


Research Article

Offloading in Mobile Edge Computing Based on Federated Reinforcement Learning

Yu Dai,¹ Qing Xue,¹ Zhen Gao,² Qihong Zhang,¹ and Lei Yang ²

¹College of Software, Northeastern University, Shenyang, China

²College of Computer Science and Engineering, Northeastern University, Shenyang, China

Correspondence should be addressed to Lei Yang; yanglei@mail.neu.edu.cn

Received 5 July 2021; Revised 27 December 2021; Accepted 20 January 2022; Published 8 February 2022

Academic Editor: Huaming Wu

Copyright © 2022 Yu Dai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Mobile edge computing (MEC) has become a more and more popular technology, which plays a very important role in various fields. In view of the task of offloading of multiple users, most of the existing studies do not take into account data sharing and cooperation among users, which can easily lead to less generalization of the model trained by a single user, and some data sharing may also cause privacy leakage. Then, this paper uses the method of federated reinforcement learning to solve this problem in order to insure privacy. Besides, considering the poor quality of local models, which leads to the poor versatility of the overall parameters, this paper proposes a federated reinforcement learning method based on Attention mechanism to aggregate the parameter weights, which makes the new model more generalized. The experimental results show that the federated reinforcement learning task offloading model with Attention mechanism can reduce the processing delay of the task.

1. Introduction

In recent decades, the development of the Internet and wireless communication technology provides a very convenient channel for exchanging information in people's daily life. Internet of things has been used in smart homes, intelligent transportation, human health detection, disaster management, and many other fields. In order to solve the problem of big data computing, some researchers have introduced mobile cloud computing (MCC) [1] service based on the centralized cloud data center to cope with the rapid increase in a number of mobile applications and the long-term limitations of battery technology. However, the offloading of MCC only reduces the energy consumption burden of the mobile device, but at the same time, because the data transmission is very complex, the delay for the mobile device to obtain the processing result data is very large. In 2014, the European Telecommunication Standards Association (ETSI) explained the basic concepts and various aspects of MEC (Mobile Edge Computing) [2], placing the MEC server near the place where the data was generated, enabling the server to process user requests at the edge of the

network, reducing transmission and location-aware delays, reducing network load, and improving the user experience.

Offloading computing tasks to edge servers is usually affected by many factors, such as software and hardware environment, network bandwidth, wireless channel interference, connection quality of communication links, overall performance of mobile devices, and so on. Therefore, offloading computing tasks in MEC is a key research issue, that is, whether computing tasks are executed locally or offloaded from edge nodes. In addition, many mobile applications in real scenarios (e.g., face recognition, gesture recognition, and augmented reality [3]) are composed of dependent tasks; that is, there are storage dependencies and computational dependencies between tasks. Therefore, it is unrealistic to offload some computing tasks to the edge server and ignore the relationship between them. In addition, the existing models based on heuristic or reinforcement learning offloading algorithms lack generalization due to the lack of data sharing among devices, and each device only trains the model locally. At the same time, some methods require the device to send the user's personal data to the central server, which may lead to serious user privacy and data security problems.

In the case of dealing with the offloading problem of dependent tasks, the computational offloading method proposed in this paper uses federated reinforcement learning (FRL) to solve the problem of weak generalization and privacy leakage caused by the lack of data transmission between users. At the same time, this paper considers that the trained models are different due to the different computing power and data quality of each mobile device, and it is unreasonable to simply aggregate each device model through average, so this paper considers the weighted method (i.e., weighted federation aggregation) and proposes a federation reinforcement learning task offloading model which integrates Attention [4] mechanism.

The rest of this paper is organized as follows. Section 2 introduces the related work of task computing offloading. Section 3 gives the task model and computing model of computing task offloading. Section 4 details the computational task offloading network model combined with reinforcement learning and proposes a federated reinforcement learning computing task offloading algorithm with Attention mechanism. Section 5 verifies the effectiveness of the proposed method through simulation analysis. Section 6 summarizes the full text.

2. Related Work

Recently, researchers have done a lot of research on solving the problem of task offloading, which can be roughly divided into the following two types of algorithms: one is the use of traditional algorithms such as heuristic or approximation. In the previous research, some research works considered whether the task of the mobile device was offloaded to the edge node, while others considered which edge computing node to offload the computing task to. Also, some works apply Reinforcement Learning (RL) method to make offloading decision. RL is an effective method for computing offloading decision. Considering the reward feedback from the environment in the future state, the RL agents can adjust their strategies to achieve the best long-term goals, which solves the problem of low efficiency of traditional methods [5]. Therefore, RL is a very important solution in the dynamic MEC system.

2.1. Traditional Algorithms Such as Heuristics. The mobile cloud computing system with multiple users and multiple tasks is considered in [6]. On the premise of minimizing the total cost, an effective approximate solution MUMTO uses a separable semistereotyped relaxation method to solve the binary offloading decision. In [7], the total network revenue is considered as the optimization objective, and the offloading problem is transformed into a convex problem, and an optimization algorithm based on multiplier alternating direction method is proposed to solve the problem of task offloading and content caching. From the point of view of prolonging the life of mobile devices, a dynamic offloading algorithm based on Lyapunov optimization is proposed in [8], which achieves the goal of energy saving on the premise of meeting the execution time requirements of a given

application. The above calculation offloading method does not take into account that the offloaded tasks may include computational dependencies, which may result in the extra communication cost between the tasks, while the algorithm proposed in this paper takes into account the offloading of dependent tasks and modeling.

A new algorithm is proposed in [9]. Starting from a minimum-delay scheduling scheme, tasks are migrated between the local core and the cloud, and dynamic voltage and frequency scaling techniques are applied to achieve energy saving. Aiming at the problem of task migration, a linear time rescheduling algorithm is proposed. In [10], the authors model the application as a directed acyclic graph (DAG). The UE side of the user equipment decides which subtask should be offloaded to which compute node (AP). In their work, the problem is described as a scheduling problem. In order to solve the NP-hard problem, the author proposes a heuristic algorithm based on the list scheduling algorithm. The algorithm takes into account of the transmission time between AP and the offloading time from UE to AP. Reference [11] proposes a minimum cost offload partition (MCOP) algorithm, which aims to find the optimal partition scheme under different cost models and mobile environments (i.e., to determine which parts of the application must run on mobile devices and which parts run on cloud or edge servers). In [12], the authors propose a heuristic mobility-aware offloading algorithm (HMAOA) to obtain the approximate optimal offloading scheme. The original global optimization problem is converted into multiple local optimization problems. Each local optimization problem is then decomposed into two subproblems.

References [9–12] are all based on heuristic or approximate algorithms to solve the problem of task offloading in DAG diagrams. When the external environment of the MEC system changes, the model may not be suitable for the changing environment, so it is necessary to update the mathematical model accordingly. Therefore, these algorithms are difficult to fully adapt to the dynamic changing MEC environment. The algorithm proposed in this paper effectively solves the problem that it is difficult to adapt to the changing environment based on heuristic or approximate algorithms by using reinforcement learning.

2.2. Offloading Algorithm Based on Reinforcement Learning. In recent years, deep reinforcement learning has been widely concerned by academia. In Reinforcement Learning [13], the agent does not need to know part of the information of the network in advance and achieves the optimization goal of maximizing the long-term benefits of energy consumption and delay by adjusting strategies. Researchers have recently begun to use deep reinforcement learning algorithms to solve this problem [14].

In [15], an optimization framework based on deep Q network (DQN) is proposed to solve the problem of joint task offload and resource allocation in wireless MEC when minimizing energy consumption and delay. Reference [16] considers that in a super-dense network, each task can select and offload multiple base stations, and the DQN algorithm is

used to obtain the optimal offloading strategy. Considering the load of edge nodes in [17], and the arrival deadlines of some tasks may not be processed and discarded, the authors propose a model-free distributed reinforcement learning algorithm. The model trained by the offloading algorithm based on reinforcement learning lack generalization due to the lack of data exchange between devices, and each device trains the model locally.

In addition, some methods require users to send data from this device to a central server, which may cause serious user privacy disclosure and sensitive data security problems, even the transmission of anonymous data will still put user privacy at risk, because attackers can recover by comparing anonymous information with other data [18]. Federated Learning is a concept proposed by Google in 2016 to solve privacy issues on mobile devices. Federated Reinforcement Learning (FRL) [19] is based on the sharing of experiences among agents so that more samples can be collected in a short period of time. On the other hand, the global model of federated reinforcement learning training is to aggregate the user model with weighted average, which leads to the problem of rigid average contribution of user model to global training parameters. In this paper, an FRL task offloading model with Attention mechanism is proposed from the point of view of protecting users' privacy.

3. System Model

In this paper, we consider an environment consisting of a set of local mobile devices $C = \{1, 2, \dots, M\}$, a base station (BS), and an MEC server connected to the BS, as shown in Figure 1. At the same time, considering that an application can be decomposed into multiple interdependent tasks in real life, this paper considers multidependent tasks where the optimization goal is to achieve the least time to complete all tasks in a task-dependent graph. For each mobile device, there is a task graph G containing dependent tasks that needs to be processed. When the task arrives randomly, the task graph is calculated and processed in the fashion of First Input First Output (FIFO). In this paper, a task graph is represented as $G(T, E)$, where T represents the set of task vertices, E represents a set of edges, and $e = (t_i, t_j)$ represents a directed edge from t_i to t_j , indicating that there is data transferring from t_i to t_j , where t_i is the parent node of t_j , which can only be executed after the completion of t_i execution, emphasizing that the order in which the tasks are executed cannot be changed. The optimization goal of computing the offloading problem is to make an offloading decision for each task in the DAG diagram, that is, whether the task is executed locally or whether the task as a whole is offloaded to the MEC through channel transmission. For a DAG diagram of any user, all task offloading decisions are represented as $A_{1:n} = \{a_1, a_2, \dots, a_n\}$, where a_i represents the offloading decision of the user's t_i task, which is executed locally on the device when $a_i = 0$ and offloaded to MEC when $a_i = 1$. Tasks are processed sequentially on order of FIFO and can be processed only when the task arrives.

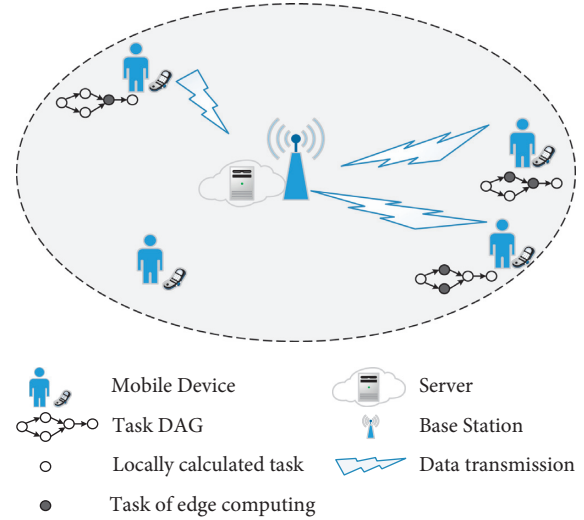


FIGURE 1: System model.

The local execution time T_i^L and the MEC execution time T_i^C can be calculated by the following formulae individually:

$$T_i^L = \frac{\text{data}_i}{f^L}, \quad (1)$$

$$T_i^C = \frac{\text{data}_i}{f}, \quad (2)$$

where data_i is the amount of data in task t_i , f^L is the local CPU cycles of the device, and f is the CPU cycles of the MEC. If a task needs to be offloaded to the edge node, it will result in uplink transmission delay T_i^{ul} as formula (3). If a task needs to transmit results locally from the edge node, it will generate a delay T_i^{dl} in the downlink as shown in formula (4):

$$T_i^{ul} = \frac{\text{data}_i^s}{R^{ul}}, \quad (3)$$

$$T_i^{dl} = \frac{\text{data}_i^r}{R^{dl}}, \quad (4)$$

where R^{ul} is the uplink transfer rate between the mobile device and the MEC and R^{dl} is the downlink transfer rate between the MEC and the mobile device. data_i^s is the amount of data that a task sends from the device to the MEC, and data_i^r is the amount of data of the task results downloaded from the edge server for the device.

In the process of local computing, computing at edge nodes, and uplink and downlink transmission, there may be some cases in which resources are occupied, so the start execution time of a task is not only affected by the task dependency graph, but also affected by the longer waiting time caused by the occupation of resources. The local start execution time of the task t_i is RT_i^L , and the start execution time of the task t_i at MEC is RT_i^C , as shown in formulae (5) and (6). The computation of start times for uploading and downloading task data RT_i^{ul} and RT_i^{dl} is shown in formulae (7) and (8):

$$RT_i^L = \max \left\{ F_i^L, \max_{k \in \text{pred}(t_i)} \{ FT_k^L, FT_k^{dl} \} \right\}, \quad (5)$$

$$RT_i^c = \max \left\{ F_i^c, \max \left\{ FT_i^{ul}, \max_{k \in \text{pred}(t_i)} FT_k^c \right\} \right\}, \quad (6)$$

$$RT_i^{ul} = \max \left\{ F_i^{ul}, \max_{k \in \text{pred}(t_i)} \{ FT_k^L, FT_k^{dl} \} \right\}, \quad (7)$$

$$RT_i^{dl} = \max \{ F_i^{dl}, FT_i^c \}, \quad (8)$$

where $F_i^L = \max \{ F_{i-1}^L, FT_{i-1}^L \}$ is the start time of locally available resources, $F_i^c = \max \{ F_{i-1}^c, FT_{i-1}^c \}$ is the start time of the resources available on the MEC, $F_i^{ul} = \max \{ F_{i-1}^{ul}, FT_{i-1}^{ul} \}$ is the start time of the available resources in the uplink of the channel, and $F_i^{dl} = \max \{ F_{i-1}^{dl}, FT_{i-1}^{dl} \}$ is the start time of the available resources in the downlink of the channel. The local execution end time FT_i^L , the end time of MEC FT_i^c , the end time of uplink transmission FT_i^{ul} , and the end time of downlink transmission FT_i^{dl} are expressed as shown in the following formulae:

$$FT_i^L = RT_i^L + T_i^L, \quad (9)$$

$$FT_i^c = RT_i^c + T_i^c, \quad (10)$$

$$FT_i^{ul} = RT_i^{ul} + T_i^{ul}, \quad (11)$$

$$FT_i^{dl} = RT_i^{dl} + T_i^{dl}. \quad (12)$$

In this paper, we define the optimization goal as minimizing the execution delay $T_{A_{1:n}}$. Because it calculates the delay of the tasks in the DAG graph, it needs to be executed by all nodes, that is, to minimize the execution time of the end node, as shown in the following formula:

$$T_{A_{1:n}} = \max_{k \in K} (FT_k^L, FT_k^{dl}), \quad (13)$$

where K is the node with a degree of 0 in the task graph (i.e., the exited task), which stipulates that the start and end of the task graph need to be executed locally.

4. Network Model

4.1. Reinforcement Learning Offloading Model. Because the deep reinforcement learning algorithm is learned from the interaction with the environment and does not need a priori knowledge of the system, it has been widely used in communication and network [20].

4.1.1. Task Modeling MDP Process. In this paper, each task is modeled as a Markov decision process (MDP). The MDP process is formally expressed as $P = (S, A, R)$, S represents the state of the agent during the interaction between the agent and the environment, A represents the action selected by the agent in a certain state of the environment, R

represents the reward obtained by the agent after performing the action, and the state, action, and reward work together to form the environment of agent interaction. The definitions of state, action, and reward are as follows:

- (1) The offloading of each task is related to the amount of data of each task in the DAG task graph where the task node is located, the required CPU cycles, and the shape of the DAG graph. The offloading of task t_i is related to the offloading of task t_1 to task t_i in DAG because the offloading of task t_1 to task t_i may affect the start execution time of task t_i . In this paper, the state is defined as follows:

$$\text{state} : \{ \text{DATA}, T_{CPU}, G(T, E), A_{1:i} \}, \quad (14)$$

where DATA is the amount of data of each task, T_{CPU} is the number of CPU cycles required for each task to process, and $G(T, E)$ is the embedding of the graph (for each task, including the index of the current task, the index of its direct parent task, and the index of direct child tasks), and $A_{1:i}$ is the offloading decision from task t_1 to task t_i .

- (2) There are two kinds of action choices for each task: one is to execute locally; the other is to offload it to the MEC to execute. In this paper, the action is defined as follows:

$$\text{action} : \{0, 1\}. \quad (15)$$

- (3) The purpose of this paper is to minimize the total task execution delay $T_{A_{1:n}}$. Define the reward as shown in formula (16). Because the optimization goal of this paper is to minimize the total task execution delay, the total execution delay is composed of the processing delay of each task, and the agent is given a reward after each task interacts with the environment, so reward is defined as the processing time of each task. The smaller the delay is, the larger the reward value is, and vice versa.

$$\text{reward} : \{ T_{A_{1+i}} - T_{A_{1i}} \}. \quad (16)$$

4.1.2. seq2seq Structure. Because of the interdependent relationship between tasks, this paper uses the seq2seq [21] structure as the component model of reinforcement learning. The seq2seq model includes the Encoder and the Decoder, where both the Encoder and Decoder are composed of the LSTM model, as shown in Figure 2.

The input of Encoder is $\{ \text{DATA}, T_{CPU}, G(T, E) \}$ in state (i.e., the embedded vector $[t_1, t_2, \dots, t_n]$ of a task graph, where n represents the number of tasks contained in each graph). Decoder outputs the offloading decision of a task graph, and the action is expressed as $[a_1, a_2, \dots, a_n]$. In order to convert the DAG task graph into embedded vectors and input into the model, the tasks are sorted and indexed according to the ascending order of the critical paths, and the formula is as follows:

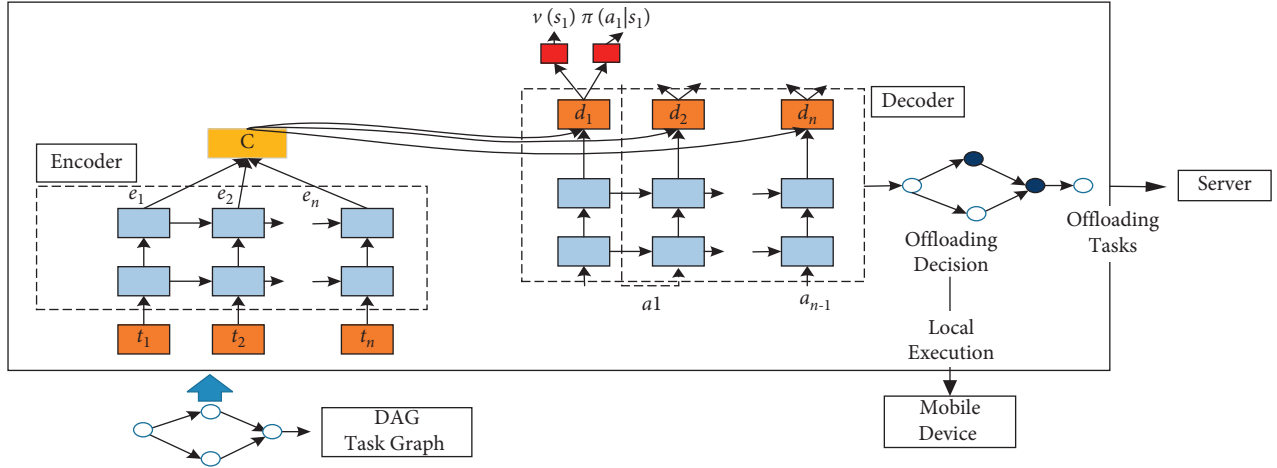


FIGURE 2: Task offloading architecture.

$$\text{rank}(t_i) = \begin{cases} T_i^o, & \text{if } (t_i \in K), \\ T_i^o + \max_{t_j \in \text{child}(t_i)} (\text{rank}(t_j)), & \text{if } (t_i \notin K), \end{cases} \quad (17)$$

where $T_i^o = \min(T_i^L, T_i^c + T_i^{ul} + T_i^{dl})$ indicates the delay of completing the task, and K is the end node of the task.

The functions of Encoder and Decoder are set to g_{enc} and g_{dec} , and the output e_i of Encoder is represented as shown in formula (18). The aggregate intermediate vector c of the Encoder encoded output e_i and the d_j in Decoder can be expressed as shown in formulae (19) and (20):

$$e_i = g_{enc}(t_i, e_{i-1}), \quad (18)$$

$$c = \frac{1}{n} \sum_{i=0}^n e_i, \quad (19)$$

$$d_j = g_{dec}(c, a_{j-1}, d_{j-1}), \quad (20)$$

where d_j is connected to two fully connected layers that approximately represent the $v(s_j)$ and $\pi(a_j|s_j)$ of task t_j , where $v(s_j)$ and $\pi(a_j|s_j)$ represent value functions and policy functions, respectively. During training, the action a_j is generated by strategy $\pi(a_j|s_j)$ sampling, where $a_j = \max \pi(a_j|s_j)$ represents the execution action of task t_j .

4.2. An Offloading Model of Federated Reinforcement Learning Based on Attention Mechanism. In federated learning, multiple device nodes participate in the training and aggregate the model weights shared by the service BS nodes. In order to make better use of the parameter information provided by each equipment node, the service BS node needs to allocate different weight coefficients according to their contribution. Considering the difference between equipment models, the Attention mechanism can be applied to achieve the demand. In this paper, an FRL task offloading model based on Attention mechanism is proposed, and the training process is as follows:

- (i) Associate the mobile device with the serving BS and report the local mobile device status to the BS. After receiving the information, BS sends the previous round of global model parameters to the local device to perform the steps of model release.
- (ii) When the local mobile device learns that it can participate in the training, it receives the global model parameters obtained from the BS.
- (iii) Each mobile device begins to train its local model parameters based on global model parameters and local data. After the local training, each mobile device collects its training evaluation indicators (e.g., average reward, average loss, etc.) during the training phase and sends them and local model parameters to BS. Then, BS calculates the aggregation weight of each equipment according to the training evaluation index. In this paper, attention mechanism is used to provide aggregation weights for different mobile devices. The update process for weighted federation aggregation is shown in Figure 3.

In this model, the indexes of m mobile devices are input into the Encoder of the model, and Query is the optimal goal in Decoder. By comparing the similarity between the indexes of each mobile device model and the optimal goal, the contribution weight of each device model to the optimal goal is calculated. In this paper, the average loss, average reward and the computing power of mobile devices are used to calculate the attention weight of the contribution of the global model.

- (i) Average loss: the quality of the model is related to the size of the loss, the greater the loss means that the model cannot well deal with the problems to be solved. On the contrary, the smaller the loss, the better the performance of the model, so we can consider using it. So, the loss can be evaluated on a model. The average loss (L_{avg}) is the average calculation of the loss function of a certain period of time by a device.

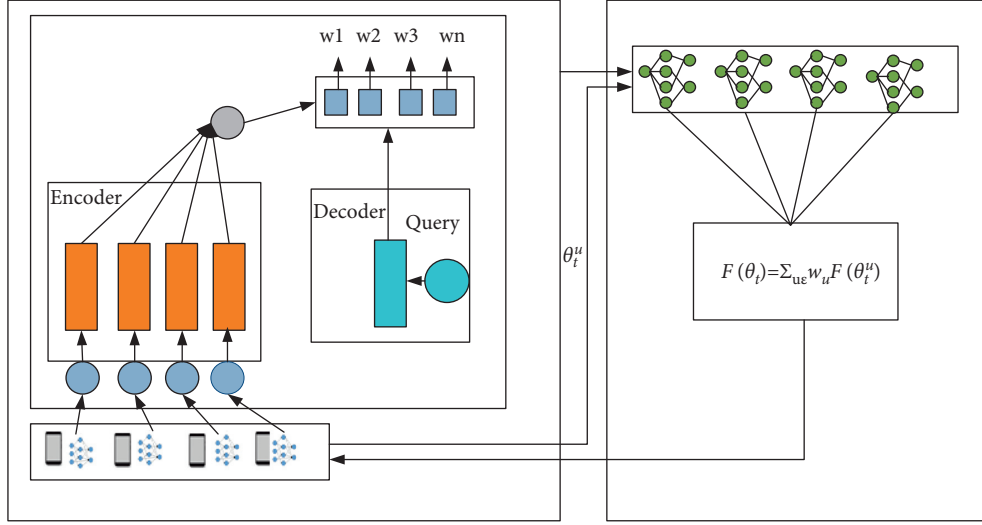


FIGURE 3: Federated Reinforcement Learning model of Attention mechanism.

- (ii) Average reward: contrary to the average loss, the average reward can also evaluate the quality of the model. When the reward is larger, the effect of the model is better, and when the reward is smaller, the model may have some deviation from the problem solved. Therefore, the average reward can also evaluate the model. Average reward (R_{avg}) is the average reward for the mobile device over a certain period of time.
- (iii) Computing power of mobile devices: the computing power of mobile devices can affect whether tasks need to be offloaded. When the computing power of mobile devices is large, the delay caused by local processing tasks is usually far less than the delay caused by transmitting task data, so the device will get a better user experience effect when the tasks are processed locally. On the contrary, if the processing power of the mobile device is too small, the processing delay will be very long, so it is better to offload the task to the edge server. Therefore, the computing power of the mobile device (com_u) also has a certain impact on the model parameters.

In this paper, we use the above index $K = [L_{avg}, R_{avg}, com_u]$ to evaluate the attention mechanism. The goal of the training model is to get more rewards and achieve the minimum loss. Therefore, the optimization goal set in this paper is shown in formula (21). Q consists of minimizing the loss function, maximizing the reward, and maximizing the computing power.

$$Q = [\min(L_{avg}), \max(R_{avg}), \max(com_u)]. \quad (21)$$

Then, the weight of the contribution of the device is calculated by using the Attention mechanism. In this paper, we use the method of dot product to calculate the similarity between Q and K to get the corresponding weight, and then use the softmax function to normalize the weight, and introduce the weight factor w_u to calculate the contribution of

the local model to the global model. The specific calculation is shown as follows:

$$w_u = \text{Attention}(Q, K) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right), \quad (22)$$

where $\sqrt{d_k}$ represents the dimension of K , from which the contribution of each mobile device model is calculated by using the Attention mechanism, and the weight of the model is weighted to calculate the parameters of the global model. After aggregation, this parameter is sent to each local model. The optimized global parameter aggregation formula is shown in the following formula. The specific algorithm is shown in Algorithm 1.

$$F(\theta_t) = \sum_{u \in U} w_u F(\theta_t^u). \quad (23)$$

5. Experiment

5.1. Experimental Setup

5.1.1. Setting Model Parameters

System Model Parameters. Considering 20 users and one edge server node, the task arrival rate is 0.3, the user CPU clock rate is 2.5 GHz, the clock rate of the server is 41.8 GHz, and the uplink and downlink transmission rate is 14 Mbps. In this paper, a synthetic DAG generator is implemented with [22].

Network Model Parameters. The Encoder and Decoder of seq2seq neural network are set to two layers of LSTM, with 256 hidden units in each layer. The learning rate is set to 0.001 and the optimizer adopts RMSprop algorithm.

5.1.2. Hardware Parameters. The hardware setting in the experiment can be seen in Table 1.


```

(1) for each equipment in parallel do
(2)   for local step  $j = 1, \dots, n$  do
(3)     for each step do
(4)       Sample action  $a \sim \pi_\theta(s)$ , reward and  $s'$ 
(5)       Input  $s$  and  $s'$  to get  $Q_w(s', a')$  and  $Q_w(s, a)$ 
(6)        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$ 
(7)        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a) \delta$ 
(8)        $w \leftarrow w + \beta \delta$ 
(9)        $a \leftarrow a', s \leftarrow s'$ 
(10)    end for
(11)     $K = [L_{avg}, R_{avg}, com_u]$ 
(12)  end for
(13)  $Q = [\min(L_{avg}), \max(R_{avg}), \max(com_u)]$ 
(14)  $w_u = \text{Attention}(Q, K)$ 
(15)  $F(\theta_t) = \sum_{u \in U} w_u F(\theta_t^u)$ 
(16) end for

```

ALGORITHM 1: A Federated Reinforcement Learning task offloading model integrated with Attention mechanism.

TABLE 1: Hardware parameter.

Configuration	Parameter
CPU	Intel® Xeon® CPU E5645 2.40 GHz
Memory	8 GB
Operating system	Windows 10
Development tools	PyCharm

5.2. Results and Analysis. According to [22], a synthetic DAG generator is implemented for simulation. 22 task graphs are used as training sets for training, and the other three task graphs are tested, and each task graph is composed of 20 tasks. The training is carried out on 22 training sets, and the experimental results are shown in Figure 4, which shows the change of the average reward with the increase of the number of iterations. It can be seen that after 1000 iterations, the average reward as a whole tends to be stabilized and shows an upward trend during the training process.

Learning rate plays a very important role in deep learning and training. Figure 5 shows that on the premise that the number of subtasks in each task graph is 10; the learning rate is set to 0.1, 0.01, and 0.0001, respectively. The experimental comparison of the proposed algorithms shows that the model with a learning rate of 0.001 has the highest delay. The performance of the model with learning rate of 0.01 and 0.0001 is similar, but the effect is the best at 0.001. Therefore, this paper uses 0.001 as the learning rate of model training.

Considering 20 users, the task graph of each user is composed of 10 subtasks. The Federated Reinforcement Learning Offloading Computation algorithm with Attention (FRLOC-Attention) is compared with DQN algorithm, Greedy algorithm, and Actor-Critic based offloading algorithm (AC). The experimental results are shown in Figure 6. The results show that after 500 episodes, FRLOC-Attention shows better performance and minimum delay, while AC and DQN offloading algorithms are not as effective as FRLOC-Attention, while greedy algorithm (Greedy) produces a larger delay, about 590 ms, and has a larger delay

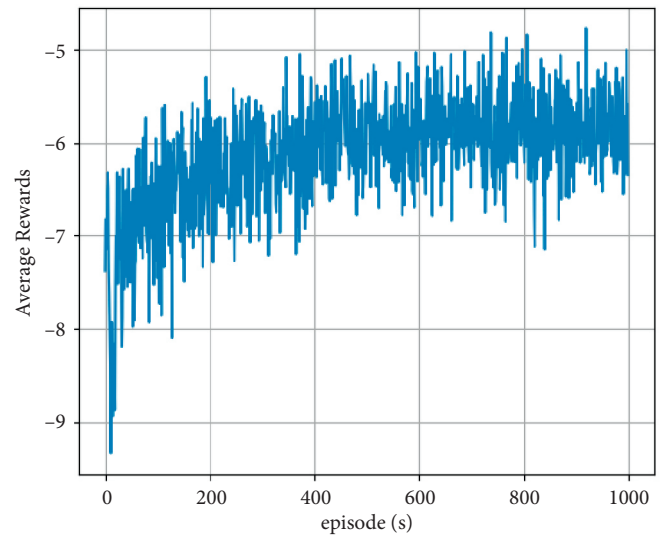


FIGURE 4: Average reward of training set.

than DQN algorithm and AC algorithm. Generally speaking, FRLOC-Attention has the lowest latency and the best performance.

In order to show that the proposed algorithm has a certain generality, this paper chooses to experiment on the number of different devices. Figure 7 shows that the number of devices is 10, 20, and 30, and the DAG task graph of each device includes 20 subtasks. From the experimental results, we can see that no matter the number of devices, the task processing delay of FRLOC-Attention is obviously lower than that of DQN algorithm and Greedy algorithm. The performance of AC algorithm is similar to that of FRLOC-

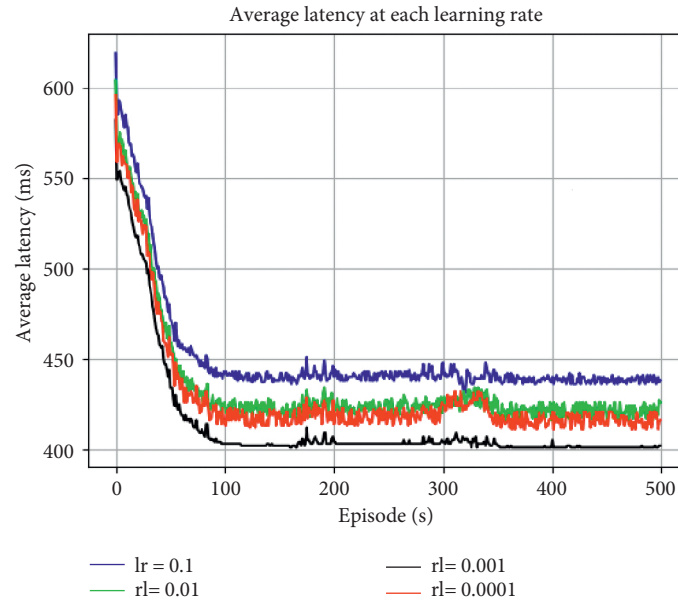


FIGURE 5: The average delay of the algorithm under different learning rates.

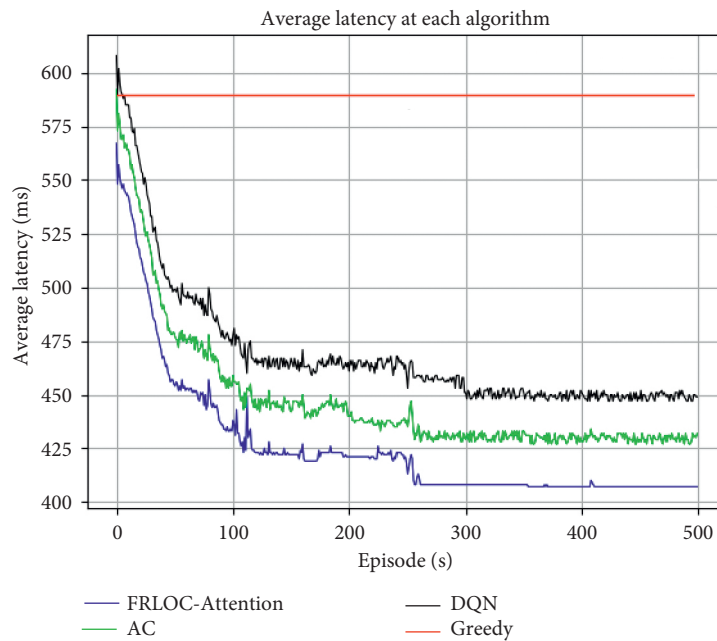


FIGURE 6: The number of tasks is 10, the average delay on different algorithms.

Attention; this is because AC algorithm is included in the process of FRLOC-Attention training model, but because this algorithm proposed uses federated reinforcement

learning combined with Attention to make the model more generalized, when there are multiple devices, the performance of task offload is better.

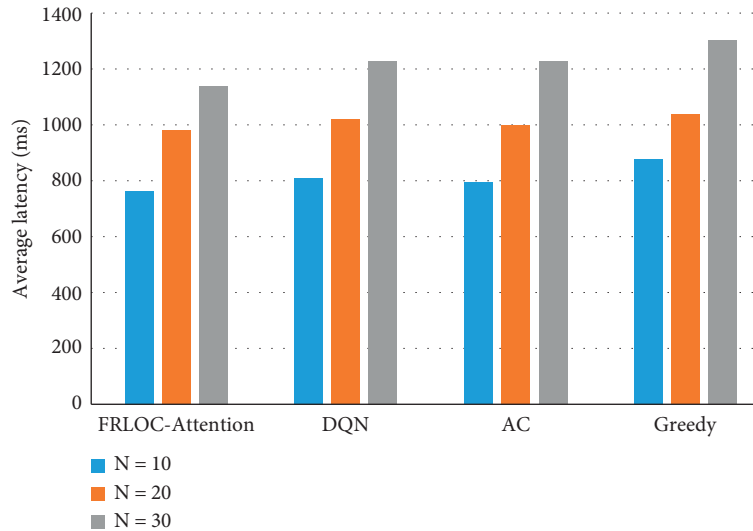


FIGURE 7: The average delay of each algorithm in the case of different number of devices.

6. Conclusion

Considering the lack of model generalization caused by the lack of data sharing among multiple users and the privacy data leakage caused by data sharing, this paper proposes a federated reinforcement learning task offloading algorithm based on Attention mechanism to solve the relevance problem of multiple mobile device training models. Experiments show that the proposed FRLOC-Attention algorithm saves more time than DQN algorithm, AC algorithm, and Greedy algorithm. On the basis of this algorithm, we can further consider the optimization of energy consumption caused by data transmission during task offloading in the future.

Data Availability

The authors declare that all data used to support the findings of this study are included within the article.

Disclosure

The findings achieved herein are solely the responsibility of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Chinese National Key Research Plan (2019YFB1405402).

References

- [1] H. Allam, N. Nassiri, A. V. Rajan, and J. Ahmad, "A critical overview of latest challenges and solutions of mobile cloud computing," in *Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017.
- [2] M. Patel, "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, vol. 29, pp. 854–864, 2014.
- [3] Y. Siriwardhana, P. Poramage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.
- [4] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, December 2017.
- [5] D. Wang, X. Tian, H. Cui, and Z. Liu, "Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network," *China Communications*, vol. 17, no. 8, pp. 31–44, 2020.
- [6] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6790–6805, 2018.
- [7] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.
- [8] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.
- [9] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2014.
- [10] P. Sun, H. Zhang, H. Ji, and X. Li, "Task allocation for multi-APs with mobile edge computing," in *Proceedings of the 2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, August 2018.
- [11] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.
- [12] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for

- mobile edge computing,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3341–3356, 2020.
- [13] R. Zhang, G. Gu, Z. Liu, and X. Wang, “Reinforcement learning theory, algorithms and its application,” *Control Theory & Applications*, vol. 17, no. 5, p. 6372, 2000.
- [14] S. Nath and J. Wu, “Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems,” *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.
- [15] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, “Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing,” in *Proceedings of the International Conference on Machine Learning and Intelligent Communications*, Hangzhou, China, July 2018.
- [16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, “Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.
- [17] M. Tang and V. W. S. Wong, “Deep reinforcement learning for task offloading in mobile edge computing systems,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.
- [18] L. Sweeney, *Simple Demographics Often Identify People Uniquely*, Carnegie Mellon University, Pittsburgh, PA, USA, 2000.
- [19] C. Nadiger, A. Kumar, and S. Abdelhak, “Federated reinforcement learning for fast personalization,” in *Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, June 2019.
- [20] N. C. Luong, D. T. Hoang, S. Gong et al., “Applications of deep reinforcement learning in communications and networking: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
- [21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2014, <https://arxiv.org/abs/1409.0473>.
- [22] H. Arabnejad and J. G. Barbosa, “List scheduling algorithm for heterogeneous systems by an optimistic cost table,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2013.