Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5520455, 13 pages
https://doi.org/10.1155/2021/5520455

Research Article

Optimal Workload Allocation for Edge Computing Network Using Application Prediction

Zhenquan Qin,[1,2] **Zanping Cheng**,[1,2] **Chuan Lin**,[3] **Zhaoyi Lu**,[1,2] **and Lei Wang**[1,2]

[1]*Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, China*
[2]*School of Software, Dalian University of Technology, 116620, China*
[3]*Software College, Northeastern University, 110819, China*

Correspondence should be addressed to Chuan Lin; chuanlin1988@gmail.com

Received 8 January 2021; Revised 16 February 2021; Accepted 9 March 2021; Published 25 March 2021

Academic Editor: Varun Menon

By deploying edge servers on the network edge, mobile edge computing network strengthens the real-time processing ability near the end devices and releases the huge load pressure of the core network. Considering the limited computing or storage resources on the edge server side, the workload allocation among edge servers for each Internet of Things (IoT) application affects the response time of the application's requests. Hence, when the access devices of the edge server are deployed intensively, the workload allocation becomes a key factor affecting the quality of user experience (QoE). To solve this problem, this paper proposes an edge workload allocation scheme, which uses application prediction (AP) algorithm to minimize response delay. This problem has been proved to be a NP hard problem. First, in the application prediction model, long short-term memory (LSTM) method is proposed to predict the tasks of future access devices. Second, based on the prediction results, the edge workload allocation is divided into two subproblems to solve, which are the task assignment subproblem and the resource allocation subproblem. Using historical execution data, we can solve the problem in linear time. The simulation results show that the proposed AP algorithm can effectively reduce the response delay of the device and the average completion time of the task sequence and approach the theoretical optimal allocation results.

1. Introduction

In recent years, the popularity of mobile devices, such as smart phones or tablet computers, has a huge impact on mobile and wireless networks, which has triggered the challenges of global mobile networks. In addition, more IoT devices have begun to be promoted and used on a large scale. IoT devices and their applications have produced a lot of data and network traffic. A new report by International Data Corporation (IDC) estimates that there will be 41.6 billion IoT devices connected to the Internet, which will generate 79.4 zettabytes (ZB) of data in 2025 [1]. However, the current infrastructure uses cloud computing as the underlying computing support, which is a centralized processing model. With the trend of exponential growth of IoT devices, the transmission of terminal computing tasks and storage tasks to the cloud computing will lead to problems such as high cloud service delay, high bandwidth occupancy, and security privacy in the big data network. Faced with these problems, edge computing emerges as the supplement of cloud computing. Different from the centralized computing model of cloud computing, edge computing is a distributed computing processing model. By deploying edge servers on the network edge, mobile edge computing provides resources and services to minimize the cloud load. Meanwhile, mobile edge computing also complements cloud computing by enhancing IoT user services for delay-sensitive applications [2].

Although the edge servers can provide computing or storage resources in edge computing network [3], the computing or storage resources of edge servers are limited. Intelligent and practical algorithms are needed to schedule and allocate resources to ensure the optimal resource allocation. Thus, efficient workload allocation strategy is extremely important, which directly determines the utilization efficiency of resources. The current academic research also focuses on how to allocate the load reasonably from the

existing resources of edge servers, dynamically according to the application of access devices and network conditions [4]. However, the resource allocation for different types of applications also affects the response delay of different types of requests. Since the computing size per request for different applications are different, the computing capacity of edge server should be optimally allocated for different applications in order to reduce the response delay of all applications of user equipment.

To solve the above problem, this paper proposes a workload allocation strategy that can be applied in real scenarios to make better use of resources in edge network and reduce the response delay of edge devices. Through the analysis of the maximum information entropy, Song et al. [5] found that the network activity information of terminal equipment has 93% predictability. Therefore, in the direction of improving the network system, the analysis of mobile data has great potential. On the one hand, mobile traffic information is very important to clearly describe how mobile terminal users use access network resources. On the other hand, the analysis rules of mobile traffic information can better guide the deployment of network system and the allocation of resources. To enable future networks to achieve "no perception" of latency, this paper first proposes AP algorithm to minimize the average task completion time, which takes into account transmission delay and computing delay. The main contributions of this paper are as follows:

(i) We innovatively put forward the concept of application prediction in edge computing architecture to solve edge workload allocation problem

(ii) In the application prediction model, the AP algorithm based on LSTM uses historical data to predict the possible tasks of future access device. By application prediction, the virtual machine is loaded on the edge server in advance, which reduces the service response delay. F-Measure parameter is weighted harmonic average of precision and recall, which is introduced to evaluate the accuracy of AP algorithm

(iii) Based on the prediction results of AP algorithm, workload allocation problem is divided into two subproblems, which are the task assignment problem and the resource allocation subproblem within the edge server

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes the system, transmission delay, and computing delay models and then formulates the problem. Section 4 describes the detailed algorithm for solving workload allocation problem. Simulation studies are conducted to demonstrate the performance of the proposed algorithm in Section 5. Section 6 concludes this paper.

2. Related Works

On the edge server side, the edge workload allocation scheme refers to how to best allocate existing resources to complete the current task when task offloading by different devices forms a task queue.

Many existing work studies computing offloading decision and involved resource allocation to optimize the time delay or energy consumption of devices in edge scenarios. Mao et al. in [6] and Lyu et al. in [7] designed the task load allocation algorithm based on the Lyapunov optimized framework and implemented the offloading decision only based on the current system state. Chen et al. in [8] proposed the joint optimization of offloading decision and allocation of communication and computing resources. Semiqualitative relaxation and a novel random mapping method were adopted to effectively solve the NP-hard problem, so as to minimize the total system cost. Du et al. in [9] designed a fairness guarantee algorithm for computing offloading and resource allocation, which jointly optimized the offloading decision and the allocation of computing resources, transmission power, and radio bandwidth, while ensuring user fairness and maximum tolerable delay. Liu et al. in [10] conducted an in-depth study on the energy consumption, payment cost, and delay of the offloading process in the edge computing system by using queuing theory. Tan et al. in [11] proposed an online task assignment and scheduling algorithm, adding a weight to the response time of each task to represent its delay sensitivity, thus assigning greater weight to delay-sensitive tasks so as to provide them with higher priority. Reference [12] proposed a new mobility management scheme in the ultradense MEC network, using Lyapunov optimization and the theory of the dobby bandit to optimize the delay of the equipment under the constraint of long-term energy consumption.

Some of the existing work is also studying the horizontal collaboration between edge servers. Jia et al. in [13] pointed out that cooperation between edge servers can balance computing load and generate huge performance improvements, which is far more than the situation of only performing tasks on edge servers alone. Xiao and Krunz in [14] proposed a collaborative task forwarding strategy to improve the user's QoE. Fan and Ansari in [15] designed a task type based allocation scheme to minimize response time by determining the computing destination and allocated resources for each task.

In 2019's section, many new strategies for reducing delay in resource allocation are also proposed. In [16], Dlamini and Gambin proposed an adaptive fog architecture, which dynamically selects different links to reduce transmission delay by measuring link delay. When the link delay is high, the performance is poor. Acharya in [17] proposed that for specific computer vision programs, services that may be needed should be loaded in advance during the offline phase of the program, so as to reduce the delay of service loading when the program is running. However, if the application is changed or too many applications are opened at the same time, the performance of this strategy will decline sharply. Xu et al. in [18] applied the technology of service caching. When the application is changed frequently, the algorithm has poor effect on the delay reduction. Shu et al. in [19] used the topology diagram and unique sequence diagram of tasks for hierarchical offloading and resource loading, but the fine-grained topological ordering of tasks needed to be

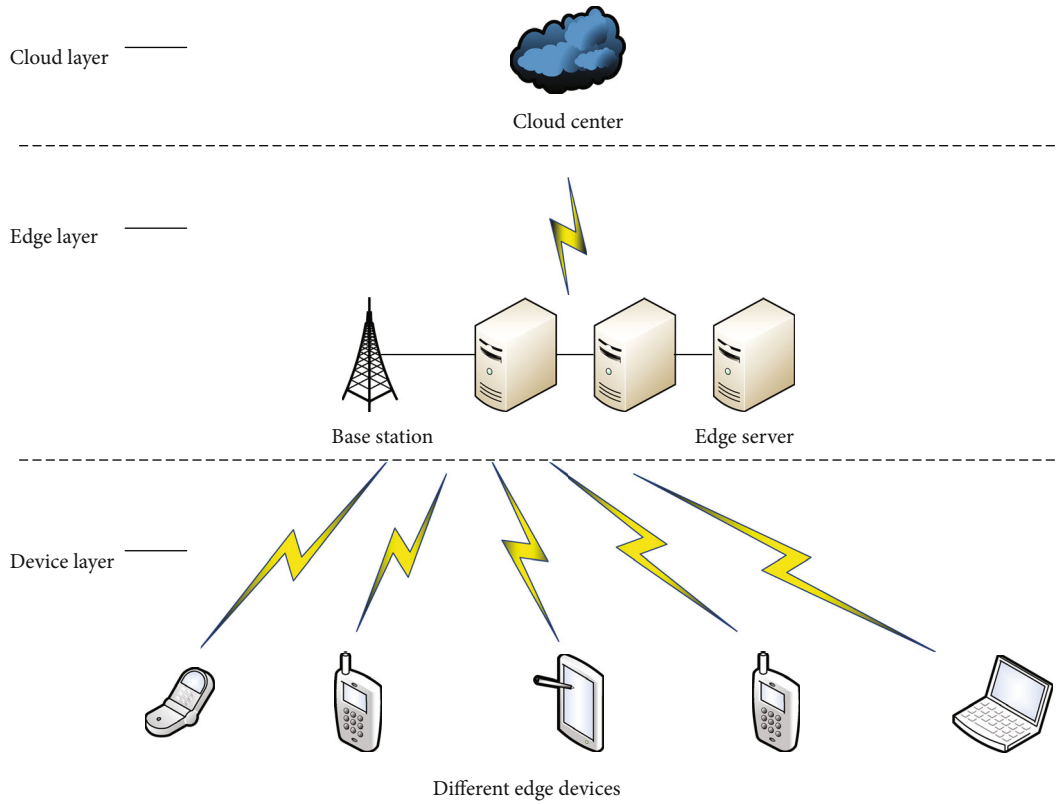


Figure 1: Edge computing workload architecture model.

determined in advance, which was not suitable for the changeable network environment.

In summary, the research boundary of edge computing or fog computing is fuzzy, but it is unified in the optimization of task assignment decision and resource coordination allocation in [20]. However, all existing research schemes have some limitations and ignore the importance of historical data. In this paper, an innovative concept based on application prediction is proposed, using historical data to predict the application to be used in the future in the edge computing network architecture. By loading the virtual service of the application on the edge server in advance, the service loading delay can be minimized, thus reducing the overall edge task response time.

3. System Model and Problem Formulation

This section briefly introduces the system, transmission delay, and computing delay models as well as problem formulation.

3.1. System Model. Figure 1 shows the three-layer edge computing workload allocation architecture adopted.

The device layer contains different edge devices, each of which commits the computing task at different times. J represents the set of edge devices (EDs). j represents different device. K represents the set of applications (APPs), all of which are in a known collection. k represents different types of application. T represents the network time slot. T_n represents the current time slot at time n. T_{n-1} represents the previous network time slot. T_{n+1} represents the next network slot. In each network slot, the tasks that device j submits to the edge layer can be represented by K_j.

In the edge layer, BS is used to represent the base station, and the User App LCM Proxy module is set up, which is responsible for receiving the task sequence from different mobile edge devices and uniformly assigning the tasks to different servers. The computing unit is different edge server hosts, with M representing the set of edge servers (ESs) and m representing different edge servers. In each edge server, B_m represents total bandwidth of edge server m, B_m^{idle} represents the remaining allocable bandwidth, C_m represents total computing resources, and C_m^{idle} represents the remaining allocable computing resources in CPU cycles per second.

In workload allocation scheme, transmission delay and computing delay are taken into account. Hence, the task requests of edge device are more likely to be allocated to the closer or lighter workload edge server. The meaning of the parameters used in this paper is listed in Table 1.

3.2. Transmission Delay Model. When the edge device task request is assigned to a different edge server, the edge device will establish a connection with the edge server. The data generated in the edge device will be transferred to the edge server. Therefore, transmission delay consists of two parts: link delay and data delay.

In general, link delay is related to channel state and power, which is not discussed in this paper. t_{jm}^{link} is used to represent transmission link delay between device j and edge

Table 1: The representative meaning of different symbols.

Symbol	Description
J	The set of EDs
K	The set of APPs
T	The different network time slots
M	The set of ESs
K_j	The task that edge device j commits to the edge layer
B_m	The total bandwidth of edge server m
B_m^{idle}	The remaining allocable bandwidth
B_{jm}	The bandwidth allocated on edge server m for the task of device j
C_m	The total computing resources of edge server m
C_m^{idle}	The remaining allocable computing resources
C_{jm}	The computing resources allocated on edge server m for the task of device
t_{jm}^{link}	The transmission link delay between device j and edge server m
t_{jm}^{tran}	The total transmission delay between device j and edge server
x_{jm}	In binary array, whether the task of device j is assigned to the edge server m
t_k^{load}	The APP k service load time on the edge server side
θ_{jk}	The calculation strength required for task k unit data of device
D_{jk}	The amount of data to be processed by task k of device j
τ_{jk}^{worse}	The maximum tolerance delay by task k of device j

server m, and the multiaccess edge coordinator in the edge computing system is responsible for update and maintenance.

In each network slot, the task that device j submits to the edge layer can be represented by K_j. In task K_j, the data size to be transmitted and processed is D_{jk}. After the task is allocated to edge server m and bandwidth B_{jm} is allocated, the data delay can be obtained as follows:

$$t_{jm}^{\text{data}} = \frac{D_{jk}}{B_{jm}}. \tag{1}$$

In the transmission of backjourney data, the return data is usually the processing result, and the backjourney data delay is not considered. Therefore, the backjourney transmission delay only considers the link delay.

The total transmission delay of a task can be expressed as follows:

$$t_{jm}^{\text{tran}} = 2 * t_{jm}^{\text{link}} + \frac{D_{jk}}{B_{jm}}. \tag{2}$$

3.3. Computing Delay Model. θ_{jk} represents the calculation strength required for task k unit data of device j in the current time slot. Therefore, when the task K_j is assigned to edge server m, the total calculation strength by the task processing data is expressed as follows:

$$C = \theta_{jk} * D_{jk}. \tag{3}$$

The computing delay required to process the task can be expressed as follows:

$$t_{jm}^{\text{comput}} = t_k^{\text{load}} + \frac{\theta_{jk} * D_{jk}}{C_{jm}}. \tag{4}$$

3.4. Problem Formulation. In this paper, the binary array x_{jm} represents the task assignment strategy. It can be expressed as follows:

$$\sum_{j \epsilon J} \sum_{m \in M} x_{jm} = \begin{cases} 1, & \text{if } K_j \text{ is assigned to edge server } m, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

In summary, the minimum cost of the system can be expressed as P1 in T_n.

$$\text{P1} \quad \min \quad \sum_{j \epsilon J} \sum_{m \in M} x_{jm} \left(t_{jm}^{\text{tran}} + t_{jm}^{\text{comput}} \right) \tag{6}$$

$$\text{s.t.} \quad \sum_{j \epsilon J} \sum_{m \in M} x_{jm} = 1, \tag{7}$$

$$0 \le \sum_{j \epsilon J} \sum_{m \in M} x_{jm}\left(B_{jm}\right) \le B_m^{\text{idle}}, \tag{8}$$

$$0 \le \sum_{j \epsilon J} \sum_{m \in M} x_{jm}\left(C_{jm}\right) \le C_m^{\text{idle}}, \tag{9}$$

$$2 * t_{jm}^{\text{link}} + \frac{D_{jk}}{B_{jm}} + t_k^{\text{load}} + \frac{\theta_{jk} * D_{jk}}{C_{jm}} \le \tau_{jk}^{\text{worse}}. \tag{10}$$

Here, constraint (7) ensures that the tasks on each device are indivisible and can only be assigned to one edge server. Constraint (8) indicates that the total bandwidth of the tasks assigned on each edge server is not greater than the remaining bandwidth of the current server. Constraint (9) indicates that the total calculation strength of the tasks assigned on each edge server is not greater than the remaining calculationstrength of the current server. Constraint (10) indicates that after all tasks are assigned, the total delay of each task is smaller than the maximum tolerance time.

P1 is a workload allocation scheme that combines task assignment with resource allocation. P1 has proved that it cannot be solved in polynomial time and is proved to be a NP hard problem [15].

4. Proposed Solution

Since P1 is a NP hard problem, in order to simplify the problem and obtain the workload allocation scheme in the fastest way, we propose AP algorithm, which creatively adds historical analysis into the consideration of the problem. In the current time slot, the application to be used in the next time slot is predicted, and the offline algorithm is adopted to preallocate edge resources. Second, the offline problem is decomposed into two subproblems: the task assignment subproblem and the resource allocation subproblem. Piecewise solving can reduce the difficulty of solving and get the suboptimal solution which is close to the optimal solution most quickly.

The edge network workload allocation algorithm based on application prediction proposed in this paper is mainly divided into two stages, as shown in Figure 2.

4.1. Application Prediction. In order to adapt to the prediction scheme in the edge computing scenario, we propose an AP algorithm based on LSTM. In the LSTM prediction method, "input gate, forgetting gate" and control parameter C_t are introduced in each neural unit.

The forgetting gate is shown as follows:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right). \tag{11}$$

σ is the sigmoid function.

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{12}$$

In the domain of $(-\infty, \infty)$, the value is [-1, 1]. W_f is the weight vector, h_{t-1} is the previous output of the iterative computing process, x_t is the current input sequence matrix, and b_f is the bias vector. f_t is used for the subsequent computing with the control parameter C_{t-1} to determine which kind of information should be discarded.

The input gate is shown as follows:

$$\begin{aligned} I_t &= \sigma(W_I[h_{t-1}, x_t] + b_I), \\ C_t' &= \tan h(W_c[h_{t-1}, x_t] + b_c). \end{aligned} \tag{13}$$

I_t represents new information to be retained, W_I is the weight vector of the input gate, b_I is the bias vector of the input gate. C_t' is the output state of the input gate, W_c is the weight vector of input gate output state, and b_c is the bias vector of input gate output state.

Update the new control vector according to

$$C_t = f_t * C_{t-1} + I_t * C_t'. \tag{14}$$

The result of the output gate is expressed as follows:

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\ h_t &= o_t * \tan h(C_t). \end{aligned} \tag{15}$$

o_t is the output gate, W_o is the weight vector of the output gate, and b_o is the bias vector of the output gate. h_t is the output of the output gate, used to calculate the next neuron.

The specific steps of the AP algorithm can be shown in Algorithm 1.

Algorithm 1 is summarized as follows:

(1) The edge network server has the network request usage record of access device. First, the network records are obtained as a reference for historical information

(2) The time information recorded in the network is used to process the data. The network records of each access device are sorted from time information data to time series data. Filter the data into a sequence with the following characteristics; each line includes time, device ID, application ID, and size of the request data volume

(3) The extracted time stamp information (including month, week, hour, minute) is encoded by one-hot as a time feature

(4) The historical data are classified according to the ratio of 8 : 2 to the training set and the test set. Meanwhile, Adam algorithm is used to optimize the parameters and the loss function is calculated by Cross Entropy Loss (CEL)

(5) The AP algorithm based on LSTM is used to train the application prediction model, and the network history information obtained in the previous step is used as the input training set to get the optimal training model

In this paper, we use the data set collected by the application layer of a stable edge system which is published in the

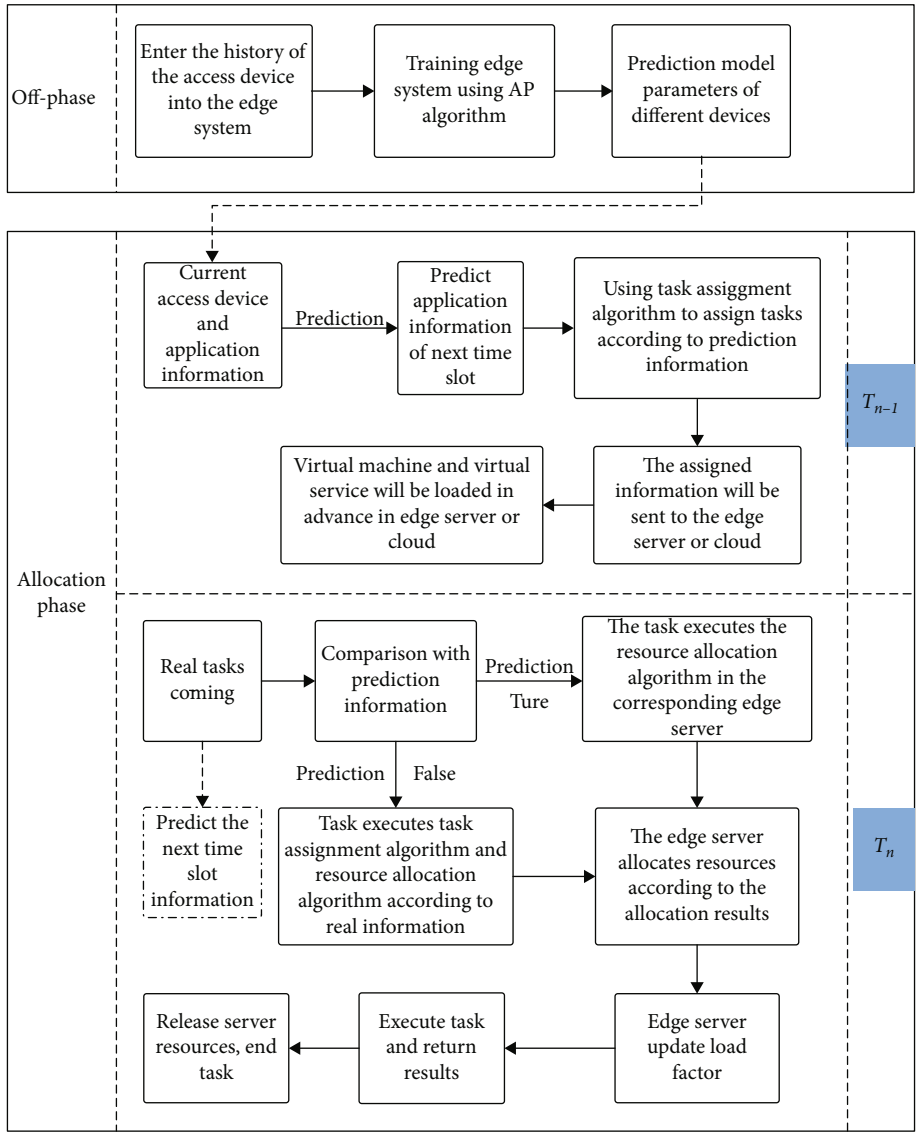


Figure 2: Algorithm execution flow diagram.

Input: the access device network request usage record as H.
Output: Model parameters set P_j of each device $j \in J$

1: Process data H into time series data H'
2: Separate H' by different device j as H'_j
3: **for** each $j \in J$**do**
4: Extract timestamp information from H'_j as feature;
5: Conduct application and different feature into one-hot code;
6: Divide the data H'_j into a training set and a test set according to 8: 2;
7: Train LSTM model;
7: Use Adam algorithm to optimize parameters P_j;
8: **end for**
9: **Return**P_j

Algorithm 1: The AP algorithm based on LSTM.

Kaggle website. The data set contains about 50 access devices and data usage in a base station system. The application usage of different equipment types in the data set is extracted, and different models are trained with the extracted data set. These models are tested, and the average accuracy is obtained. First, in order to compare the performance of different LSTM model complexities, we test the number of layers of different LSTM hidden layers, which are represented by LSTM 1, LSTM 2, LSTM 3, and LSTM 4, representing one layer, two layers, three layers, and four layers of LSTM hidden layers, respectively.

The final comparison results are shown in Table 2. With the increase of the number of LSTM hidden layers, the accuracy of the AP algorithm is also increasing, which is conducive to learning the nonlinear relationship of sample features. However, after increasing from three layers to four layers, the learning ability of model is gradually saturated, and the performance is no longer improved. If we continue to increase the hidden layer, the complexity will increase, which will lead to too many redundant operations. Therefore, under the conditions of this paper, after weighing the complexity and accuracy, the complexity of the LSTM model will adopt three hidden layers. Second, to verify the accuracy of the application prediction model, we compare the LSTM algorithm with the prediction model based on Markov and the prediction model based on RNN. F-Measure parameters are introduced for horizontal comparison. F-Measure parameters are weighted harmonic average of precision and recall, as shown in the formula, which are used to evaluate the merits and demerits of the classification model.

$$F\text{-Measure} = \left(1 + \alpha^2\right) \frac{\text{precision} * \text{recall}}{\alpha^2 * \text{precision} + \text{recall}}, \tag{16}$$

where α is usually equal to 1 and F-Measure is also regarded as F1. In F-Measure of multiclassification problem, the multiclassification problem is usually transformed into N binary classification problems. The F1 of each binary classification can be calculated. The average value of NF1 can obtain the macro F1 of the whole model, which can be used as the evaluation standard of multiple classification models. The higher the value of macro F1 is, the more accurate the model is and the better the performance is. The final comparison results are shown in Table 3, which prove that the prediction model based on LSTM is more accurate.

By using the application prediction model, P1 is divided into two subproblems, namely, task assignment subproblem and resource allocation subproblem.

4.2. Task Assignment Subproblem. In T_n, the arrived tasks form a task set. Each device uses the AP algorithm to predict the application that will be used in T_{n+1}, forming the prediction task set. In T_n, the prediction task set of T_{n+1} is preallocated. In the preallocated resources, the amount of data for each task is estimated based on the historical record, and the computing resources and bandwidth required to meet its delay constraints are given.

Therefore, the value of D_{jk}, B_{jm}, θ_{jk}, and C_{jm} can be determined in formula (6), so it can be converted into a constraint

Table 2: Comparison of accuracy of different model layers.

Model layers	Accuracy
LSTM 1	78.61%
LSTM 2	84.20%
LSTM 3	87.20%
LSTM 4	87.20%

Table 3: Comparison of performance of different models.

Model	Accuracy rate	Macro F1
Markov	65.53%	0.52
LF-Markov	76.35%	0.61
RNN	82.30%	0.67
LSTM	87.20%	0.74

condition. Owing to the preallocation, the service load time t_k^{load} is already loaded before the next time slot arrives and is not counted in the calculation. Then, the variables are only the link delay t_{jm}^{link}, which is between the device j and server m , and the task assignment strategy x_{jm}.

Given the prediction task set, the task assignment subproblem in T_{n+1} is determined by the link delay and task assignment strategy and thus can be obtained by solving the following problem:

$$\begin{aligned}
\text{P2} \quad \min \quad & \sum_{j\epsilon J} \sum_{m\in M} x_{jm}\left(2 * t_{jm}^{\text{link}}\right) \\
\text{s.t.} \quad & \sum_{j\epsilon J} \sum_{m\in M} x_{jm} = 1, \\
& 0 \le \sum_{j\epsilon J} \sum_{m\in M} x_{jm}\left(B_{jm}\right) \le B_m^{\text{idle}}, \\
& 0 \le \sum_{j\epsilon J} \sum_{m\in M} x_{jm}\left(C_{jm}\right) \le C_m^{\text{idle}}, \\
& 2 * t_{jm}^{\text{link}} \le \tau_{jk}^{\text{worse}} - \frac{D_{jk}}{B_{jm}} - \frac{\theta_{jk} * D_{jk}}{C_{jm}}.
\end{aligned} \tag{17}$$

In P2, all variables represent the predictive variables of the next time slot. P2 is converted to a packing problem. A descending best fit algorithm based on greedy thought is adopted to solve this problem.

Algorithm 2 shows that the tasks to be assigned are arranged in a descending order according to the resource consumption. Since there are two constraint variables, the tasks are arranged according to the amount of data. In order to find the edge server that can meet its needs, the one with the lowest link delay is selected to assign the tasks to the edge server.

Algorithm 2 is summarized as the following steps:

(1) The set Z of prediction application K_j for the next time slot and the corresponding prediction information tuple $(D_{jk}, B_{jm}, \theta_{jk}, C_{jm})$ is input

Input: Z, M, and the estimated information tuple $(D_{jk}, B_{jm}, \theta_{jk}, C_{jm})$ of K_j
Output: Task Allocation plan x_{jm}
1: **Initialization:** $\sum_{j\epsilon J}\sum_{m\epsilon M} x_{jm} = 0$
2: Given the predicted application set of next time slot Z, sort K_j by D_{jk} from largest to smallest as Z'
3: **for** $K_j \in Z'$ **do**
4: **for** $m \in M$ **do**
5: If B_{jm} and C_{jm} can be satisfied by m, add to temp set m'
6: **end for**
7: **for** $m \in m'$ **do**
8: Find the lowest t_{jm}^{link}, which represents transmission link delay between device j and server m, set $x_{jm} = 1$
9: **end for**
10: Clear m' and remove K_j from Z'
11: Update C_m^{idle}, B_m^{idle}, μ_m^B and μ_m^C of server m
12: **end for**
13: If $Z' \neq \varnothing$ send Z' and Information tuple to cloud center
14: **Return** x_{jm}

Algorithm 2: Task assignment.

(2) The tasks to be assigned are in a descending order according to the amount of data D_{jk}

(3) By finding all the edge servers whose remaining bandwidth and computing resources satisfy B_{jm} and C_{jm}, the tasks in a descending order are unloaded to the server with the lowest link delay, and then, the remaining bandwidth and computing resources of the server, as well as the load factors μ_m^B and μ_m^C, are updated. After all tasks are assigned, if the task set is not empty, it proves that the current task set has exceeded the edge server's load capacity, and the remaining tasks are assigned to the cloud for execution

After Algorithm 2 is executed, the task assignment scheme of all tasks will be output.

4.3. Resource Allocation Subproblem. After determining the task assignment scheme, x_{jm} is determined. The constraint formula can be expressed as P3 in each edge server m.

$$\text{P3} \quad \min \quad \sum_{j\epsilon J}\sum_{m\epsilon M} x_{jm}\left(2 * t_{jm}^{\text{link}} + \frac{D_{jk}}{B_{jm}} + \frac{\theta_{jk} * D_{jk}}{C_{jm}}\right) \tag{18}$$

$$\text{s.t.} \quad 0 \le \sum_{j\epsilon J}\sum_{m\epsilon M} x_{jm}\left(B_{jm}\right) \le B_m^{\text{idle}}, \tag{19}$$

$$0 \le \sum_{j\epsilon J}\sum_{m\epsilon M} x_{jm}\left(C_{jm}\right) \le C_m^{\text{idle}}, \tag{20}$$

$$0 \le 2 * t_{jm}^{\text{link}} + \frac{D_{jk}}{B_{jm}} + \frac{\theta_{jk} * D_{jk}}{C_{jm}} \le \tau_{jk}^{\text{worse}}. \tag{21}$$

In solving this problem, there are two optimization variables, B_{jm} and C_{jm}. Based on the preallocated computing and bandwidth resources, the ratio of bandwidth resources and computing resources of edge server m is calculated μ_m^B and μ_m^C, also known as load factor. In general, the load time of virtual service increases with the increase of load factor.

$$\begin{aligned} \mu_m^B &= \sum_{j\in J} \frac{x_{jm}\left(B_{jm}\right)}{B_m}, \\ \mu_m^C &= \sum_{j\in J} \frac{x_{jm}\left(C_{jm}\right)}{C_m}. \end{aligned} \tag{22}$$

Compared with μ_m^B and μ_m^C, the parameter with high occupancy rate is determined as the main constraint of delay. For example, when $\mu_m^B = 0.2$ and $\mu_m^C = 0.6$, it can be determined that the current computing resource is the main constraint of delay. Then, the parameter of preallocated computing resource is taken into P3, and C_{jm} is used as the known condition to optimize the allocation of B_{jm}.

In general, the update process of the algorithm needs to avoid server overload and the prediction error leading to resource reallocation. Therefore, according to prior experience, $\mu_m^C \le 0.7$ and $\mu_m^B \le 0.8$.

When a parameter is fixed, the second partial derivative of formula (18) is greater than 0, the Hessian matrix is a positive definite matrix. Moreover, the constraints of P3 is transformed into linear constraints; P3 is a convex programming problem.

As P3 is a convex problem, we can derive the optimal solution of P3 by solving the KKT condition [21]. Algorithm 3 can be expressed as the following steps.

(1) By traversing each server, μ_m^B and μ_m^C of the current server are calculated according to the predicted bandwidth and computing resources

Input: x_{jm}, the predicted bandwidth B'_{jm}, and predicted computing resources C'_{jm}
Output: B_{jm}, C_{jm}
1: **for**m $\in M$**do**
2: Add all B'_{jm} to calculate μ^B_m;
3: Add all C'_{jm} to calculate μ^C_m;
4: **if**$\mu^B_m \geq \mu^C_m$**then**
5: Let $B_{jm} \leftarrow B'_{jm}$;
6: Set formula (18) Lagrange function on C_{jm}, get $L(C_{jm}, \lambda, \mu)$;
7: Calculate KKT condition, get the result set C_{jm};
8: **else**
9: Let $C_{jm} \leftarrow C'_{jm}$;
10: Set formula (18) Lagrange function on B_{jm}, get $L(B_{jm}, \lambda, \mu)$;
11: Calculate KKT condition, get the result set B_{jm};
11: **end if**
12: **end for**
13: **Return**B_{jm}, C_{jm}.

Algorithm 3: Resource allocation.

(2) The constraints of the current server task are determined by comparing μ^B_m and μ^C_m

(3) According to the comparison results, the fixed parameters are determined and the corresponding Lagrange functions are obtained

(4) It calculates the KKT condition and gets the result

5. Simulation

5.1. Simulation Setting. In this section, we set up simulations of the proposed scheme to evaluate its performance, which uses iFogSim simulation platform [22]. In parameter setting, the simulation environment consists of 5 edge servers, where the assignable bandwidth of each edge server is 1000 Mbps and the CPU frequency of each edge server is 1 Ghz. Meanwhile, the length of each time slot is set as 1 s. According to prior experience, $\mu^C_m \leq 0.7$ and $\mu^B_m \leq 0.8$. Moreover, the link delay between different devices and edge servers is randomly chosen according to the normal distribution. The link delay connected to different devices is different. The size of the link delay is set within 20 ms. In addition, the loading time of virtual service among edge server is only related to the application types, and the loading time of virtual service for different applications is set between 20 ms and 50 ms. The number of connected devices of the edge system is set between 0 and 50.

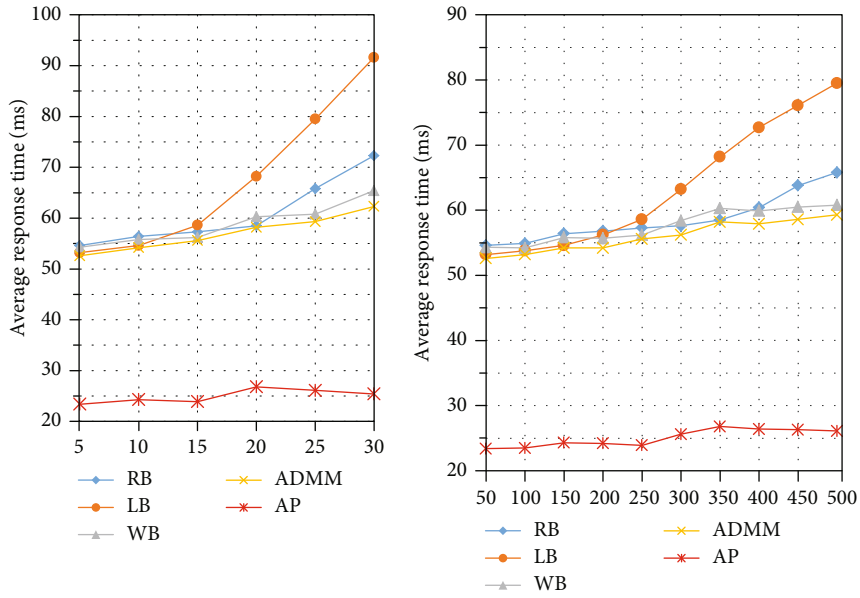
We select four other workload allocation strategies for comparison: (1) the random-based strategy (RB), (2) the latency-based strategy (LB), (3) the weighted-based strategy (WB) [11], and (4) the ADMM algorithm based on cooperative task offloading [23]. The basic idea of RB is that tasks will be randomly assigned to different computing servers after they arrive, and a competition scheme of preemptive resources is adopted. LB is used to select the link with the lowest delay according to different link delays when a task arrives and then offload the task to the edge server. WB is used to set different weights for different tasks. The weight is set according to the delay sensitivity of tasks. Through the optimization iteration of weights, the workload allocation scheme is optimized. The ADMM algorithm based on cooperative task offloading is a series of reconstructions through the reconstruction linearization technique, and a parallel optimization framework is proposed by using ADMM and difference of convex function (DC) programming to solve the problem.

5.2. Comparison of Average Response Time. We discuss the impact of the number of access devices and network traffic on the average response time of tasks (also known as the average wait time of tasks). Task response time is defined as the time between the submission of a task request to the edge server, the edge server setting up the virtual machine, and the notification of the device to start offloading task data. Fast response times are critical to improving user QoE.

Figure 3(a) shows the average response time for different numbers of access devices, in which AP achieves lower response time as compared to the other four algorithms. In Figure 3(b), the average response time for different network traffic in AP is significantly smaller than that of RB, LB, WB, and ADMM.

As shown in Figure 3, when the number of tasks increases, the average response time of LB grows the fastest. When the link delay of a server to most devices is the lowest, it is easy for LB to cause users to gather in a specific server. Therefore, when the access devices are increased, LB has a large probability to overload a single server, which affects the virtual machine establishment process, resulting in a large increase in the average response time. The average response time increases from 53.2 ms for the first 5 access devices to 91.6 ms for the 30 access devices.

RB can largely avoid server overload. However, the task on each device cannot be assigned to the optimal server node, which leads to the increase of link delay. In both experiments,



(a) Total number of access equipment

(b) Total network traffic per slot (M)

Figure 3: Comparison of average response time of different algorithms.

it can be seen that when the number of devices increases, the growth trend is slightly smaller than that of LB.

WB is a workload allocation strategy based on different weights, so its performance is better than LB and RB. When the number of access devices initially increases, the response time does not change significantly. When the access devices reach 30, the average response time only increases from 54.3 ms to 65.4 ms.

The task assignment of the ADMM algorithm is close to the theoretical optimal performance, so it can allocate tasks optimally. When the number of access devices increases, the response time change is not obvious, which can have better performance than WB. When the access devices reach 30, the average response time increases from 52.6 ms to 62.3 ms.

AP algorithm is proposed in this paper, and it can be seen that the average response time is greatly reduced. Since the average response time mainly depends on link delay and server load time, the AP algorithm in this paper has loaded the virtual machine of the application on the server in advance, so the server load time can be saved. Theoretically, the average response time of the AP algorithm is only affected by link delay and prediction accuracy. It can be seen that when the access devices reach 30, the response time is only 25.4 ms, which can still reduce the response delay by about 60% compared with the current best performance ADMM algorithm.

5.3. Comparison of Average Completion Time. The average completion time is defined as the time interval from the task request submitted to the edge server until the server returns the final result. It is a direct index to measure the performance of workload allocation scheme.

5.3.1. The Impact of Different Access Devices. In parameter setting, in order to better compare the performance of different strategies, we set the task data size within 20 M per time slot without extreme data. The calculation strength θ_{jk} required for each task unit data is generated between $1 * 10^7$ CPU cycle/M and $2 * 10^7$ CPU cycles/M. The impact of energy consumption on transmission delay is ignored.

Comparing average completion time with different numbers of access devices under different algorithms in Figure 4, it can be seen that the completion time increases with the increase of the number of access devices. When 5 devices are connected, each server node is in no load state, and the computing and bandwidth resources are sufficient, so the transmission delay is the main part of the total completion time of the task. The performance of LB, WB, and ADMM algorithms is relatively consistent, and the average completion time is between 560 ms and 580 ms. The average completion time of the RB algorithm is higher than that of other algorithms due to the high link delay of some tasks, which is 594.6 ms.

With the increase of access devices, the LB algorithm may cause some server overload, resulting in the average completion time higher than other algorithms. Therefore, the RB algorithm performs better than the LB algorithm in small and medium task load.

The overall performance of the WB algorithm is better than the RB algorithm and LB algorithm. When the number of tasks increases from small to large scale, the average completion time increases within a reasonable range, close to linear growth. Since the RB algorithm and LB algorithm are close to exponential growth, user QoE drops sharply. The RB algorithm has the worst performance. When the access device reaches 30, the average task completion time of the RB algorithm has increased to 952.3 ms.

The ADMM algorithm is close to the theoretical optimal value. When the number of tasks is small, the average task completion time almost does not increase. When the number

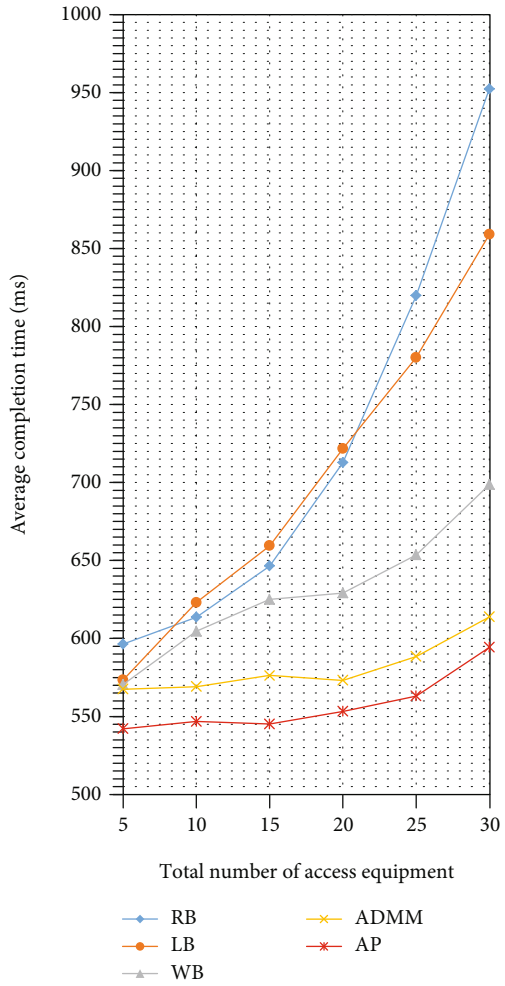


Figure 4: Comparison of average completion time of different access devices.

of tasks increases to a large scale, it only shows a small increase. Meanwhile, owing to ignoring service load delay by application prediction, the AP algorithm is better than the ADMM algorithm.

5.3.2. The Impact of Different Server Processing Capabilities. In order to further compare and explore the performance of different algorithms, we fix 15 access devices in the experiment, which are in the state of medium load. By changing the processing capacity of each server, the average completion time curve is obtained as shown in Figure 5. It can be seen that the average completion time curve decreases significantly between 0.6 and 1.0 Ghz in processing capacity. After 1.0 Ghz, with the increase of processing capacity, the average completion time decreases less. Thus, the server can still be under normal load when the processing capacity is around 1.0 Ghz. When the processing capacity is reduced, each server is gradually full or even overloaded, and the average completion time increases sharply. When the load capacity is gradually increased, the impact on the average completion time is not significant.

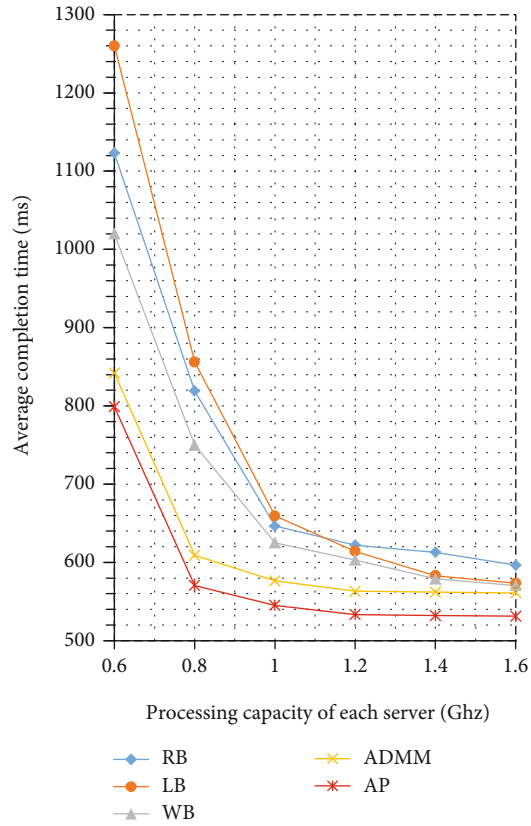


Figure 5: Comparison of average completion time of servers under different processing capabilities.

In the comparison of the average completion curves of the five algorithms, it can be seen that the LB algorithm has the worst performance when the server processing capacity is insufficient. The RB algorithm has the worst performance when the server performance is sufficient. The WB algorithm is slightly better than the LB algorithm and RB algorithm. However, the AP algorithm proposed in this paper has the optimal performance under different server processing capacities. The average completion time is lower than that of the ADMM algorithm between 20 ms and 50 ms, which is close to the theoretical performance.

5.3.3. The Impact of Different Network Traffic. Figure 6 shows the average completion time curve of different algorithms with the change of network traffic in a day.

In the experiment, we select one day's data from the data set and input them into the simulation platform. The abscissa unit is hour, 1 represents the data between 0 and 1. A total of 24 hours of data is counted. The network traffic represents the peak of network traffic within an hour. The ordinate represents the average completion time when network peak occurs.

It can be seen that the RB algorithm and LB algorithm are greatly affected by the network traffic. When the network traffic increases, the average completion time will also increase significantly. The performance of the WB algorithm is better than the RB and LB algorithm, but worse than the

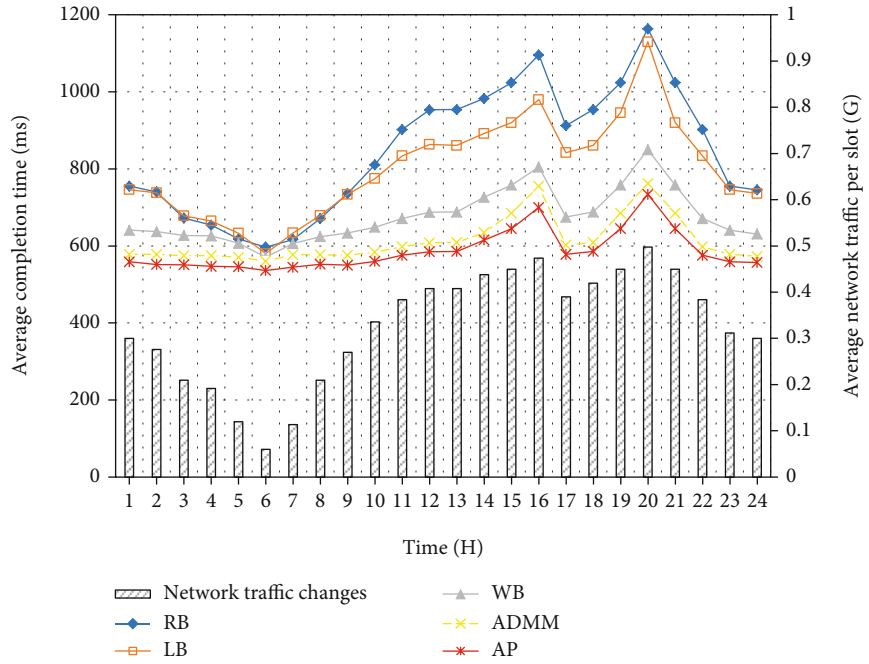


Figure 6: Comparison of average completion time under network traffic changes.

ADMM algorithm. When the network traffic is below 0.55 G, with the change of network traffic, the average task completion time does not change much, ensuring a certain stability. The AP algorithm proposed in this paper, due to the preloading of the virtual machine, reduces the time of virtual machine establishment process for most tasks and shows the optimal performance.

The experiment results confirm the feasibility of the AP algorithm. Meanwhile, when the traffic increases to a large load, the growth of the average completion time is the same as that of the ADMM algorithm, which proves that the AP algorithm in this paper is also close to the theoretical optimal result in workload allocation.

6. Conclusions

In this paper, we propose the edge workload allocation scheme using the AP algorithm. By using the AP algorithm, the workload allocation problem is divided into two subproblems, which are the task assignment subproblem and the resource allocation subproblem. The task assignment subproblem can be transformed into a packing problem and solved by the descending best fit algorithm. Based on the task assignment scheme, the resource allocation subproblem can be transformed into a convex programming problem by changing the constraint conditions and solved by KKT condition. The simulation results show that compared with the current best performance ADMM algorithm, the AP algorithm proposed in this paper can reduce the response delay by about 60%. In terms of average completion time, compared with the current best performance ADMM algorithm, the performance is still improved by 5%-9%, which has high practical significance.

Data Availability

The simulation data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The work was supported by "National Natural Science Foundation of China" with No. 61902052, "National Key Research and Development Plan" with No. 2017YFC0821003-2, "Dalian Science and Technology Innovation Fund" with Nos. 2019J11CY004 and 2020JJ26GX037, and "the Fundamental Research Funds for the Central Universities" with Nos. DUT19RC(3)003 and DUT20ZD210.

References

[1] D. Reinsel, J. Gantz, and J. Rydning, "Data age 2025: the digitization of the world from edge to core," pp. 1–29, 2018, IDC White Paper Doc#US44413318.

[2] W. Li, Z. Chen, X. Gao, W. Liu, and J. Wang, "Multimodel framework for indoor localization under mobile edge computing environment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4844–4853, 2019.

[3] A. Yousefpour, C. Fung, T. Nguyen et al., "All one needs to know about fog computing and related edge computing paradigms: a complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[4] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: a survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[5] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[6] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[7] X. Lyu, W. Ni, H. Tian et al., "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[8] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2868–2881, 2018.

[9] J. du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[10] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[11] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, Atlanta, GA, USA, 2017.

[12] Y. Sun, S. Zhou, and J. Xu, "EMM: energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[13] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017.

[14] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, Atlanta, GA, USA, 2017.

[15] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[16] T. Dlamini and A. F. Gambin, "Adaptive resource management for a virtualized computing platform within edge computing," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Boston, MA, USA, 2019.

[17] A. Acharya, *Edge-Assisted Workload-Aware Image Processing System*, Boise State University Theses and Dissertations, 2019.

[18] J. Xu, L. Chen, and Z. Pan, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, Honolulu, HI, USA, 2018.

[19] C. Shu, Z. Zhao, Y. Han, and G. Min, "Dependency-aware and latency-optimal computation offloading for multi-user edge computing networks," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Boston, MA, USA, 2019.

[20] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, thirdquarter.

[21] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge university press, 2013.

[22] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[23] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: an ADMM framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.