and determine a new cluster center;
6:      if |CenterOld-CenterNew| <0.01 then
7:         Set the Boolean flag with false.
8:      else
9:         Set the Boolean flag with true.
10:     end if
11:end while
```

To solve this problem, this paper classifies tasks according to the required resources of tasks. For example, if there are three task classes after clustering, task class *A* needs more resources, task class *B* needs medium resources, and task class *C* needs less resources. According to the number of provided resources, virtual machines can also be divided into three types: more, medium and less. After the task is classified. In task class *A*, the required resources of tasks are also divided into more, medium and less. Task class *B* and task class *C* is the same as task class *A*.

Algorithm 2 Classification

```
Input: Task clustering result
Output: Mixed task clustering
1: Create $K$ empty clusters clusterSet.
2:for(Cluster cluster: clustertaskSet) do
3:   Initializing $K$ cluster centers.
4: end for
5:for(Cluster cluster:clusterSet) do
6:      for(Cluster clustertask:clustertaskSet) do
7:         Add the task to the new set.
8:      end for
9:end for
10: Calculate the new cluster center according to Formula (5)
```

The pseudo-code of classification is shown in Algorithm 2. Firstly, *K* empty clustering sets are created and *K* clustering centers are initialized. The tasks in the cluster set are successively distributed to new *K* cluster sets. Finally, the new clustering center is calculated based on Formula (5).

D. PROFIT MATRIX CONSTRUCTION

According to Formula (4), the distance between cluster centers is calculated, and the profit matrix is constructed by selecting the group with the smallest distance between task class and virtual machine class. Profit matrix ($p[T^{id}, VM^{id}]$) is a matrix with task as row and virtual machine as column. The profit of task scheduling is got by using Formula (6).

$$\begin{aligned}
\text{profit}_{i,j,\text{k}} &= value_{i,j} - \cos t_{i,j,k} \\
&= (T^{deadline}_{\text{i,j}} * (a * T^{ram}_{i,j} + b * T^{bw}_{i,j} + c * T^{comp}_{i,j}) \\
&\quad - (\frac{T^{length}_{i,j}}{VM^{comp}_{k}}) * (a * VM^{ram}_{k} + b * VM^{bw}_{k} \\
&\quad + c * VM^{comp}_{k}) \qquad (6)
\end{aligned}$$

Calculating the value of profit matrix in $p[(i,j),k] = profit_{i,j,k}$. If the required resources of task $TK_{i,j}$ are greater than the provided resources of virtual machine VM_k, then $p[(i,j),k] = 0$.

The pseudo-code of the profit matrix is illustrated in Algorithm 3. When the related parameters of the task are less than the related parameters of virtual machine, the profit of the current task scheduling is calculated according to Formula (6).

Algorithm 3 Profit Matrix Construction

```
Input: the related parameters of task set and virtual machine set
Output: weight matrix
1:for i = 0:m do
2:    for j = 0: |Rq_i| do
3:       for k = 0:n do
4:          if($T^{ram}{}_{i,j} < VM^{ram}{}_k$&&$T^{comp}{}_{i,j} < VM^{comp}{}_k$
               &&$T^{bw}{}_{i,j} < VM^{bw}{}_k$)
5:          Calculate the task scheduling profit assignment
               to matrix according to Formula (6).
6:          else
7:             Matrix value assigned to 0
8:          end if
9:       end for
10:  end for
11:end for
```

E. THE OPTIMIZATION OF SCHEDULING STRATEGY

The profit matrix is represented by bipartite graph $G =< V, E >$. The rows and columns of the matrix are formed into *V*, and the matrix value is *E*. In this paper, the *KM* (Kuhn-Munkres) optimal search algorithm is used to solve the optimal matching of tasks and virtual machines in task scheduling. In a bipartite graph, the left vertex is task, and the right vertex is virtual machine. Its improved algorithm (*KM* algorithm) assigns the left vertex as the maximum value in the profit matrix, and the right vertex is 0. If there is a best matching, it will continue. Otherwise, the index value is modified and the *KM* algorithm is used again. Finally, the best matching results are obtained.

F. ALGORITHM IMPLEMENTATION

The pseudo-code of the RSPOTS algorithm is illustrated in Algorithm 4. First, Algorithm 1 is used to cluster tasks and virtual machines to reduce the search scope and achieve the global optimization. Second, the tasks with different requirements are classified reasonably (Algorithm 2). Third, the profit matrix is determined, and the profit matrix is obtained (Algorithm 3). Finally, the profit matrix is used as the input of KM algorithm to achieve the optimal matching of task and virtual machine with profit maximization.

Algorithm 4 Implementation Steps of RCTSPO Algorithm

```
Input: Task and virtual machine
Output: Task scheduling strategy
1: Clustering preprocessing is executed by using Algorithm 1, and clustering information is obtained.
2: The clustering results are classified by Algorithm (2).
3: Determining the weighted matrix of clustering information and getting the weight matrix by Algorithm (3).
4: The profit matrix is calculated from the optimal scheduling strategy, and the task scheduling strategy with the maximum profit is obtained.
```

G. ALGORITHM COMPLEXITY ANALYSIS

The complexity analysis of RCTSPO algorithm is analyzed from the clustering preprocessing algorithm, classification algorithm and the optimal scheduling strategy. We will analyze from the time complexity and space complexity.

The improvement of clustering preprocessing algorithm is the selection of initial clustering center. The time complexity is $O(n)$, the space complexity is $O(a)$. n is the number of task, and a is a constant. k is the number of cluster, and $k = (n/|VM|)$. So the time complexity of K-means algorithm is $O(I^*n^*n^*m/|VM|)$, and the space complexity is $O(n^*m)$. Where m is the number of attributes of each element, and I is the number of iterations. I and m can be regarded as constants, so the time and space complexity is reduced to $O(n^2/|VM|)$.

Classification algorithm. It needs the extra space to store classification results, and the space complexity is $O(n)$. The system needs to traverse the task set twice. The time complexity of classification algorithm is $O(k^*(n/k))$.

The optimal task scheduling strategy needs to find the augmenting path for $O(n)$ times, and each augmentation needs to modify the top mark at most $O(n)$ times. The complexity of modifying the top scale is $O(n^2)$. The time complexity is $O(n^4)$. The space complexity of the code is $O(n^*2)$, which is $O(n)$.

Therefore, the time and space complexity of the proposed RCTSPO algorithm are $O(n^4)$ and $O(n^2/|VM|)$ *respectively.*

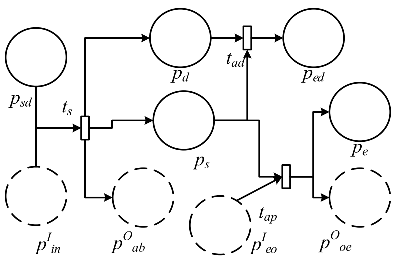


FIGURE 3. **Modeling task.**

V. TASK SCHEDULING MODEL

This section focuses on the requirements and characteristics of the edge cloud, and constructs the task scheduling model. The task scheduling model is constructed based on the scheduling strategy. The scheduling strategy is abstracted as an input to the task scheduling model. The model can be used to describe the execution process of edge cloud.

As a formal model with rich mathematical basis, Petri net can be widely used to describe the system with concurrent, asynchronous and distributed characteristics [29]. Therefore, Petri net is very suitable for describing a loosely coupled distributed system such as edge computing [30].

We model the basic components of edge cloud based on the requirements, then construct the task scheduling model. In the modeling process, only some basic concepts are introduced, the other concepts can refer to Ref [31]. In order to distinguish the input interface and output interface, the input interface is marked with superscript I, and the output interface is marked with superscript O.

A. MODELING RESOURSE

The task scheduling model abstracts all resources and information into as an individual $d_i = (it, st, RW_{\mathrm{i}})$, $it \in \{T, V, C, d\}$ indicates the object type described by the individual, T, V, C, d represent task, virtual machine, scheduling strategy and data package. st represents the location of the individual, such as the individual $d^T{}_{i,j} = (T, (i,j), RW_{\mathrm{i,j}})$ corresponds to task $TK_{i,j}$, $RW_{\mathrm{i,j}} = (T^{length}{}_{,i,j}, T^{com,}{}_{i,j}, T^{ram}{}_{i,j}, T^{bw}{}_{i,j}, T^{deadline}{}_{i,j})$ describes the required task length, computing power, memory capacity, network bandwidth and task deadline. When it is equal to V, $RW_{\mathrm{i}} = (V^{comp}{}_{i}, V^{ram}{}_{i}, V^{bw}{}_{i})$ indicates the computing power, memory capacity and network bandwidth capacity of virtual machine. When it is equal to C, $RW_{i,j}$ is the scheduling virtual machine of task $TK_{i,j}$. When it is equal to d, RW_{i} is the information carried by packets. The common packets in the system are abstracted into individual φ. If there is no special explanation, all individuals in task scheduling model are φ.

B. MODELING TASK

The task scheduling model of task $TK_{i,j}$ is shown in Figure 3. The specific operation process is as follows:

(1)Place p_s is used to store the execution mode of the task. Place p_{sd} stores the individuals of virtual machine

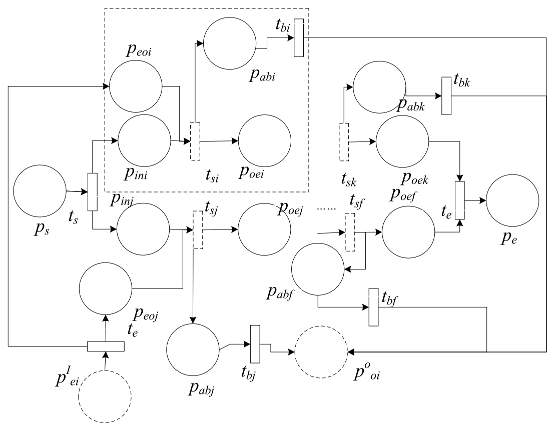


FIGURE 4. Modeling user request.

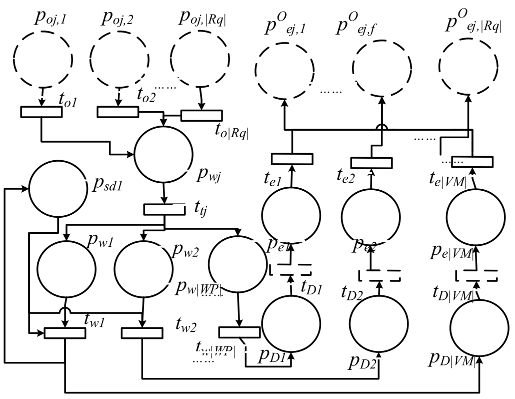


FIGURE 5. Modeling dispatching center.

$(M_0(TK_{i.j\bullet}p_{sd}) = d^c{}_{i,j} = (d^J_{i,j}, d^v_k))$ assigned by the task according to the above RCTSPO algorithm, assuming that the scheduling strategy assigns task $TK_{i.j}$ to the virtual machine VM_k).

(2) After obtaining the input parameter $p^I{}_{in}$, the task individual is put into the waiting queue $p^O{}_{ab}$ of virtual machine according to the task scheduling result. If the results of the task are fed back $(M(p^I{}_{eo}) \neq \emptyset)$, then fire the transition t_{ap} to make the task be in the termination position (p_e), and the execution result is sent to the output interface $p^O{}_{oe}$.

(3) p_d is used to control the deadline of task, $ct(p_d) = T^{deadline}{}_{i,j}$. If the task cannot realize the function within the deadline $(M(p_s) \neq \emptyset)$, transition t_{ad} is fired to make the task be in the timeout position.

C. MODELING USER REQUEST

The model of user request Rq_i is shown in Figure 4. The model mainly describes the tasks and their relationships in the user request, then dynamically outputs the tasks to be scheduled and the received execution results of the tasks.

(1) The transition t_s and place p_i are introduced to describe the beginning operation and location of user request, and the whole user request is initialized according to the characteristics of the task. The transition t_e and place p_e are introduced to describe the termination operation and location of user request respectively, and the corresponding output is synthesized according to the relationship between tasks. ${}^\bullet p_s = \emptyset$, $p_s^\bullet = t_s$, ${}^\bullet t_s = p_s$, $t_s^\bullet = \{p_{aj} | \forall\ TK_{i.k} \in TK_i, RL(TK_{i.k}, TK_{i.j}) \neq> \}$, $t_e^\bullet = p_e$, $p_e^\bullet = t_e$, ${}^\bullet t_e = \{p_{tj} | \forall\ TK_{i.k} \in TK_i, RL(TK_{i.j}, TK_{i.k}) \neq> \}$, $p_e^\bullet = \emptyset$, ${}^\bullet t_{dt} = \{p_a, p_c\}$, $t_{dt}^\bullet = p_{dt}$.

(2) Each task is modeled as a dotted box, p_{ini} and p_{oei} represent the beginning and termination of the task. p_{eoi} and p_{abi} represent the input of operating results and the output of task scheduling. Transitions t_{si}, t_{sj}, t_{sk}, t_{sf} represent the pages executed by tasks $TK_{i,i}$, $TK_{i,j}$, $TK_{i,k}$, $TK_{i,f}$, respectively.

(3) p^o_{oj} and p^I_{ej} are used to dynamically store the task set and results of user requests.

D. MODELING SCHEDULING CENTER

The scheduling center is used to connect user requests with virtual machines. The model is shown in Figure 5.

(1) Firstly, the scheduling strategy is used to assign the tasks. If a task $(M(\mathrm{p}_{oj,i}) \neq \emptyset)$ of user request Rq_i is submitted, then invoke transition t_{oi} to store the task in the place p_{wj} and wait for further classification. Transition t_{tj} is fired to configure all tasks to the cache according to their types. Place p_{Di} is introduced to represent the set of tasks realized by the virtual machine vm_i.

(2)The transition t_{Di} is introduced to represent the execution page of the virtual machine. If there is a virtual machine vm_i which feedbacks a set of tasks p^I_{ei} realized their function, the transition t_{ei} is fired to summarize the results to place p_{ej}. Then the transition t_{ej} is started and t_{fj} feedbacks the results to the corresponding interface of user request.

E. MODELING VIRTUAL MACHINE

The virtual machine realizes the function of the tasks and transmits the results to the scheduling center. The task scheduling model of virtual machine vm_k is shown in Figure 6. (1) p_D is used to store the queuing task of virtual machine. (2) If the virtual machine is idle $(M(p_s) \neq \emptyset)$, then fire the transition t_{st} to select a task $TK_{i,j}$ from the place p^I_{wt} according to the queuing order, and make it be in the running position $p_{int}(ct(p_{int}) = dw_{i,j}/ts_k)$. Transition t_{at} represents the execution process of tasks in virtual machine layer. (3) Releasing the virtual machine if the task execution is finished.

F. INSTANTIATION OF SPECIFIC REQUIREMENTS

The task scheduling model is modeled as follows:

(1) According to the properties and relations of tasks, the scheduling model of all tasks and user requests in the system is constructed, because the model mainly considers the execution process of tasks. Set $M_0(p_s) = \varphi$.

(2) According to the properties of virtual machine, the scheduling model of all virtual machines is constructed, and the initial resource distribution is set.

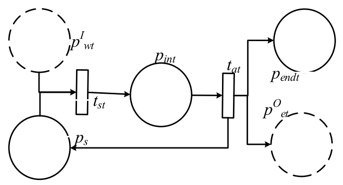


FIGURE 6. **Modeling virtual machine.**

(3) Based on the relationship between user request and virtual machine, we can construct the model of scheduling center, the same place and transition is merged.

Task scheduling model is mainly used to model virtual machine, task, and the relationship between tasks. Place, transition and interface describe the location, possible operation and input / output parameters of components.

Any $x \in (P \cup T), {}^{\bullet}x = \{y|y \in (P \cup T) \wedge (y,x) \in F\}$ and $x^{\bullet} = \{y|y \in (P \cup T) \wedge (x, y) \in F\}$ correspond to the input and output of x.

Because the place has time factor, the concept of waiting time is introduced to place. Waiting time TS is an attribute of the place, which is used to explain how long the system can use the individuals stored in the place. $TS(pi) = m$ indicates that the model must wait m time units before using the individual in place pi. A tuple $S = (M,TS)$ is called a state of task scheduling model, where M is a marking. And TS is the set of waiting time of all places under marking M. Initial state $S_0 = (M_0, TS_0)$, where TS_0 is a zero vector. The state is used to describe the resource distribution, such as the available resources, the position of each object.

Because $A_F(t)$ and $A_T(t)$ of model have the variables, the values of these variables are uncertain. $AT(t) < d_1, d_2, \ldots, dn >$ and $AF(p,t) < d_1, d_2, \ldots, dn >$ are the values got by replacing formulas $AT(t)$ and the predicates $AF(p,t)$ of input arc with individuals $d_1, d_2, \ldots, dn$. If $A_T(t) < d_1, d_2, \ldots, dn >=$ true, then $t < d_1, d_2, \ldots, dn >$ is called a feasible replacement of t. All the feasible replacements of t under S are denoted by set $ET(S)$. $H(S) = \{t < d_1, d_2, \ldots, dn > |{}^{\bullet}ti \cap {}^{\bullet}tj = \emptyset \wedge t < d_1, d_2, \ldots, dn >$ is a feasible replacements of $t\}$ is called the greatest firing set of S. Because each transition t may have several feasible replacements, $H(S)$ is not unique.

The process that S reaches S' by firing a feasible replacement $ti < d_1, d_2, \ldots, dn >$ of t_i is denoted by $S [ti < d_1, d_2, \ldots, dn > S'$. S' is called the reachable state of S. If there is a firing sequence $H_1, H_2, \ldots, Hk$ and state sequence $S_1, S_2, \ldots, Sk$, which makes $S\ [(H_1, \omega 1) > S_1[(H_2, \omega 2) > M_2 \ldots Sk - 1[(H_k, \omega k) > Sk$, then Sk is a reachable state from S. All the possibly reachable states of S are denoted by $R(S)$.

G. PROPERTY ANALYSIS

The task scheduling model mainly describes the functional requirements of the edge cloud, so it is necessary to analyze the structural correctness of task scheduling model. In addition, the components are marked in front of the node, task, transition and place. For example, the beginning transition t_s of task $TK_{i,j}$ is expressed as $TK_{i,j\bullet}t_s$.

Theorem 1: Let the task scheduling model be Ω, $R(\Omega)$ is the corresponding set of reachable states, then:

(1) $\forall TK_{i,j} \in Rq_i$, $\exists S' \in \mathrm{R}(\Omega)$, then $TK_{i,j\bullet}t_s \in FT(S')$

(2) If the system has K available virtual machines, $\forall 1 < k \leq K$, there is $\exists S$'$\in R(\Omega)$, which makes $|M'(VM_{i\bullet}p_D)| = 1$

Proof: (1) Recursive method, $\forall TK_{i,j} \in TK$, according to the relationships between $TK_{i,j}$ and other tasks in the execution process, it can be divided into two cases.

Case A: The forward set of task $TK_{i,j}$ is $Fork(TK_{i,j}) = \emptyset$. According to the modeling process of virtual machine, $TK_{i,j\bullet} p^I_{in} \in t^{\bullet}_s$. Because $t_s \in FT(S_0)$, there is $S_1 \in R(\Omega)$, which makes $|M_1(TK_{i,j\bullet}\ p^I_{in})| = 1$. Because ${}^{\bullet}TK_{i,j\bullet}t_s = TK_{i,j\bullet}p^I_{in}$, there is $S_2 \in R(S_1)$, $TK_{i,j\bullet}t_s \in FT(S_2)$. Let $S' = S_2$, we can get $\exists S' \in R(\Omega)$, which makes $TK_{i,j\bullet}t_s \in FT(S')$ when $Fork(TK_{i,j}) = \emptyset$.

Case B: The forward set of $TK_{i,j}$ is $Fork(TK_{i,j}) \neq \emptyset$. Let $\forall TK_{i,k} \in Fork(TK_{i,j})$, $\exists S' \in R(\Omega)$, which makes $TK_{i,k\bullet}t_s \in FT(S')$, we can get that $TK_{i,j}$ meets the sub proposition (1). Let $Fork(TK_{i,\mathrm{j}}) = \{TK_{i,k}, TK_{i,f}, \ldots, TK_{i,g}\}$. According to the assumption of the proposition, all forward tasks of $TK_{i,j}$ are possible to be executed. So there is $S_1 \in R(\Omega)$, which makes $|M_1(TK_{i,k\bullet}p^O{}_{oe})| = |M_1(TK_{i,f\bullet}p^O{}_{oe})| = \ldots = |M_1(TK_{i,g\bullet}p^O{}_{oe})| = 1$.

Because $RL(TK_{i,k}, TK_{i,j}) = \ldots = RL(TK_{i,g}, TK_{i,j}) =>$. According to the modeling process of the relationships between tasks, we can get that there is a corresponding transition in the model, which is used to summarize the execution results of $TK_{i,k}$,..., $TK_{i,g}$ to $TK_{i,j}$. That is, there is $S_2 \in R(S_1)$ that makes $|M_2(TK_{i,j\bullet}p^I_{in})| \neq 0$. Because ${}^{\bullet}TK_{i,j\bullet}t_s = TK_{i,j\bullet}p^I_{in}$, there is $S_3 \in R(S_2)$, which makes $TK_{i,j} \bullet t_s \in FT(S_3)$. *Set* $S' = S_3$, $\exists S' \in R(\Omega)$, which makes $TK_{i,j} \bullet t_s \in FT(S')$ if $Fork(TK_{i,j}) \neq \emptyset$.

To sum up, the sub proposition (1) is proved.

Similarly, the sub proposition (2) is also established.

Theorem 1 shows that any virtual machine can be invoked and a task may be executed in task scheduling model. That is, all virtual machines and tasks can be invoked in the model.

Theorem 2: Let the task scheduling model be Ω, $\mathrm{R}(\Omega)$ is the set of reachable states. $\forall\ TK_{i,j} \in TK$, $\forall\ VM_k \in VM$, if the scheduling algorithm assigns $TK_{i,j}$ to VM_k, then:

(1)$\exists S$'$\in R(\Omega)$, which makes $d^J{}_{i,j} \in M(VM_{k\bullet}p^I_D)$

(2)$\exists S' \in R(\Omega)$, which makes $d^J{}_{i,j} \in M(p^O{}_{ej,i})$

Proof: Because $TK_{i,j}$ is assigned to VM_k by using the scheduling algorithm, according to the task scheduling model, we can get $M_0(TK_{i,j\bullet}p_{sd}) = d^C{}_{i,j} = (d^J{}_{i,j}, d^v{}_k)$. According to Theorem 1, there is $S_1 \in R(\Omega)$, which makes $TK_{i,j\bullet}t_s \in FT(S')$. Because $TK_{i,j\bullet}t^{\bullet}_s = p^o{}_{ab}$. So there is $S_2 \in R(S_1)$, which makes $d^C_{i,j} \in M_2(TK_{i,j\bullet}p^o{}_{ab}) = M_2(Rq_{i\bullet}\ p_{abi})$. Because $Rq_{i\bullet}p^{\bullet}_{abi}= Rq_{i\bullet}t_{bi}$,and $Rq_{i\bullet}t^{\bullet}_{bi}= Rq_{i\bullet}p^o{}_{oi}$. So there is $S_3 \in R(S_2)$, which makes $d^C_{i,j} \in M_3(Rq_{i\bullet}p^o{}_{oi}) = M_3(p_{oi,i})$. Because $p^{\bullet}_{oi,i} = t_{oi}$ *and* $t^{\bullet}_{oi} = p_{wj}$. There is $S_4 \in R(S_3)$, which makes $d^C_{i,j} \in M_4(p_{wj})$. Because $p^{\bullet}_{wj} = t_{tj}$ *and* $t^{\bullet}_{tj} = \{p_{w1},$

TABLE 1. Parameters of task and virtual machine.

Attributes/ Unit	Task	Virtual Machine
memory/MB	[1024,2048]	[1024,3072]
computing power / (million instructions per second)	[100,600]	[500,1000]
bandwidth /(MB/S)	[1000,3000]	[2000,4000]
length / (million instructions)	[3024,11048]	none
time limit /(MS)	[90,330]	none

$p_{w2}, \ldots\}$. There is $S_5 \in R(S_4)$, which makes $d^J{}_{i,j} \in M_5(p_{wk})$. Because $p^\bullet_{wk} = t_{wk}$ and $t^\bullet_{wk} = p_{Dk}$. There is $S_6 \in R(S_5)$, which makes $d^J{}_{i,j} \in M_6(p_{Dk}) = M_6(VM_{k\bullet}p^I_D)$. Set $S' = S_6$, sub proposition (1) is proved.

Similarly, the sub proposition (2) is also established.

Theorem 2 shows that task scheduling model can correctly describe the process that tasks invoke the virtual machine and feedback the execution results of virtual machine. Therefore, task scheduling model can effectively describe the execution logic between tasks and virtual machines.

VI. EXPERIMENTAL SIMULATION AND ANALYSIS

In this section, the performance and applicability of the proposed RCTSPO algorithm will be evaluated.

A. EXPERIMENT SETUP

Due to the lack of a unified standard library, Cloudsim is used to automatically generate task scheduling requirements for the edge cloud, including task set, user request set, virtual machine set, etc [32]. One of the advantages of using Cloudsim is that we can set different parameters based on the actual requirement. In the *dataCenterBroker* class, we rewrites the *bindcloudlettovm* (*cloudletlist, vmlist*) method to realize the core function of RCTSPO algorithm.

According to the task scheduling requirements, we randomly generate the parameters of the task and virtual machine, which are shown in Table 1.

The proposed RCTSPO algorithm aims to maximize the profit of resource constrained task scheduling in edge cloud. This paper intends to evaluate the algorithm from the total time, reliability, profit and load balancing of task scheduling.

Total time of task scheduling: *Makespan* represents the total time of task scheduling. *Makespan* is equal to the maximum working time of all virtual machines.

Reliability of task scheduling: The proportion of tasks completed in $T^{deadline}$ divided by the total tasks.

The profit of task scheduling: Calculating the total profit of task scheduling according to Formula (7), that is, the sum of all profits for realizing the tasks [28].

$$\text{Total profit} = \sum_{i=0,j=0,k=0}^{|Rq|,|Rq_i|,|VM|} profit_{i,j,k} \qquad s.t. Time_{i,j} < T^{deadline}_{i,j} \tag{7}$$

Load balancing: The standard deviation of the working time of the virtual machines represents the load balancing of virtual machines [33]. According to Formula (8), it can get:

$$\delta = \sqrt{\frac{1}{|\mathrm{VM}|}\sum\nolimits_{j=1}^{|\mathrm{VM}|}(PT_j - \mathrm{avg}(PT))^2} \tag{8}$$

PT_j represents the working time of virtual machine J in task scheduling, $avg(PT)$ is the average time of all virtual machines. The smaller the standard deviation, the closer the working time of different virtual machines, and the better the load balance of task scheduling.

The comparison algorithms used in this paper are FCFS (First Come First Serve), MIN-MIN [34], ACO (Ant Colony Algorithm), GA (Genetic Algorithm) and KM algorithm. It is assumed that task scheduling can be completed only if the resource constraints are satisfied.

B. IMPACT OF CLUSTERING

Experiments 1 and 2 are done to get the optimal clustering value K. In order to weaken the influence of the unit price of different resource types on profits, we set $a = b = c$ in these two experiments.

1) IMPACT OF CLUSTERING K VALUE

Experiment 1: The RCTSPO algorithm proposed in this paper converts the ratio of the number of tasks to the number of virtual machines into the K value of clustering. In order to evaluate the performance of RCTSPO algorithm when the number of virtual machines is fixed while the number of tasks is dynamically changing (The values of clustering K are different). The completion time, reliability, load balancing and total profit of task scheduling in the evaluation criteria are compared to get the best clustering value K.

The experimental steps of Experiment 1 are as follows.

Step 1: Assuming that the number of virtual machines is 10, and the number of tasks is 30, 40, 50, 60, 70, 80, 90, 100 (that is, cluster value K is 3, 4, 5, 6, 7, 8, 9, 10).

Step 2: Setting the initial profit weight to $a = b = c = 3$. The purpose of this experiment is to find the best clustering value K, so the profit weight is the same by default.

Step 3: Comparing the performance of RCTSPO algorithm and FCFS, MIN-MIN, ACO, GA and KM algorithm in task scheduling of edge cloud.

Step 4: Calculating the completion time, reliability, total profit and load balancing under task scheduling strategies.

The results of Experiment 1 are shown in Figure 7(a)-(d), we can observe: (1) The number of virtual machines remains unchanged when the number of tasks increases gradually, and the deadline of task is fixed. The running time of virtual machine increases gradually, Figure 7(a) and Figure 7(c) show an upward trend, and Figure 7(b) shows a downward trend. That is because the waiting time is too long, it is easy to increase the number of tasks exceed the deadline, which makes the reliability of task scheduling

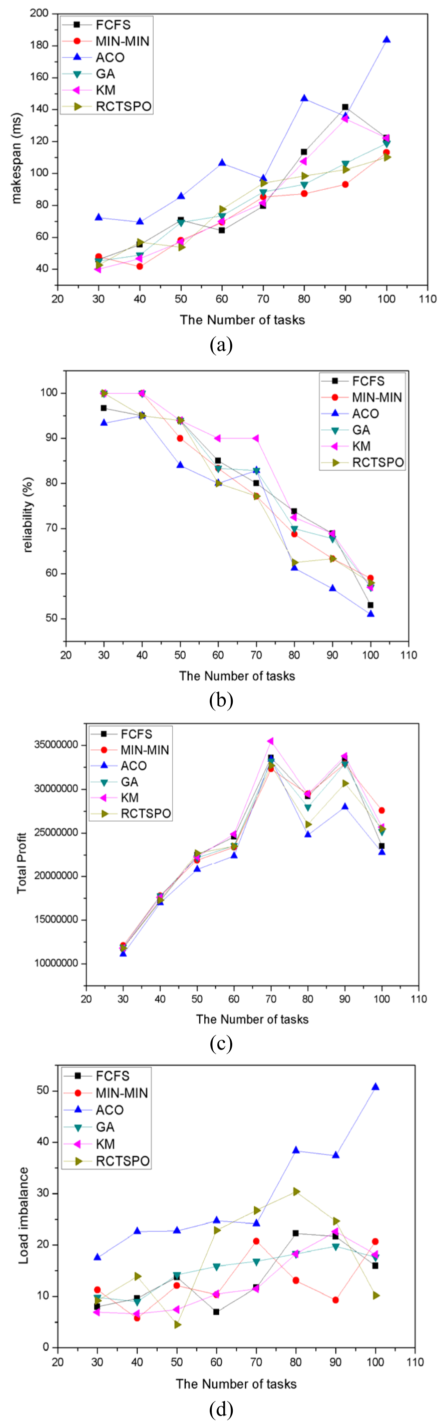


FIGURE 7. Optimal clustering k value for task scheduling.

decrease, while the task scheduling time and profit increase. (2) The load balancing of task scheduling is fluctuating. When the number of tasks is 50, the load balance of task scheduling is optimal. (3) The clustering value K is 5, RCTSPO is better than the compared algorithm in completion time, reliability, total profit and load balancing of task scheduling, and the best clustering value K is 5.

2) IMPACT OF THE NUMBER OF TASKS

Experiment 2: Because RCTSPO algorithm includes the K-means clustering algorithm, Experiment 2 is done to get the optimal clustering value K. In order to distinguish the results of the clustering algorithm in task scheduling, we will design the experiment to analyze the impact of the number of tasks on the performance of task scheduling.

The steps of Experiment 2 are as follows:

Step 1: With the optimal value K ($k = 5$), the weight of profit is initialized as: $a = b = c = 3$.

Step 2: The number of tasks is 10, 100, and 1000, the number of virtual machines is 1/5 of the number of tasks.

Step3: Compare RCTSPO algorithm with FCFS, MIN-MIN, ACO, GA, and KM on completion time, reliability, total profit, and load balancing performance of task scheduling.

The results of Experiment 2 are shown in Figure 8(a)-(d): When the task is 10, RCTSPO algorithm does not show the great advantages in completion time, reliability, total profit and load balancing of task scheduling. When the number of tasks increases to 100 and 1000, the performance of RCTSPO algorithm is better than other algorithms. This is because the clustering algorithm of RCTSPO is suitable for large data volume. When the data volume is small, the clustering results are not obvious. It can draw that RCTSPO algorithm is more suitable for the large number of tasks.

C. PROFIT MAXIMIZATION OF TASK SCHEDULING

1) IMPACT OF WEIGHT ON TASK SCHEDULING PROFIT

Experiment 3: Because user must pay for cloud provider in task scheduling, we add the profit weights in Formula (3). The profit weights of *ram*, *bw*, and *comp* in Formula (3) are *a, b, c*, and $a + b + c = 9$. We will design an experiment to find the optimal profit weight of task scheduling under different profit weights of RCTSPO.

The steps of Experiment 3 are as follows:

Step 1: The number of tasks is set to 100, 200, 300, 400, and 500, respectively. The number of virtual machines is 1/5 of the number of tasks (that is, the optimal clustering value K).

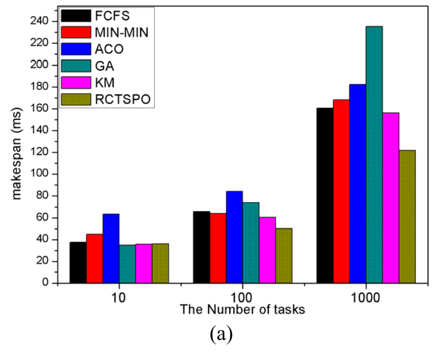
Step 2: The profit weights are set to $a = b = c = 3; a = 7, b = c = 1; a = c = 1, b = 7$, and $a = b = 1, c = 7$.

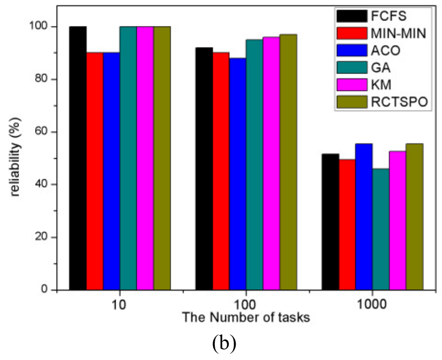
Step 3: RCTSPO algorithm is used to calculate the task scheduling strategy respectively. Calculating the total profit under all task scheduling strategies.

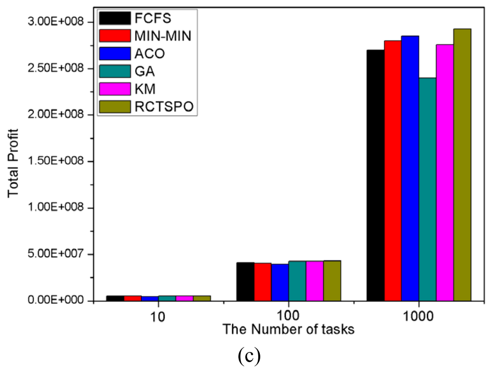
The results of Experiment 3 are shown in Figure 9, we can observe: The profit of RCTSPO algorithm also shows an upward trend with the number of tasks increasing. The profit of task scheduling under different profit weights increases with the number of tasks increasing. When the profit weight is $a = c = 1$ and $b = 7$, the profit of task scheduling is larger than others. This is because the bandwidth resources are more important than memory and computing power.

2) THE RESULTS OF DIFFERENT ALGORITHMS

Experiment 4: The optimal profit weight of task scheduling based on Experiment 3 is $a = c = 1$, $b = 7$. In order to







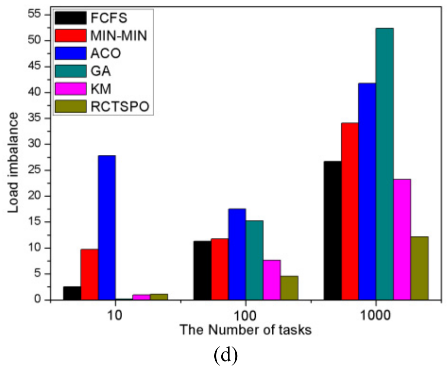


FIGURE 8. **Impact of the number of tasks on task scheduling.**

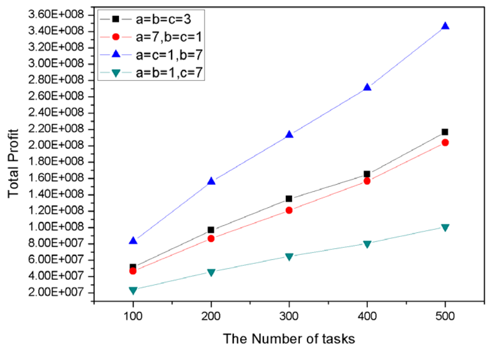


FIGURE 9. **Total profit of different tasks.**

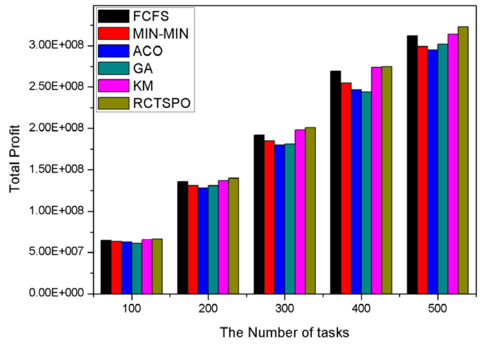


FIGURE 10. **Profit of different algorithms.**

verify that RCTSPO algorithm has better performance in task scheduling than the compared algorithm under the best profit weight and the best clustering value K.

The steps of Experiment 4 are as follows:

Step 1: The number of tasks is 100, 200, 300, 400, 500. The number of virtual machines is 1/5 of the number of tasks (that is, the optimal clustering value K). The profit weight is $a = b = 1, c = 7$.

Step 2: RCTSPO algorithm and FCFS, MIN-MIN, ACO, GA and KM are used to calculate the task scheduling strategy respectively.

Step 3: Calculating the total profit under all task scheduling strategies.

The results of Experiment 4 are shown in Figure 10, we can observe: (1) The total profit of task scheduling of all algorithms increases with the number of tasks increasing. As the number of tasks increases, the RCTSPO algorithm gets more profit than the compared algorithms. (2) The increase of the number of completed tasks makes the total profit increase. When the number of tasks is small, the difference of total profit of each scheduling algorithm is not large. When the number of tasks increases gradually, the number of completed tasks is different by using the different algorithms, and the difference between them is larger. (3) RCTSPO and KM algorithms can get the higher total profit of task scheduling, followed by FCFS and MIN-MIN algorithms, and ACO and GA algorithms are the less. The total profit of task scheduling using RCTSPO algorithm is always greater than that of KM algorithm, which proves the advantage of the proposed RCTSPO algorithm.

D. LOAD BALANCING OF TASK SCHEDULING

1) LOAD BALANCING UNDER DIFFERENT PROFIT WEIGHTS

Experiment 5: This paper designs the profit weight in objective function, which is the unit price of resource.

Different profit weights play an important role in resource constrained task scheduling. We will analyze the optimal profit weight of RCTSPO algorithm in load balancing of task scheduling.

The steps of Experiment 5 are as follows:

Step 1: The number of tasks is 100, 200, 300, 400, 500. The number of virtual machines is 1/5 of the number of tasks (that is, the best clustering value K).

Step 2: The profit weight is $a = b = c = 3$; $a = 7$, $b = c = 1$; $a = c = 1, b = 7$ and $a = b = 1, c = 7$, respectively.

Step 3: RCTSPO algorithm is used to calculate the task scheduling strategy. Calculating the total profit under all task scheduling strategies.

The results of Experiment 5 are shown in Figure 11, we can observe: The profit weight is $a = b = c = 3$; $a = 7, b = c = 1$; $b = 7, a = c = 1$; $a = b = 1, c = 7$. The average standard deviation of execution time on virtual machine is 6.27, 3.57, 8.29 and 6.12 respectively. It can get that the performance of RCTSPO algorithm in load balancing of task scheduling is the best when the profit weight is $a = 7, b = c = 1$, which is better than that of other profit weights. It can get that RCTSPO algorithm focuses more on memory and load balancing in task scheduling.

2) LOAD BALANCING OF DIFFERENT ALGORITHMS

Experiment 6: According to Experiment 5, the optimal profit weight of load balancing on task scheduling is $a = 7$, $b = c = 1$. In order to verify that RCTSPO algorithm has better performance in task scheduling and load balancing than the compared algorithm with the best profit weight and the best clustering value K.

The steps of Experiment 6 are as follows:

Step 1: The number of tasks is 100, 200, 300, 400, 500. The number of virtual machines is 1/5 of the number of tasks (that is, the best clustering value K). The profit weight is set to $a = 7, b = c = 1$.

Step 2: RCTSPO algorithm and FCFS, MIN-MIN, ACO, GA and KM are used to calculate the task scheduling strategy respectively.

Step 3: Calculating the load balancing under all task scheduling strategies.

The results of Experiment 6 are shown in Figure 12, we can observe: (1) The workload imbalance of the compared algorithm increases with the number of tasks increasing, but RCTSPO algorithm shows a downward trend. It can get that RCTSPO algorithm is suitable for task scheduling with a large number of tasks in terms of load balancing. RCTSPO algorithm is better than compared algorithm in load balancing of task scheduling. (2) As the KM algorithm aims to find the maximum weight matching under perfect matching. Therefore, *KM* algorithm is better than other algorithms in load balancing of task scheduling. In RCTSPO algorithm, resource constraints are added, tasks are clustered and requirements are matched to avoid the idle resources. (3) It can get that RCTSPO algorithm has the best effect on load balancing

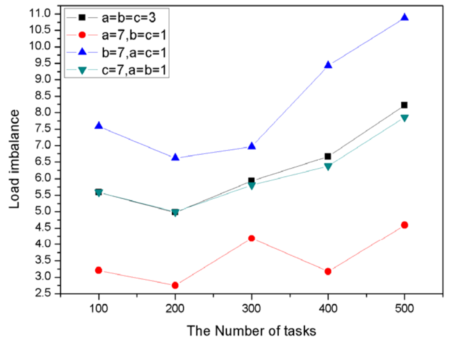


FIGURE 11. Load balancing under different profit weights.

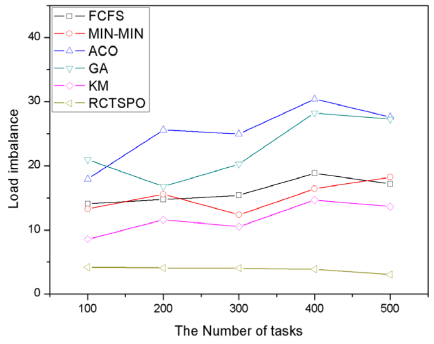


FIGURE 12. Load balancing of different algorithms.

in task scheduling. Experiment results show that RCTSPO algorithm is superior to the other algorithms in load balancing of task scheduling.

E. STATE SPACE ANALYSIS

Experiment 7: This paper verifies the properties of task scheduling model by using the related tools of Petri net, including the correctness of user request, task scheduling process and so on. Therefore, it is necessary to analyze the changing rules of the state space of the task scheduling model.

The purpose of Experiment 7 is to analyze the changing rules of the state space of task scheduling model. The specific experimental steps are as follows.

(1) Randomly generating 200 user requests, each use request has 10-30 tasks, the attributes and relationships between tasks are randomly generated. 200 user requests are divided into 4 groups (each group has 50 user requests). Randomly generating 100 virtual machines.

(2) Taking 10 virtual machines as the initial resources and adding 10 virtual machines (10VM, 20VM, 30VM, 40VM, 50VM) each time. The task scheduling model of each user request is constructed according to the attributes of task, user request and virtual machine respectively. Finally, we will calculate the size of state space of the model, as shown in Figure 13(a).

(3) Let each group have 25 virtual machines, the system will select 10 user requests to construct the initial requirement, then add 10 user requests (10U, 20U, 30U, 40U, 50U) each time. Constructing the task scheduling model for each

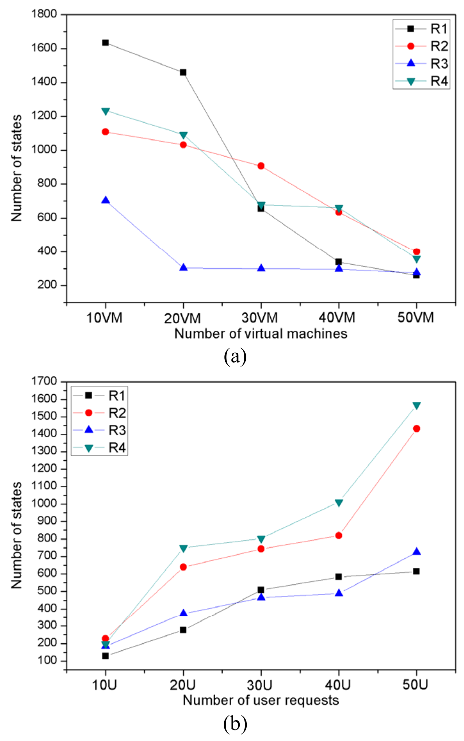


FIGURE 13. **The state space of task scheduling model.**

user request and calculating the size of state space of the constructed model, which is shown in Figure 13(b).

The results of Experiment 7 are shown in Figure 13, we can draw: (1) As shown in Figure 13(a), the number of reachable states of the model decreases with the increase of virtual machines. The reason is that the increase of virtual machines makes more tasks run concurrently, thus reducing the number of states in the task scheduling model. The increase of the attributes of the virtual machine also affects the reduction speed of state space. Such as the state space of *R1* only reduces 10.65% at 20VM, while the state space of *R1* is reduced by 55% at 30VM. The reason is that the first 10 virtual machines provides the fewer resources. (2) As shown in Figure 13(b), when the number of user requests gradually increases, the number of reachable states of task scheduling model tends to increase. The reason is that new user request makes the number of tasks executed on the virtual machine increase. In addition, the attributes of the task are different, so the increase of the number of user requests will make more tasks asynchronously execute, which increases the reachable states of the model. In addition, the number of states is related to the execution process. Such as the number of states of *R1* and *R3* is less, while the number of *R2* and *R3* is more. It gets that the number of parallel tasks in the first two user requests are more, which make the number of reachable states increase slowly.

According to the results of Experiments, when the value *K* of task clustering is 5, RCTSPO algorithm has better in time, reliability, profit and load balance of task scheduling. With the number of tasks increases, the scheduling results are better. It can draw that RCTSPO algorithm can greatly improve the scheduling performance. At the same time, the influence of different unit price of resource on task scheduling profit and load balance is analyzed. It can draw that RCTSPO algorithm is applicable to the case which has a large number of tasks in task scheduling.

VII. SUMMARY AND FUTURE EXPECTATION

In this paper, a resource constrained cloud task scheduling algorithm is proposed to solve the problems of resource constraints, and profit optimization in the task scheduling process of edge cloud. Based on the task scheduling strategy calculated by using the proposed algorithm, the task scheduling model of edge cloud is constructed to describe its business process and characteristics. First, RCTSPO algorithm uses the improved *K*-means algorithm to cluster tasks and virtual machines. It can reduce the search scope and achieve global optimization, the resource constraints is also considered in the task scheduling process to achieve load balancing. Second, RCTSPO algorithm gets the best matching using the KM algorithm and profit matrix to improve the profit of task scheduling. Third, according to the task scheduling strategy and business process, a formal model of task, virtual machine, user request and scheduling center is constructed based on Petri net, which is used to analyze the related properties of task scheduling process. The simulation results show that the proposed RCTSPO algorithm has better effectiveness on load balancing and profit.

The algorithm in this paper does not consider the resource utilization and energy consumption of task scheduling. In the future work, we will study the resource utilization and energy consumption of task scheduling.

REFERENCES

[1] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[2] T.-D. Nguyen, Y. Kim, D.-H. Kim, and E.-N. Huh, "A proposal of autonomic edge cloud platform with CCN-based service routing protocol," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 802–809.

[3] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[4] Y. Zhang, X. Lan, Y. Li, L. Cai, and J. Pan, "Efficient computation resource management in mobile edge-cloud computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3455–3466, Apr. 2019.

[5] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[6] Y. Zhao, R. Calheiros, G. Gange, J. Bailey, and R. Sinnott, "SLA-based profit optimization resource scheduling for big data analytics-as-a-service platforms in cloud computing environments," *IEEE Trans. Cloud Comput.*, early access, Dec. 27, 2018, doi: 10.1109/TCC.2018.2889956.

[7] NSF Workshop. (2016). *Looking Beyond the Internet*. [Online]. Available: https://lookingbeyondtheinternetblog.wordpress.com/

[8] OpenStack. (2018). *Cloud Edge Computing: Beyond the Data Center*. [Online]. Available: https://www.openstack.org/edge-computing/ cloud-edge-computing-beyond-the-data-center?lang=en_US

[9] Gartner. (2019). *Gartner Top 10 Strategic Technology Trends for 2020*. [Online]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020

[10] G. Honghao, H. Wanqiu, and D. Yucong, "The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: A QoS prediction perspective.," *ACM Trans. Internet Technol.*, 2020, doi: 10.1145/3391198.

[11] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[12] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed Petri nets," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1216–1228, Oct. 2017.

[13] C.-M. Li, Y.-R. Ho, W.-L. Chen, C.-H. Lin, M.-Y. Chen, and Y.-Z. Chen, "Nasogastric tube dislodgment detection in rehabilitation patients based on fog computing with warning sensors and fuzzy Petri Net," *Sensors Mater.*, vol. 31, no. 1, pp. 117–130, Jan. 2019.

[14] S. Zahra, M. Alam, Q. Javaid, A. Wahid, N. Javaid, S. U. R. Malik, and M. K. Khan, "Fog computing over IoT: A secure deployment and formal verification," *IEEE Access*, vol. 5, pp. 27132–27144, 2017.

[15] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.

[16] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge computing in the industrial Internet of Things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 44–51, Feb. 2018.

[17] M. Zahid, N. Javaid, K. Ansar, K. Hassan, and M. Waqas, "Hill climbing load balancing algorithm on fog computing," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. (PGCIC)*, Oct. 2018, pp. 238–251.

[18] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[19] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 89–99, Jun. 2019.

[20] X. Wei, J. Fan, T. Wang, and Q. Wang, "Efficient application scheduling in mobile cloud computing based on MAX–MIN ant system," *Soft Comput.*, vol. 20, no. 7, pp. 2611–2625, Jul. 2016.

[21] H. Aziza and S. Krichen, "Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing," *Computing*, vol. 100, no. 2, pp. 65–91, Feb. 2018.

[22] Y. Chen, S. Deng, H. Ma, and J. Yin, "Deploying data-intensive applications with multiple services components on edge," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 426–441, Apr. 2020.

[23] X Yan, M Razeghi-Jahromi, A Homaifar, B. A. Erol, and E. Tunstel, "A novel streaming data clustering algorithm based on fitness proportionate sharing," *IEEE Access*, pp. 1960–1965, 2017.

[24] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 391–401, Apr. 2019.

[25] M. Ramalingam and R. Thangarajan, "Mutated k-means algorithm for dynamic clustering to perform effective and intelligent broadcasting in medical surveillance using selective reliable broadcast protocol in VANET," *Comput. Commun.*, vol. 150, pp. 563–568, Jan. 2020.

[26] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for IoT devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.

[27] K. M. Kumar and A. R. M. Reddy, "An efficient K-means clustering filtering algorithm using density based initial cluster centers," *Inf. Sci.*, vols. 418–419, nos. 286–301, pp. 0–36, Jul. 2017.

[28] T. Wang, X. Wei, T. Liang, and J. Fan, "Dynamic tasks scheduling based on weighted bi-graph in mobile cloud computing," *Sustain. Comput., Informat. Syst.*, vol. 19, pp. 214–222, Sep. 2018.

[29] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*. Berlin, Germany: Springer-Verlag, 2003.

[30] R. Wiśniewski, A. Karatkevich, and M. Wojnakowski, "Decomposition of distributed edge systems based on the Petri nets and linear algebra technique," *J. Syst. Archit.*, vol. 96, pp. 20–31, Jun. 2019.

[31] G. Fan, L. Chen, H. Yu, and D. Liu, "Modeling and analyzing dynamic fault-tolerant strategy for deadline constrained task scheduling in cloud computing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 4, pp. 1260–1274, Apr. 2020.

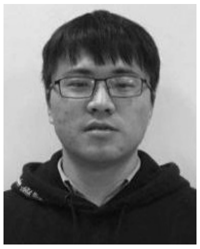
[32] E. Ataie, R. Entezari-Maleki, S. E. Etesami, B. Egger, D. Ardagna, and A. Movaghar, "Power-aware performance analysis of self-adaptive resource management in IaaS clouds," *Future Gener. Comput. Syst.*, vol. 86, pp. 134–144, Sep. 2018.

[33] Y. Qiao and G. V. Bochmann, "Load balancing in peer-to-peer systems using a diffusive approach," *Computing*, vol. 94, nos. 8–10, pp. 649–678, Sep. 2012.

[34] S. Elsherbiny, E. Eldaydamony, M. Alrahmawy, and A. E. Reyad, "An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment," *Egyptian Informat. J.*, vol. 19, no. 1, pp. 33–55, Mar. 2018.

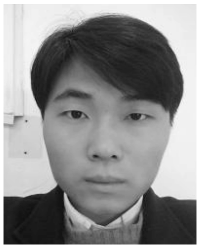
LIQIONG CHEN (Member, IEEE) received the Ph.D. degree from the East China University of Science and Technology (ECUST), in 2009. She is currently an Associate Professor with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. Her research interests include formal methods for complex software systems and hardware/software codesign of embedded systems.

KUN GUO received the B.S. degree in computer science from the Suqian College, in 2018. He is currently pursuing the master's degree with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. His research interests include cloud computing, edge computing, and task scheduling.



GUOQING FAN received the B.S. degree in computer science from the Tongda College, Nanjing University of Posts and Telecommunications, in 2018. He is currently pursuing the master's degree with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. His research interests include cloud computing and service recommendation.

CAN WANG received the bachelor's degree in management from the Wuchang Institute of Technology, in 2019. He is currently pursuing the master's degree with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. His research interest includes python code analysis.



SHILONG SONG received the B.S. degree in computer science from the Luoyang Institute of Science and Technology, in 2019. He is currently pursuing the master's degree with the School of Computer Science and Information Engineering, Shanghai Institute of Technology. His research interests include soft program analysis, automatic repair of software defects, and software optimization.