Federated Deep Reinforcement Learning for Online Task Offloading and Resource Allocation in WPC-MEC Networks

LIANQI ZANG, XIN ZHANG, AND BOREN GUO
School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, Beijing, 100876, China
Corresponding author: Xin Zhang (e-mail: zhangxin@bupt.edu.cn).

ABSTRACT Mobile edge computing (MEC) is considered a more effective new technological solution for developing the Internet of Things (IoT) by providing cloud-like capabilities for mobile users. This article combines wireless powered communication (WPC) technology with an MEC network, where a base station (BS) can transfer wireless energy to edge users (EUs) and execute computation-intensive tasks through task offloading. Traditional numerical optimization methods are time-consuming approaches for solving this problem in time-varying wireless channels, and centralized deep reinforcement learning (DRL) is not stable in large-scale dynamic IoT networks. Therefore, we propose a federated DRL-based online task offloading and resource allocation (FDOR) algorithm. In this algorithm, DRL is executed in EUs, and federated learning (FL) uses the distributed architecture of MEC to aggregate and update the parameters. To further solve the problem of the non-IID data of mobile EUs, we devise an adaptive method that automatically adjusts the FDOR algorithm's learning rate. Simulation results demonstrate that the proposed FDOR algorithm is superior to the traditional numerical optimization method and the existing DRL algorithm in four aspects: convergence speed, execution delay, overall calculation rate and stability in large-scale and dynamic IoT.

INDEX TERMS Mobile edge computing, federated learning, deep reinforcement learning, online computing offload, wireless powered communication

I. INTRODUCTION

The internet of things (IoT) technology has entered the next stage with the comprehensive combination of artificial intelligence (AI) and 5G network technology [1]. However, deploying a large number of IoT devices will face two technical challenges: (1) Many IoT-enabled devices are resource-constrained, with only insufficient storage space and limited computing power. (2) Because a large number of IoT devices are located in coverage areas, deployment costs will rise substantially by manually replacing the battery or utilizing wired charging. With the maturity of wireless powered communication (WPC) technology, energy stations can provide power to edge user (EU) batteries more efficiently and steadily over the air [2]. Meanwhile, the evolution of mobile edge computing (MEC) technology enables EUs to offload computing-intensive and delay-sensitive tasks to MEC servers, effectively completing more complicated work [3]. Therefore, the combination of MEC and WPC technology has solved the limitations of IoT devices in terms of battery charging and computing power, providing a better environment for IoT development [4].

The advantage of the WPC-MEC network is deployment of energy stations near EUs. The energy station provides power for EUs in real time through WPC, and EUs apply the collected energy to transmit their computing tasks to the MEC server. Hence, the core of the WPC-MEC network is the joint allocation of energy transmission and task offloading of all EUs. Many traditional methods are based on the Lyapunov optimization approach, online dynamic task scheduling, and game theory to resolve this problem [5]. However, in complex MEC networks, the computational complexity of these approaches is hard to control. In addition, these methods are challenging to apply to real-time offloading policies because channel gains are continuously changing dramatically in fast fading channels, and they need to constantly re-solve optimization problems [6]. In recent years, deep reinforcement learning (DRL) has become a new research trend to address optimization problems [7]. Because
DRL can adjust its strategy with an unstable environment and learn more about complex MEC scenarios in continuous states, some studies have progressed in applying DRL methods to optimize task offloading in MEC networks. However, the current DRL algorithms aggregate data for centralized training. In view of the increasingly complex mobile networks and more configurable parameters, the centralized training data approach may not be efficient and increases the risk of data leakage. In addition, in a large-scale MEC network, the DRL algorithm will encounter some problems in the implementation process (for example, when the number of EUs = 50, the state-action space of the MEC system is $2^{50}$, which is not realistic for Q-learning and DQN algorithms). In the multi-agent DRL (MA-DRL) algorithm, a single agent cannot observe the global environment and easily falls into the local optimal solution. To tackle this problem, federated learning (FL), as a new paradigm applied in the MEC field, has the following advantages: (1) Privacy protection of personal data. FL transfers the parameter updates of the DNN model instead of the original data to the server for aggregation so that user data are only stored locally. Although the transmission of training model parameters still has the risk of leakage, the security of the data has been well guaranteed to a certain extent. With the development of FL, many researches have strengthened data privacy and security on FL architecture. [8], [9] and [10] enhance user privacy protection through secure multi-party computing (SMC), homomorphic encryption (HE) and differential privacy (DP). [11] proposed a Byzantine-robust FL algorithm to ensure privacy and filter the abnormal parameters of Byzantine opponents to solve member attacks and reverse attacks. (2) Better adaptation to the large-scale dynamic MEC environment [12]. We combine FL and DRL to propose an online offloading framework, which is used to jointly optimize the task offloading decision of each EU in the fast fading channel and the time allocation between WPC and EU calculation offloading. At the same time, in view of the impact of non-IID data on FL, an adaptive learning rate method is proposed to ensure the rapid convergence of the algorithm and the stability of the results. Compared to existing numerical optimization and DRL methods, our contributions are summarized as follows:

1) We propose an online algorithm based on FL and DRL in a WPC-MEC system with a BS and multiple EUs. This proposed method, federated DRL-based online task offloading and resource allocation (FDOR), has better performance in improving the overall EU calculation rate and reducing the execution latency of making offloading decisions and resource allocation.

2) Compared with the DRL algorithm proposed in [6], FDOR distributes the DRL model to the EU for training. This method of combining FL and DRL addresses problems that droo can only apply to fixed EU scenarios. Moreover, in the large-scale dynamic MEC network, the algorithm still maintains good stability and low computational delay, and its performance is better than other existing DRL algorithms.

3) In the edge network environment, the difference in the range and mode of each EU’s movement leads to EU’s non-IID channel gains, and the size of offloading tasks for different EUs is also non-IID. To mitigate the impact of non-IID data on the system, we propose an adaptive method to adjust the learning rate to speed up the convergence of the FDOR algorithm. At the same time, it ensures that the total calculation rate of the EU is close to the best result.

The remainder of this article is organized as follows. Section 2 reviews the state of the art and related work. We describe the system model and problem in Section 3. In Section 4, we formulate the detailed design of the FDOR algorithm. Section 5 presents simulation results. Finally, our paper is concluded in Section 6.

II. RELATED WORK

As an extension of cloud computing, MEC alleviates the pressure from limited EU resources by generating computing offloading strategies and allocating computing and communication resources. [13] studied the task allocation scheduling scheme of power consumption and execution latency maximization in an MEC system with energy harvesting capability based on the Lyapunov optimization method. [14] designed three algorithms, including heuristic search, the reformulation linearization technique, and semi-definite relaxation, to solve the above problems. [15] shows that the increase of EUs is also non-IID. To mitigate the impact of non-IID data on the system, we propose an adaptive method to adjust the learning rate to speed up the convergence of the FDOR algorithm. At the same time, it ensures that the total calculation rate of the EU is close to the best result.

Since DRL aims to maximize long-term benefits, it is well suited to complex MEC systems and can adaptively make offloading decisions and allocate resources. [16], [17] studied an offloading strategy based on a deep Q-network (DQN) to optimize computational performance. [18] studied a double DQN-based strategy to maximize the long-term utility performance. However, as the number of EUs increases, the resulting discrete offload operations grow exponentially, which is not amenable to a DQN-based approach. [6] presents an online offloading algorithm for MEC networks based on DRL (DROO), making real-time resource allocation possible for wireless MEC networks in fading channels.

However, the DRL models proposed in the above work perform centralized training by aggregating data. In the face of a large-scale dynamic MEC network, the convergence of the centralized DRL model may be challenging to achieve or even unable to learn effectively. In a dynamic MEC network with frequent access, reasonably designing the size of the DRL model to ensure the system’s stability has also become a problem. With the rapid growth of big data, data privacy and information security are increasingly valued [19]. FL, as a learning method of distributed training data, can effectively solve the above problems. [20] shows that the increase of edge devices can speed up the convergence speed of the federated learning model. [21] propose a matching with the
incomplete preference list algorithm based on FL to address the problem of the latency minimization in a large-scale MEC network scenario. [22] proposed a non-interactive FL algorithm that guarantees privacy. This method can ensure that privacy is not leaked even when multiple FL participants collude. While protecting user privacy, [23] considered how to verify the correctness of the server’s aggregated data and proposed a verifiable federated learning privacy protection framework (VerifyNet) based on an analysis of the methods to solve the above problems. In this paper, we add FL to the DRL model. Taking full advantage of the characteristics of DRL and FL, we present an online FDOR strategy that can adaptively allocate computing and communication resources in large-scale dynamic MEC scenarios and protect the privacy of personal data to a certain extent, providing a new method to resolve the issues above.

III. SYSTEM MODEL

A. NETWORK MODEL

In this paper, we consider a WPC-MEC network composed of a BS and a set of $N = \{1, 2, ..., N\}$ mobile EUs. As shown in Fig. 1, a BS consists of an MEC server and an access point (AP) that transmits wireless energy to the EU, receives offloading tasks from the EUs, and returns the calculated results to the EUs. The computation tasks offloaded to the BS are performed in the MEC server. Each EU has a single antenna and a rechargeable battery that receives and stores collected energy to power the calculation and use of the device. This article considers a binary offloading strategy where tasks can be computed locally in the EU or offloaded to a BS. Let $x_{t,i}$ be an offloading decision of the $i$th EU in the $t$th time frame, where $x_{t,i} = 1$ represents that the EU offloads the calculation task to the BS and $x_{t,i} = 0$ represents that the EU computes the task itself.

We divide the time into $L$ consecutive time frames, which are denoted as a set of $\mathcal{L} = \{1, 2, ..., L\}$. Each time frame is set to be the same length $T$ and less than the channel coherence time. To simplify the model, the EU’s task offloading and received energy are executed in the same frequency band. Therefore, time-division multiplexing (TDD) circuits are applied to allocate WPC and the time of the task offloading for each EU. In fast fading channels, wireless channel gains largely determine the speed of communication between the EU and BS. $h_{t,i}$ represents the wireless channel gains in the $t$th time slot between the BS and $i$th EU, assuming that the channel gains are constant in each time frame but vary with the position of EUs and the random fading factor between different time frames. Suppose that the working EUs have a computation task in a time frame that needs to be accomplished by utilizing the energy transmitted by the AP. We let $D_{t,i}$ denote the number of computation cycles required to process 1 bit of an offloading task. We divide a time frame $T$ into three parts, as shown in Fig. 1. To ensure maximum energy utilization, WPC executes at the beginning of $aT$ in each time frame, $a_t \in [0, 1]$. The $i$th EU can obtain $E_{t,i} = \mu h_{t,i} PaT$ in the $t$th frame, where $P$ denotes the transmission power of the AP, and $\mu \in (0, 1)$ denotes the energy collection efficiency [2]. Then, the intermediate time is used for task offloading for EUs in this time frame. We represent $\tau_{t,i} T$ as the offloading time of the $i$th EU in the $t$th time frame, $\tau_{t,i} \in [0, 1]$. Finally, the remaining time is used to calculate the offloading tasks of EUs. We represent $\eta_{t,i} T$ as the calculation time of the $i$th EU in the $t$th time frame, $\eta_{t,i} \in [0, 1]$. Considering that the size of the calculated results returned is much smaller than the size of the offloading task, we ignore the time during which AP returns the calculated results, making a time frame composed of task offloading, computation and WPC, that is,

$$\sum_{t=1}^{N} \tau_{t,i} + \sum_{t=1}^{N} \beta_{t,i} + \alpha_t \leq 1, \quad \forall t \in \mathcal{L} \quad (1)$$

B. LOCAL COMPUTING MODEL

EUs complete their tasks in calculation units in the local computing model. Let $f_{t,i}$ and $t_{t,i}$ denote the processor calculation speed and calculation time for the $i$th EU in the $t$th time frame, respectively. The calculated energy consumption is constrained by $k_i f_{t,i}^3 t_{t,i} \leq E_{t,i}$, where $k_i$ denotes the calculation of the energy efficiency factor [24]. To maximize the amount of processing data in a time frame, every EU should calculate the task throughout the time frame and exhaust the collected energy. Therefore, we obtain

$$f_{t,i}^\text{max} = T \quad (2)$$

$$f_{t,i}^\text{max} = (E_{t,i} / k_i T)^{1/3} \quad (3)$$

Therefore, the local calculation rate is

$$r_{t,i} = f_{t,i}^\text{max} / D_{t,i} T = (\mu P / k_i)^{1/3} (a_t h_{t,i})^{1/2} / D_{t,i} \quad (4)$$

C. EDGE COMPUTING MODEL

EUs should offload tasks to the AP at the $\tau_{t,i} T$ specified by the WPC-MEC network in the edge computing model. To maximize the calculation rate, EUs should run out of energy at the end of task offloading, so the transmission power of the EU is

$$P_{t,i}^\text{max} = E_{t,i} / \tau_{t,i} T \quad (5)$$

Accordingly, the calculation rate of the EU in the edge computing model is

$$r_{t,i}^\text{e} = B \frac{\tau_{t,i}}{v_u} \log_2(1 + \frac{\mu P h_{t,i}^2 a_t}{\tau_{t,i} N_0}) \quad (6)$$

where $B$ denotes the communication bandwidth, $v_u$ denotes the communication overhead ratio, and $N_0$ denotes the noise power.

Finally, the time required to calculate the offloading task $\beta_{t,i} = f_{t,i} / D_{t,i}$, where $f_{t,i}$ denotes the MEC server calculation speed.
D. PROBLEM FORMULATION

Our objective is to maximize the EU’s computation rate using FDOR algorithms for real-time generation offloading policies and wireless channel resource allocation. According to the local calculation and edge calculation formula given in (4) and (6), we can obtain the computation rate of all EUs over the whole system time.

\[
Q = \sum_{t=1}^{L} \sum_{i=1}^{N} ((1 - x_{t,i})r_{l,i}^t + x_{t,i}r_{o,i}^t)
\]  

(7)

We consider that the wireless channel gains \( h = \{h_1, h_2, ..., h_i | i \in \mathcal{N} \} \) and the number of computation cycles for offloading tasks \( D = \{D_1, D_2, ..., D_i | i \in \mathcal{N} \} \) vary in the different time frames, whereas the others are fixed parameters. Because channel gains \( h \) and the number of computation cycles for offloading tasks \( D \) for different time frames is independent and irrelevant, we can define optimization problems as maximizing the weighted sum computation rate of the WPC-MEC system at one time frame:

\[
(P1) : Q^* (h, D) = \max_{x, \tau, a} \sum_{i=1}^{N} ((1 - x_{i})r_{l,i}^t + x_{i}r_{o,i}^t)
\]

subject to \( \sum_{i=1}^{N} \tau_{i} + \sum_{i=1}^{N} \beta_{i} + a \leq 1, \quad a \geq 0, \quad \tau_{i} \geq 0, \quad \beta_{i} \geq 0, \quad x_{i} \in \{0, 1\}, \forall i \in \mathcal{N} \)  

(9)

(10)

Problem (P1) is a mixed-integer-non-convex program for which the solution requires exponential computational complexity, so we decompose problem (P1) into two sub-problems: offloading decision-making and wireless channel resource allocation.

FL sinks the DRL model to the EU for training and periodically aggregates the model to make an offloading decision. In each time frame, the EU obtains the offloading decision through the DRL model and uploads it to the MEC server. Once the MEC server obtains the offloading decision of all EUs \( x = \{x_1, x_2, ..., x_i | i \in \mathcal{N} \} \), (P1) becomes a wireless resource allocation problem (P2). Since (P2) is a convex problem, the weighted sum of the maximum computing rate...
can be easily obtained through the one-dimensional two-segment search algorithm with $O(N)$ complexity proposed in [25].

$$(P2) : Q^*(h, D) = \max_{\tau, a} \sum_{i=1}^{N} ((1 - x_i) r_i + x_i r_i^0)$$

subject to $\sum_{i=1}^{N} \tau_i + \sum_{i=1}^{N} \beta_i + a \leq 1$, $a \geq 0, \tau_i \geq 0, \beta_i \geq 0, \forall i \in \mathcal{N}$

IV. THE FDOR ALGORITHM

The framework of the FDOR algorithm is shown in Fig. 2. The FDOR algorithm consists of four main components: generation of offloading action, update of offloading policies, DNN model aggregation and adaptive learning rate method. These four steps are described in detail below.

A. OFFLOADING ACTION GENERATION

FL distributes the DNN that generates offloading actions to the EU. At the beginning of the $t$th time frame, each EU obtains channel gains $h_{t,i}$, the number of computation cycles $D_{t,i}$, and inputs it to the DNN. The DNN obtains the corresponding relaxed offloading action $x_{t,i}^*$ by parameter operation $x_{t,i}^* = f_{\theta_{t,i}}(h_{t,i}, D_{t,i})$ and uploads to the BS. After all EUs’ offloading actions are obtained at the BS, the MEC server organizes the offloading actions into

$$x_t^* = \{x_{t,i}^*|x_{t,i}^* \in [0, 1], i = 1, ..., N\}$$

Then, we use an order-preserving quantization method to quantify relaxed offloading action $x_t^*$ as $K$ binary offloading actions. The order-preserving quantization method follows the rules below:

1) The first binary offloading decision is

$$x_{t,i}^1 = \begin{cases} 1 & x_{t,i}^* > 0.5, \\ 0 & x_{t,i}^* \leq 0.5. \end{cases}$$

2) To generate the $k$-th binary offloading decision, we first order the entries of $x_t^*$ with respect to their distances to 0.5. Then, we calculate

$$x_{t,i}^k = \begin{cases} 1 & x_{t,i}^* > x_{t,i}^{k-1}, \\ 1 & x_{t,i}^* = x_{t,i}^{k-1} \text{ and } x_{t,i}^{k-1} \leq 0.5, \\ 0 & x_{t,i}^* = x_{t,i}^{k-1} \text{ and } x_{t,i}^{k-1} > 0.5, \\ 0 & x_{t,i}^* < x_{t,i}^{k-1}. \end{cases}$$

The order-preserving quantization method produces a larger and more diverse distance between offloading actions [6] proved that the method has good convergence performance.

After obtaining $K$ binary offloading actions, we solve (P2) separately for each binary offloading action $x_k$, to obtain the weighted sum computation rate $Q^*(h, D)$. Finally, we select $\{\hat{x}_t, \hat{a}_t, \hat{\tau}_t\}$ corresponding to the best $Q^*(h, D)$ as the final result. The $t$th EU uses $x_{t,i} \in \{\hat{x}_t\}$ as an offloading action. The BS allocates $a_t T$ as WPC and $\tau_t$ as EUs offload the task to the MEC server. After the EUs obtain the offloading action, the newly acquired state-action $(h_{t,i}, D_{t,i}, \hat{x}_{t,i})$ is added to the memory.

To further reduce the execution delay, we use the adaptive $K$ algorithm proposed in [6]. In the $t$th time frame, $k$ denotes the index of the binary offloading strategy corresponding to the best $Q^*(h, D)$. $K = \min(\max(k_{t-1}, ..., k_{t-\Delta}) + 1, N)$ when $t \mod \Delta = 0$, $\Delta$ denotes the updating interval for $K$.

B. OFFLOADING POLICY UPDATE

We set the training interval $\delta$ of the DNN as the offloading policy update frequency. When offloading policy updates, EUs obtain a portion of state actions from memory to train the DNN. In this article, we use the Adam algorithm [26] to train the DNN parameters $\theta_{t,i}$ of all EUs to reduce the training loss as follows:

$$L(\theta_t) = -\frac{1}{|M|} \sum_{\beta \in \mathcal{M}} ((x_{t,i}^*)^t \log f_{\theta_{t,i}}(h_{t,i}, D_{t,i}) + (1 - x_{t,i}^*) \log(1 - f_{\theta_{t,i}}(h_{t,i}, D_{t,i})))$$

where $|M|$ denotes the size of the training data. For the design of memory, we use a fixed memory size. New state-action data will override old data when memory overflows. The advantage of this design is that the new data provide better results than the old data because of constantly being trained by the DNN. New data can also be used to train DNN parameters more efficiently. This closed-loop reinforcement learning mechanism can continuously improve the offloading strategy of DNN until it converges.

C. DNN MODEL AGGREGATION

We assume that the fraction of EUs $S$ is selected in each round, and EUs upload DNN parameters in every $E$ time frame. At the beginning of each round, EUs upload locally trained DNN parameters to the MEC server. The MEC server aggregates all the DNN parameters uploaded by EUs to generate the parameters of the global DNN in the next round and transmits global DNN model parameters to all EUs. Here, we use FedAvg [19] as the model aggregation method.

$$\theta_{t,g} = \sum_{i=1}^{N} \frac{n_i}{n} \theta_{t,i}$$

where $n$ denotes the sum of $n_i$ and $n_i$ denotes the number of tasks completed in the $i$th EU. EUs use local data to train DNN parameters during offloading action updates. This process continues until the entire algorithm converges.

In the MA-DRL algorithm, each agent learns its policies individually and no parameter sharing occurs. In this case, the gradient of the training loss is unbounded. Therefore, the convergence of the multi-agent reinforcement learning

VOLUME 4, 2016
algorithm cannot be guaranteed. [27], [28] proved the convergence of FL on non-convex problems through mathematical derivation, and found that the convergence and stability are better than the MA-DRL algorithm.

Compared with the DROO algorithm, FDOR has several advantages in transferring the DNN training model from the MEC server to the EU: (1) The DNN model is not limited by the number of EUs. The algorithm follows the prescribed process and will not be significantly affected by turning on and off large-scale EUs. (2) Contrary to DROO, as the number of EUs increases, the total amount of training data for FDOR will undoubtedly increase, which speeds up the rate of model aggregation and allows the algorithm to maintain excellent performance over time.

In the transmission DNN model between a BS and EUs, the channel propagation time is relatively short, and the DNN model is usually aggregated through multiple time frames (usually 10, 20, or 50). Therefore, we safely ignored the time of the transmission DNN model when resource allocation was used in this paper.

D. ADAPTIVE LEARNING RATE UPDATE

Due to the diversity of IoT devices, the channel gains and size of offloading tasks generated among different EUs are heterogeneous. The channel gain distribution of the EU is non-IID due to the difference in mobility, which has always been an important and challenging problem in FL. Therefore, under the training of heterogeneous data, there will be significant differences between the optimal local model and the optimal global model of the EU. This will lead to global model performance degradation problems and slow convergence speed of the standard FL method under non-IID data.

[20] proves that the convergence rate of FedAvg is $O(E/L)$ for strongly convex smooth problems. In addition, the convergence speed of FedAvg with non-IID data must meet a necessary condition: even if the entire gradient is used, the learning rate must decrease; otherwise, the solution will deviate from the optimal solution. Therefore, we propose a method to adjust the learning rate adaptively. We define the learning rate of the Adam optimizer in the EU as

$$
\alpha_t = \frac{\alpha_{max}}{\lambda} \ln((1 - \psi) * (e^{\lambda} - 1) + 1) + \epsilon
$$

where $\lambda$ denotes the rate of decline in learning rates, $\alpha_{max}$ denotes the initial learning rate in DNN training, $\psi$ denotes the accuracy of the offloading actions, and $\epsilon > 0$ is a tiny number, ensuring that the DNN is continually being trained. This method allows the learning rate to adjust to the accuracy of the current training results, making the DNN training update of each EU more suitable for its own data, and avoiding difficulties in FL convergence caused by a learning rate that is too large or too small. Furthermore, in the time-varying fast fading channel, if an EU movement mode or the type of an offloading task is significantly different from previous occurrences, the method can improve the learning rate through accuracy feedback and quickly adjust the update of DNN parameters, maintaining the stability of the system.

Finally, the pseudo-code of the FDOR algorithm is provided in Algorithm 1.

V. SIMULATION RESULTS

In this section, we used PyTorch 1.8.1 to implement the FDOR algorithm in Python and performed simulations to evaluate its performance. All simulations were performed on an Intel Core i5-6300HQ 2.30 GHz CPU with 8 GB memory. The time-varying wireless channel gains $h_t = [h_{t,1}, h_{t,2}, ..., h_{t,N}]$ are generated from a Rayleigh fading
Algorithm 1 The FDOR algorithm

**Input:** wireless channel gains $h_i$, and the number of computation cycles for offloading tasks $D_i$ at each time frame $t$

**Output:** offloading actions $\hat{x}_i$, of all EUs $\theta_t$ for WPC, and $\hat{r}_i$ for offloading tasks of EUs.

1. Set the total time frame $t$, model aggregation interval $E$, the number of quantized actions $K$ and the training interval $\delta$, the number of quantized actions $K$

2. Initialize the DNN parameters $\theta$ of all EUs

3. for $t = 1, 2, ..., L$ do

   4. Generate action $x_{t,i} = f_{\theta_{t,i}}(h_{t,i}, D_{t,i})$ of each EU

   5. Quantize $x_{t,i}$ into $K$ binary actions $x_k = g_K(x_{t,i})$

   6. Compute $Q^*$ for all $\{x_k\}$

   7. Select the best action $x_{t,i} = \arg\max x_k$

   8. for $i = 1, 2, ..., N$ do

   9. Add state-action $(h_{t,i}, D_{t,i}, x_{t,i})$ to the memory

   10. if $t$ mod $\delta = 0$ then

        11. Select data set $\{(h_j, D_j, x_j)|j \in M\}$

        12. Train the DNN and update $\theta_{t+1,i} \leftarrow \theta_{t,i}$

        13. Update learning rate $\alpha$ by (16)

   end if

   end for

16. if $t$ mod $E = 0$ then

17. Upload $\theta_{t,i}$ of all EUs to MEC

18. $\theta_{t,g} \leftarrow \sum_{i=1}^{N} \frac{n_i}{n} \theta_{t,i}$

19. $\{\theta_{t+1,i}|i = 1, ..., N\} \leftarrow \theta_{t,g}$

20. end if

21. end for

**TABLE 1.** The setting of environment parameters

| Parameters | Values |
|------------|--------|
| $P$        | 3W     |
| $v_{iu}$   | 1.1    |
| $\phi$     | 100    |
| $B$        | 2 MHz  |
| $N_0$      | $10^{-10}$ |
| $\mu$      | 0.51   |
| $k_i$      | $10^{-26}$ |
| $f_o$      | 10000  |
| Distance between EUs and AP $d_{e,i}$ | $(1.0, 10.0) m$ |
| Number of computation cycles for offloading task $D_{t,i}$ | $(20, 200)$ |
| Antenna gain $A_d$ | 4.11 |
| Carrier frequency $f_c$ | 915 MHz |
| Path loss index $d_e$ | 2.8 |

channel model as $h_{t,i} = h_{i}(a_{t,i})$, where $h_{i}(a_{t,i}) = A_d(2^{\gamma_{f_i}d_{e,i}})$ and $a_{t,i}$ is an independent random channel fading factor. The setting of the environmental parameters is shown in Table 1.

In the WPC-MEC network, because of the computing power limitations of the EU, we use as small neural networks as possible to accomplish our algorithm. Considering the computing power of the EU and the WPC-MEC network’s performance, we simply use a fully connected DNN consisting of one input layer, two hidden layers, and one output layer, where the first and second hidden layers have 24 and 12 hidden neurons, respectively. We use ReLU and sigmoid as the activation functions in the hidden layer and output layer, respectively. In addition, we set the training interval $\delta = 5$, memory size as 1024, training batch size $|M| = 128$, global DNN model aggregation interval $E = 10$, fraction of EUs selected in each training round $S = 1.0$, and initial learning rate $\alpha = 0.03$.

### A. CONVERGENCE PERFORMANCE IN DIFFERENT SCENARIOS

We primarily evaluate the convergence performance of the FDOR algorithm in different scenarios. To compare the calculation rate of each algorithm more intuitively, we use the normalized calculation rate as the evaluation standard of model performance.

$$\hat{Q}(h, D) = \frac{Q^*(h, D)}{Q_{max}^*(h, D)} \quad (17)$$

Since the time of the enumeration method increases exponentially, we use the coordinate descent algorithm in [25] to find the optimal solution $Q_{max}^*(h, D)$.

We first evaluated the convergence speed of the training loss of the FDOR algorithm. In Fig. 3, we plot the average training loss $L_{\theta_{t,i}}$ of the DNN model with $N = 20$ mobile EUs. The training loss $L_{\theta_{t,i}}$ is reduced and stabilized at approximately 0.1 after $t > 2500$, which means that the FDOR has automatically completed the update of the offloading action strategy and converges to excellent performance.

![FIGURE 3. Training loss $L_{\theta_{t,i}}$ of FDOR with 20 mobile EUs](image)

Then, we evaluated the FDOR performance of the WPC-MEC network with $N = 20$ mobile EUs. After every 100 time frames, 10–50% of EUs are randomly reselected, and the new location of selected EUs is within the range $(1.0,10.0)$. In Fig. 4, the curve and the shadow represent the average $\hat{Q}$ in 200 time frames and maximum-minimum $\hat{Q}$ in the past 50 time frames, respectively. MA-DRL is the multi-agent DRL algorithm that the DRL model is executed independently in EUs but not aggregated. $\hat{Q}$ reaches the optimal solution $Q_{max}^*(h, D)$ for $t > 2000$. Nevertheless, the DROO and MA-DRL algorithm has not fully converged after 10,000 time frames. Moreover, its offloading strategy is unstable due to the large fluctuation of $\hat{Q}$. 

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
In the dynamic MEC network environment, a large number of EUs can be frequently turned off/on. Therefore, we evaluate FDOR for the WPC-MEC network where mobile EUs randomly turned off/on. At the beginning of the simulation, we only turned on 50% of the EUs and let them work. For every 1000 time frames, we randomly reselect 10–50% of the EUs to stop working and obtain the other portion as EUs that are working. As shown in Fig. 5, $\hat{Q}$ reaches the optimal solution $Q_{t,\text{max}}(h, D)$ for $t > 2000$. After the EUs are turned off/on, $\hat{Q}$ only fluctuates slightly when $t = 5000$ and 6000 but soon converges, and the average $\hat{Q}$ in the last 2000 iterations is always greater than 0.98. Because the DNN is trained independently on each EU, it will stop training and not participate in the aggregation of the global DNN after the EU is turned off. After the EU is turned on, the DNN in the EU will synchronize the global DNN model and start training. Therefore, the random turning off/on of the EU is equivalent to the FDOR algorithm transforming from selecting the entire EU training model to selecting part of the EUs. Although the fluctuation is relatively large at the beginning of training, it has no major impact on the convergence speed and performance of the algorithm. In contrast, the random turning off/on of the EU leads to frequent changes to the overall offloading strategy of the DROO algorithm, which makes it impossible to learn from it effectively. Therefore, the DROO and MA-DRL algorithm has difficulty converging and has poor stability. Therefore, we can conclude that FDOR can better adapt to dynamic WPC-MEC networks.

We evaluate the ability of FDOR to support WPC-MEC networks with different mobile distribution EUs. For example, in a WPC-MEC network with $N = 20$ EUs, the $i$th EU moves in the range of $\left(\frac{i}{2}, \frac{i+1}{2}\right)$, and we set different offloading task size ranges and location distributions to simulate various types of EUs.

We compare FDOR with other FL algorithms, namely, FedAvg and q-FedSGD, in the WPC-MEC network, which verifies the efficiency and stability of FDOR. First, we briefly introduce FedAvg and q-FedSGD.

1. FedAvg: In every training round, the MEC server takes all EUs as participants and uses a simple average aggregation method. As one of the most classical algorithms of FL, this algorithm is used as a reference method in many papers related to FL. To ensure the convergence of FedAvg, we use L2 regularization in the Adam optimizer and set the parameter to 0.5.

2. q-FedSGD [29]: Using the improved parameter aggregation algorithm, the calculation expression of the $t+1$ round global DNN model parameters is as follows:

$$\theta_{g,t+1} = \theta_{g,t} - \frac{\sum_{i=1}^{N} \Delta_{i,t}}{\sum_{i=1}^{N} h_{i,t}}$$

(18)

The expressions of $\Delta_{i,t}$ and $h_{i,t}$ are as follows

$$\Delta_{i,t} = J_{i}^{q}(\theta_{t,i})\nabla J_{i}(\theta_{t,i})$$

(19)

$$h_{i,t} = qJ_{i}^{q-1}(\theta_{t,i})\|\nabla J_{i}(\theta_{t,i})\|^{2} + lJ_{i}^{q}(\theta_{t,i})$$

(20)

where $J_{i}(\cdot)$ denotes the loss function, $\nabla J_{i}(\cdot)$ denotes the drop gradient of the loss function, and $q, l$ are constants. This method aims to ensure the fairness of FL and reduces the variance by adjusting the combined weights.

In Fig. 6, we plot the normalized calculation rate $\hat{Q}$ of FDOR, FedAvg and q-FedSGD. We can see that the average $\hat{Q}$ of the FedAvg algorithm can only be maintained at approximately 0.95, which does not achieve the best offloading strategy. In contrast, q-FedSGD achieves good computation rate performance but has a slow convergence speed and is barely stable when $t > 6000$. Therefore, both of them are not easy to apply to the complex and dynamic MEC network environment. However, the FDOR algorithm combines FedAvg and q-FedSGD, whose offloading strategy reaches the optimal offloading strategy when $t > 3500$. This experimental result proves the effectiveness of the adaptive learning rate algorithm. In contrast, the constant learning rate is combined with the $E$ steps of the local epoch update that may be biased to form a sub-optimal update scheme. In the case of $E > 1$ and any fixed learning rate, FedAvg will not converge to the optimal value. With the training of the DNN, the adaptive learning rate algorithm gradually increases the accuracy rate and reduces the learning rate. The gradually reduced learning rate can gradually eliminate this deviation. At the same time,
if the EU mobile mode or offloading task type is significantly different from previous occurrences, this method can adjust the learning rate through accuracy feedback to maintain the stability of the system.

In Fig. 7i, We choose $\lambda = 2$ through the experimental results because too large or too small $\lambda$ will affect the performance and convergence speed of the algorithm.

B. COMPARISON OF CALCULATION RATES

In this section, we compare our FDOR algorithm with DROO and two other benchmark algorithms regarding the weighted sum rate performance:

Linear relaxation algorithm [24]: The LR algorithm relax the binary offloading decision variable $x_{t,i}$ to a real number between 0 and 1, as $\hat{x}_{t,i} \in [0, 1]$. Then, the problem (P1) with constraints (9)-(10) is convex with respect to $\{\hat{x}_{t,i}\}$ and the optimal solution can be easily found. Then we can use $\hat{x}_{t,i}$ to determine the binary offloading strategy $x_{t,i}$:

$$x_{t,i} = \begin{cases} 1 & \hat{r}_{t,i}^o > r_{t,i}^o, \\ 0 & \hat{r}_{t,i}^o \leq r_{t,i}^o. \end{cases} \quad (21)$$

Coordinate Descent algorithm [25]: The CD algorithm randomly generates an offloading decision $x$, and then in each round inverts each $x_i$ and calculates the sum calculation rate, and saves the offloading decision corresponding to the largest calculation rate as the initial offloading strategy for the next round. The CD algorithm can achieve near-optimal performance in different environments.

In Fig. 8, we compare the computation rate of mobile EUs under $N = \{10, 20, 30, 40, 50\}$. Since the range of WPC is generally 10 meters level [30], when the number of EUs reaches 50, we can consider that the equipment density of the MEC network is larger. Therefore, we define the number of EUs in a large-scale MEC network as 50 [31], [32]. Considering the trade-off between the dynamic MEC network and long-term benefits, FDOR and DROO were trained with 10000 independent wireless channel gains before the evaluation, and their offloading strategy tended to converge. Therefore, the results are the average $\hat{Q}$ of 2000 independent wireless channels. The results show that under different EU numbers, the performance of the FDOR algorithm is equal to the optimal performance of the CD algorithm, and is significantly better than the DROO and LR algorithm.

Similarly, in Fig. 9, we compare the computation rate of EUs with non-IID data achieved by different FL algorithms with $N = \{10, 20, 30, 40, 50\}$. We see that as EUs $N$ increases, the performance of FedAvg and q-Fed gradually declines. FDOR still achieves similar near-optimal performance with $N = \{10, 20, 30, 40, 50\}$, and the performance is slightly reduced compared to the CD algorithm only when $N = 50$, which shows that the FDOR algorithm solves the problem of large-scale MEC networks with different distributed EU scenarios.

More specifically, we focus on comparing the performance of FDOR and DROO in the mobile EU environment. In Fig. 10, we evaluated the stability of FDOR and DROO under varying the number of EUs and $N$ by plotting boxplots. We can see that for FDOR, the average is approximately 0.99, and the median is always up to 1.0. The range of average $\hat{Q}$
FIGURE 7. The normalized computation rate $\hat{Q}$ under different hyperparameters. (a) memory sizes, (b) batch sizes, (c) training intervals, (d) local epochs, (e) learning rates, (f) training fraction of EUs, (g) number of binary offloading decisions, (h) update interval of $K$ and (i) decline rate of learning rates.

FIGURE 8. The weighted sum computation rate $Q^*$ under different offloading algorithms with mobile EUs.

FIGURE 9. The weighted sum computation rate $Q^*$ under different FL schemes with non-IID EU data.
TABLE 2. CPU Execution Latency under Different Offloading Algorithm

| Number of EUs | FDOR | DROO | LR | CD |
|---------------|------|------|----|----|
| 10            | 0.018 s | 0.031 s | 0.18 s | 0.20 s |
| 20            | 0.030 s | 0.047 s | 0.58 s | 1.3 s |
| 30            | 0.038 s | 0.10 s | 0.94 s | 4.0 s |
| 40            | 0.049 s | 0.11 s | 1.5 s | 9.1 s |
| 50            | 0.055 s | 0.18 s | 2.2 s | 16 s |

is (0.94, 1.0) only when $N = 50$ and stable at other times. In contrast, the average and median of the DROO algorithm are less than 1.0 and decrease significantly as EU's increase. When the number of EUs $N \geq 30$, the range of $\hat{Q}$ of the DROO algorithm even extended to (0.6, 1.0).

This simulation result proves that FL can enhance the stability of large-scale MEC networks. Generally, the DRL algorithm is limited by the number of EUs (for example, in the DROO algorithm, the number of input values = 20 when $N=10$, and the number of input values = 100 when $N=50$). This difference will seriously affect DNN model performance. The FDOR algorithm sinks the DNN model into the EU; regardless of how the number of EUs changes, it will not change the number of inputs and outputs of the DNN model, so it is not limited by the number of EUs. Overall, the performance of the DROO algorithm is not stable when $N \geq 30$, and many offloading actions differ greatly from the best $\hat{Q}$, whereas the performance of the FDOR algorithm is always excellent and stable.

C. EXECUTION LATENCY

Finally, we evaluate the execution delay of the FDOR algorithm. Similarly, the average execution delay of the DROO and FDOR algorithms listed in Table 2 has passed the training of 10,000 independent wireless channels. From Table 2, we can see that FDOR has much less CPU execution latency compared to other offloading algorithms. Especially when $N = 50$, FDOR only needs 0.05 seconds to generate offloading actions, but DROO has a CPU execution delay that takes approximately 3 times longer, and CD and LR are even extended by 290 times and 44 times, respectively.

The WPC-MEC network may correspond to a large-scale dynamic IoT network in reality. Under normal circumstances, we believe that the channel coherence of the unchanged channel is not less than 2 seconds, so the time frame can be assumed to be 2 seconds. We can see that in the WPC-MEC network with $N=50$, the execution delay of FDOR is 0.055 s, which only accounts for less than 3% of the total time frame, which is also an acceptable overhead in reality. In contrast, the execution delay of DROO accounts for 9%, which may have a certain impact on system performance. The execution time of the LR and CD algorithms is even longer than the time frame, which is unrealistic.

VI. CONCLUSION

In this paper, we propose an online offloading algorithm FDOR based on the combination of DRL and FL. Based on DROO, this algorithm transfers the DRL model from the MEC server to the EU, which improves the accuracy of offloading action. Meanwhile, we propose an adaptive learning rate adjustment method to improve the convergence of FL under the non-IID of EU data. Therefore, the FDOR algorithm solves the problem of difficulty in convergence in the mobile EU environment. Furthermore, compared with the DROO algorithm, the FDOR algorithm has a better effect on convergence speed, computation rate, and CPU execution delay. With the increase in the number of EUs, the FDOR’s offloading actions also maintain excellent performance.

REFERENCES

[1] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, and L. Zhao, “Deep reinforcement learning based dynamic resource management for mobile edge computing in industrial internet of things,” IEEE Trans. Ind. Inform., vol. PP, no. 99, pp. 1–1, 2020.

[2] S. Bi, C. K. Ho, and R. Zhang, “Wireless powered communication: Opportunities and challenges,” IEEE Commun. Mag., vol. 53, no. 4, pp. 117–125, 2014.

[3] W. Shi, C. Jie, Z. Quan, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[4] F. Wang, J. Xu, X. Wang, and S. Cui, “Joint offloading and computing optimization in wireless powered mobile-edge computing systems,” IEEE Trans. Wireless Commun., vol. PP, no. 99, pp. 1–1, 2017.

[5] Q. V. Pham, F. Fang, V. N. Ha, M. J. Piran, and Z. Ding, “A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art,” IEEE Access, vol. PP, no. 99, pp. 1–1, 2020.

[6] L. Huang, S. Bi, and Y. Zhang, “Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks,” IEEE Trans. Mobile Comput., vol. PP, no. 99, pp. 1–1, 2020.

[7] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, and I. K. Douc, “Applications of deep reinforcement learning in communications and networking: A survey,” IEEE Commun. Surveys Tuts., vol. PP, no. 99, pp. 1–1, 2019.

[8] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, “An adaptive federated learning scheme with differential privacy preserving,” Future Generation Computer Systems, vol. 127, pp. 362–372, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21003617

[9] G. Baeck, S. Kim, D. Nana, and J. Park, “Enhancing differential privacy for federated learning at scale,” IEEE Access, vol. 9, pp. 148 090–148 103, 2021.

[10] C. Fang, Y. Guo, Y. Hu, B. Ma, L. Feng, and A. Yin, “Privacy-preserving and communication-efficient federated learning in internet of things,” Computers & Security, vol. 103, pp. 102 199, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821000237

[11] X. Ma, Y. Zhou, L. Wang, and M. Miao, “Privacy-preserving byzantine-robust federated learning,” Computer Standards & Interfaces, vol. 80, pp. 103 561, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548921000568

[12] Y. Wei, N. C. Luong, D. T. Hoang, Y. Jiao, and C. Miao, “Federated learning in mobile edge networks: A comprehensive survey,” IEEE Commun. Surveys Tuts., vol. PP, no. 99, pp. 1–1, 2020.

[13] Guanglin, Zhang, Wenqian, Zhang, Yu, Cao, Demin, Li, and Wang, “Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices,” IEEE Trans. Ind. Inform., vol. 14, no. 9, pp. 4411–4420, 2018.

[14] J. Liu and Q. Zhang, “Offloading schemes in mobile edge computing for ultra-reliable low latency communications,” IEEE Access, vol. 1–1, 2018.

[15] N. Shan, X. Cui, and Z. Gao, “drl + fl: An intelligent resource allocation model based on deep reinforcement learning for mobile edge computing,” Computer Communications, vol. 160, pp. 14–24, 2020.

[16] Z. Tong, X. Deng, F. Ye, S. Basodi, and Y. Pan, “Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment,” Information Sciences, vol. 537, 2020.

[17] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhang, “Learning-based computation offloading for iot devices with energy harvesting,” IEEE Trans. Veh. Technol., 2019.
FIGURE 10. Comparing the median, 50%, and 75% quartile $Q^*$ and outliers of the FDOR-(a) and DROO-(b) algorithms in different EUs

[18] 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, pp. 1–1, 2018.

[19] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, “Communication-efficient learning of deep networks from decentralized data,” 2016.

[20] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” 2019.

[21] D. Chen, C. S. Hong, L. Wang, Y. Zha, and Z. Han, “Matching theory based low-latency scheme for multi-task federated learning in mec networks,” IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1–1, 2021.

[22] M. Hao, H. Li, X. Luo, G. Xu, and S. Liu, “Efficient and privacy-enhanced federated learning for industrial artificial intelligence,” IEEE Transactions on Industrial Informatics, vol. PP, no. 99, pp. 1–1, 2019.

[23] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, “Verifinet: Secure and verifiable federated learning,” IEEE Transactions on Information Forensics and Security, vol. PP, no. 99, pp. 1–1, 2019.

[24] S. Guo, B. Xiao, Y. Yang, and Y. Yang, “Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing,” in IEEE INFCOM 2016 - IEEE Conference on Computer Communications, 2016.

[25] S. Bi, J. Ying, and Zhang, “Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading,” IEEE Trans. Wireless Commun., vol. PP, no. 99, pp. 4177–4190, 2017.

[26] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” Computer Science, 2014.

[27] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, “Federated multi-agent actor-critic learning for age sensitive mobile edge computing,” IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1–1, 2021.

[28] F. Haddadpour and M. Mahdavi, “On the convergence of local descent methods in federated learning,” arXiv preprint arXiv:1910.14425, 2019.

[29] T. Li, M. Sanjabi, A. Beirami, and V. Smith, “Fair resource allocation in federated learning,” 2019.

[30] S. Bi, Y. Zeng, and R. Zhang, “Wireless powered communication networks: An overview,” IEEE Wireless Communications, vol. 23, no. 2, 2015.

[31] Y. Xu, P. Cheng, Z. Chen, M. Ding, Y. Li, and B. Vucetic, “Task offloading for large-scale asynchronous mobile edge computing: An index policy approach,” IEEE Transactions on Signal Processing, vol. 69, pp. 401–416, 2020.

[32] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, “Deep-learning-based joint resource scheduling algorithms for hybrid mec networks,” IEEE Internet of Things Journal, 2019.