Cooperative Edge Caching via Federated Deep Reinforcement Learning in Fog-RANs

Min Zhang[1], Yanxiang Jiang[1,2], Fu-Chun Zheng[1,2], Mehdi Bennis[3], and Xiaohu You[1]
[1]National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China
[2]School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518055, China
[3]Centre for Wireless Communications, University of Oulu, Oulu 90014, Finland
E-mail: { mzhing@seu.edu.cn, yxjiang@seu.edu.cn, fzheng@ieee.org, mehdi.bennis@oulu.fi, xhyu@seu.edu.cn }

***Abstract*—In this paper, cooperative edge caching problem is investigated in fog radio access networks (F-RANs). By considering the non-deterministic polynomial hard (NP-hard) property of this problem, a federated deep reinforcement learning (FDRL) framework is put forth to learn the content caching strategy. Then, in order to overcome the dimensionality curse of reinforcement learning and improve the overall caching performance, we propose a dueling deep Q-network based cooperative edge caching method to find the optimal caching policy in a distributed manner. Furthermore, horizontal federated learning (HFL) is applied to address issues of over-consumption of resources during distributed training and data transmission process. Compared with three classical content caching methods and two reinforcement learning algorithms, simulation results show the superiority of our proposed method in reducing the content request delay and improving the cache hit rate.**

***Index Terms*—Fog radio access network, cooperative edge caching, deep reinforcement learning, dueling deep Q-network, horizontal federated learning.**

I. Introduction

With the advent of 5G, the number of mobile devices and applications are increasing rapidly, generating the massive amount of data putting huge traffic pressure on wireless cellular networks [1]. Fog radio access network (F-RAN) is a very promising way to solve the problem of cellular network communication congestion [2]. In F-RANs, edge caching places popular contents in fog access points (F-APs), closer to users [3]. F-APs can effectively reduce the load pressure of the backhaul link and content transmission delay [4]. Due to the limited communication resources and local storage capacity in F-AP, how to cache the most popular content is the prime challenge of edge caching [5].

There has been a flurry of research work in content placement of edge caching. In recent years, reinforcement learning has become a popular tool to optimize cooperative caching of content in F-RANs. In [6], the authors used two types of recurrent neural networks to predict user mobility and content popularity, and then, deep reinforcement learning framework is applied to implement dynamic caching decisions and optimize content delivery issues. In [7], content caching placement in edge networks is modeled as a stochastic game. The authors proposed an alternate Q-learning caching algorithm with multi-agent cooperation to solve the stochastic game. In [8], the hidden Markov process is adopted to characterize the content request model, which is used to describe the traffic demand of temporal and spatial fluctuations in F-RANs. A value function approximation method is combined with Q-learning to find the optimal caching strategy. In order to overcome the dimensionality curse of reinforcement learning and improve the speed of convergence, a distributed edge caching algorithm based on the deep Q-network was considered in [9], [10]. In [11], a learning automata-based Q-learning algorithm was investigated to obtain the optimal caching action selection for the predicted model in a random and stable environment. In the heterogeneous networks, a Q-learning based content caching solution was exploited in [12] to realise content caching securely. The double auction game was quoted to describe the collaborative interaction between edge server and content provider, so that each participant can get the maximum benefit from it. The above-mentioned methods based on reinforcement learning can find the optimal caching strategy dynamically and efficiently. However, these methods can damage sensitive data of user when they use user data to train the model directly, especially in some commercial and industrial scenarios. Meanwhile, these caching methods deploy learning agents in a single user or a single F-AP. The individual training of separate and low-relevant learning agents can also lead to an additive waste of resources.

Motivated by the above discussion, a cooperative caching method based on federated deep reinforcement learning (FDRL) framework is proposed to find the optimal caching policy in F-RANs. First, content caching scenario composed of multiple cooperative F-APs is formulated. We use the M-Zipf distribution of random parameters to describe the dynamic content popularity in the scenario. Then, we adopt the dueling deep Q-network to learn the user request content data in each F-AP. Based on the content popularity distribution, the dueling deep Q-network can make optimal caching decision and reduce content request delay. Finally, we apply horizontal federated learning (HFL) to aggregate the local model of each F-AP into a global model in the cloud server. Through the iterative training of the global model, the data privacy of users in each F-AP can be protected and the resource cost of individual training for separate learning agents can be reduced.

The rest of this paper is organised as follows. The system model and problem formulation are introduced in Section II. Section III describes the FDRL-based cooperative caching policy elaborately. Simulation results are given in Section IV

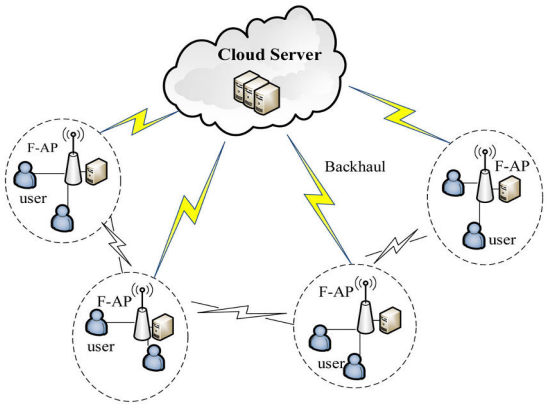

Fig. 1: Illustration of cooperative caching scenario in F-RANs.

and Section V summarizes this paper.

II. System Model and Problem Formulation

A. System Model

The considered cooperative caching scenario is illustrated in Fig. 1, where a cloud server is connected with several F-APs and multiple users are served by each F-AP. The set of F-APs is $\mathcal{N}=\{1,2,...,n,...,N\}$ and the set of users served by F-AP n is denoted by $\mathcal{U}_n=\{1,2,...,u_n,...,U_n\}$. Request contents of all users is denoted by a content library $\mathcal{F}=\{1,2,...,f,...,F\}$ and the size of content f is L. It is assumed that all requested contents from users are cached in the cloud server. However, each F-AP can only cache a certain number of request contents because of its limited cache capacity. Let C $(C \ll F)$ denotes the storage capacity of F-AP n. Furthermore, we assume that all F-APs have the same cache capacity and all contents have the same size.

We consider that time is divided into discrete time slots. The set of time slots is denoted by $\mathcal{T}=\{1,2,...,t,...,T\}$ and the network operates in each time slot. At time slot t, the network needs to perform three operations. At the first stage, each F-AP sends its local information table of caching contents to the neighboring F-APs and the cloud server to share its content caching state. At the second stage, each F-AP transmits content to user based on the requested content of user. If the requested content is cached in the local F-AP, the local F-AP directly sends the content to the user. If the requested content is not cached in the local F-AP, the local F-AP will deliver the content to the user from the neighboring F-APs or the cloud server. Finally, each F-AP will update its local caching contents according to the requested contents from its serving users for the next time slot. In time slot t, let $P_f(t)$ denote the global popularity of content f, and $P_f(t)$ also represents the probability distribution of content f being requested by all users in the network. The set of global popularity of content is $\mathcal{P}(t)=\{P_1(t), P_2(t), ..., P_f(t), ..., P_F(t)\}$. Let $p_{nf}(t)$ denote the probability distribution of content f in F-AP n that is requested by users. We consider $P_f(t) = \sum_{n\in\mathcal{N}} p_{nf}(t)$, and $P_f(t)$ satisfies the Mandelbrot-Zipf (M-Zipf) distribution [13] as follows:

$$P_f(t) = \frac{{i_f}^{-\alpha_t}}{\sum_{j\in\mathcal{F}} {i_j}^{-\alpha_t}}, \forall f \in \mathcal{F}, \tag{1}$$

where i_f represents the rank of content f in descending order of global content popularity, α_t denotes the skewness factor of M-Zipf distribution. The larger α_t is, the fewer contents occupy the majority of user requests.

B. Problem Formulation

Based on the discussion of the above system model, there are three ways for users to obtain their requested content:

- The users obtain the requested content from the local F-AP serving them. If the requested content is cached in the local F-AP, the local F-AP directly delivers the content to users. Each F-AP transmits content to users from its local cache with the content transmission delay d_{n1}. Let $D_{\text{F-U}}(t)$ denote the average delay of fetching content from the local F-AP for the whole user requests in slot t, then we have:

$$D_{\text{F-U}}(t)=\sum_{f=1}^{F}\sum_{n=1}^{N} m_{f,n}(t) * p_{nf}(t) * d_{n1}, \tag{2}$$

  where $m_{f,n}(t)$ is a binary variable representing whether users fetch the requested content from the local F-AP serving them or not in slot t. $m_{f,n}(t)=1$ indicates that users can obtain the requested content from the local F-AP in slot t, otherwise $m_{f,n}(t)=0$.
- The users obtain the requested content from the other neighboring F-APs. If the requested content is not cached in the local F-AP, users can obtain the requested content from the neighboring F-APs. Let d_{n2} denote the transmission delay between F-APs. The average delay of obtaining content from the neighboring F-APs is denoted by $D_{\text{F-F-U}}(t)$, then we have:

$$D_{\text{F-F-U}}(t)=\sum_{f=1}^{F}\sum_{n=1}^{N}\sum_{k=1}^{N} m_{f,n,k}(t) * p_{nf}(t) * (d_{n1}+d_{n2}), \tag{3}$$

  where $\forall k \in \mathcal{N}, n \neq k$. $m_{f,n,k}(t)=1$ indicates that users obtain the requested content from the neighboring F-APs, otherwise $m_{f,n,k}(t)=0$.
- The users get the requested content from the cloud server. If the requested content is not cached in the local F-AP and the neighboring F-APs, users need to fetch the content from the cloud server. Let d_{n3} denote the transmission delay of sending the content to the F-AP from the cloud server, and $D_{\text{C-F-U}}$ denotes the average delay of obtaining the content from the cloud server for the whole user requests, then we have:

$$D_{\text{C-F-U}}(t)=\sum_{f=1}^{F}\sum_{n=1}^{N} m_{f,n,\text{C}}(t) * p_{nf}(t) * (d_{n1}+d_{n3}), \tag{4}$$

where $m_{f,n,\mathrm{C}}(t) = 1$ indicates that users can obtain the requested content from the cloud server, otherwise $m_{f,n,C}(t) = 0$.

Without loss of generality, we consider $d_{n1} < d_{n2} \ll d_{n3}$. In time slot t, for a given caching state in F-AP n, the total content request delay for any user can be expressed as follows:

$$D_{\text{total}}(t) = D_{\text{F-U}}(t) + D_{\text{F-F-U}}(t) + D_{\text{C-F-U}}(t). \tag{5}$$

The goal is to find the optimal caching strategy π^* to minimize the total content request delay $D_{\text{total}}(t)$ in the network. Therefore, considering the limited storage space of F-AP and the integer characteristics of cache decision variable, cooperative caching problem is described as follows:

$$\underset{\pi^*}{\operatorname{argmin}} D_{\text{total}}(t) \tag{6}$$

$$\text{s.t.} \sum\nolimits_{f\in\mathcal{F}} L * m_{f,n}(t) \le C, \tag{6a}$$

$$m_{f,n}(t), m_{f,n,k}(t), m_{f,n,\mathrm{C}}(t) \in \{0,1\}. \tag{6b}$$

The constraint (6a) requires the amount of locally cached content in each F-AP cannot exceed its maximum storage capacity. The constraint (6b) is to ensure that each content is cached in a single F-AP at most, which improves the diversity of locally cached content in each F-AP.

In the cooperative caching network we have discussed, the cache hit rate of content is considered to be a good indicator for evaluating network performance. In this paper, the cache hit rate of each F-AP in slot t is calculated as follows:

$$h_n(t) = 1 - \sum_{k\in\mathcal{N}} \sum_{f\in\mathcal{F}} \left(m_{f,n,k}(t) + m_{f,n,C}(t)\right) * p_{nf}(t). \tag{7}$$

The cache hit rate indicates the probability that the requested content of users can be obtained from the local cache of the F-AP serving them.

III. FDRL-BASED COOPERATIVE CACHING POLICY

In this section, we propose an FDRL-based cooperative caching policy to solve the content caching problem in (6). We combine the dueling deep Q-network in deep reinforcement learning with horizontal federated learning. The dueling deep Q-network is able to make optimal caching decision based on the requested content information of users and the popularity of content in each F-AP. However, individual training of separate learning agents can lead to an additive waste of resources and sensitive data of user may be endangered during training process. We apply horizontal federated learning to aggregates all the local models from each F-AP to obtain a global model in the cloud server. By minimizing the loss function of the global model, the content request delay of the entire network is reduced and the total cache hit rate is improved. Federated learning can not only enable better cooperation between each F-AP, but also protect user data privacy because model parameters are transmitted between F-AP and the cloud server, not user data [14].

A. Deep Reinforcement Learning Framework

Among deep reinforcement learning algorithms, the deep Q-network (DQN) can effectively combine deep learning and reinforcement learning. The main reason is the DQN has three core elements:

(1) Objective function: Based on the Q-learning algorithm, an objective function learned by deep learning is constructed. The neural network in the DQN is used to approximate the action value function $Q(s(t), a(t))$ in a high-dimensional and continuous state, where $s(t)$ and $a(t)$ are the state and the action of time slot t. In order to obtain the objective function that the neural network can learn, the DQN constructs a loss function through the Q-learning algorithm. The update formula of the Q-learning algorithm is:

$$\begin{aligned} &Q^*(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \\ &\alpha(r + \gamma \max_{a(t+1)} Q(s(t+1), a(t+1)) - Q(s(t), a(t))), \end{aligned} \tag{8}$$

where r is a reward for transferring to state $s(t+1)$ after performing action $a(t)$ under state $s(t)$. γ is the future discount factor and α is the learning rate. According to (8), the loss function of the DQN is:

$$L(\theta) = \left[\left(\text{Target Q} - Q(s(t), a(t), \theta)\right)^2\right], \tag{9}$$

where θ is the weight parameter of the prediction neural network model. The target Q-value (i.e., Target Q) is:

$$\text{Target Q} = r + \gamma \max_{a(t+1)} Q\left(s(t+1), a(t+1), \theta^-\right), \tag{10}$$

where θ^- is the weight parameter of the target neural network model. After obtaining the loss function of the DQN, the gradient descent method can be used to obtain the gradient update of the weight parameter θ of the loss function $L(\theta)$ in (9).

(2) Target network: The DQN uses two neural networks for learning: prediction network $Q(s(t), a(t), \theta)$ is to evaluate the Q-value function of the current state-action pair; target network $Q(s(t), a(t), \theta^-)$ is to generate the target Q-value in (10). The DQN updates the parameter θ of the prediction network according to the loss function in (9). After each M rounds of iteration, the parameters θ of the prediction network are copied to the parameters θ^- of the target network.

(3) Experience playback mechanism: The DQN introduces an experience playback mechanism to store the experience sample data obtained by the interaction between the agent and the environment in the replay memory $\mathcal{D}$ at each time step. Then, small batches of data are randomly selected from the replay memory $\mathcal{D}$ for neural network training. The DQN saves a large amount of historical experience sample data, and each experience sample data is stored in $(s(t), a(t), r(t), s(t+1))$. It indicates the learning agent reaches the new state $s(t+1)$ after performing the action $a(t)$ under the state $s(t)$ and obtains the corresponding reward $r(t)$. By introducing the experience playback mechanism, randomly

sampling small batches for training is helpful to remove the correlation and dependence between samples and reduce the deviation in the evaluation of the value function. It solves the problem of data correlation and non-static distribution to make the network model easier to converge.

B. Dueling Deep Q-Network

We use deep reinforcement learning network to solve the content caching problem in (6), and each F-AP is our learning agent. We model the content caching replacement process as a Markov decision process (MDP) [15]. The state space, action space and reward function are described as follows:

(1) **State space $s(t)$**: The state space $s(t)$ is the index set of the cached content in F-AP n [9], i.e.,

$$s\left(t\right)=\left[i_1,i_2,...,i_c,...,i_C\right],\forall i_c\in\left[1,F\right]. \tag{11}$$

The index in the state space will increase rapidly as the content number of the content library increases, which is the curse of dimensionality. We sort the content indexes in the state space in descending order according to the request frequency, which can reduce the probability of low-frequency content indexes appearing in the state space.

(2) **Action space $a(t)$**: $a(t)\in\{0,1\}$ indicates whether the cached content in F-AP n needs to be replaced or not. Contents in the content library that are not cached in F-AP form a collection $\mathcal{K}$. If $a(t)=1$, k $(k<C)$ contents will be randomly selected from $\mathcal{K}$ and exchanged with the last contents stored in F-AP n. After sorting the content index based on the request frequency, $s(t+1)$ is obtained. It can ensure that the replaced contents are the least popular contents during each time slot in the current storage of F-AP n. $a(t)=0$ indicates the cached content in F-AP n does not need to be replaced.

(3) **Reward function $r(t)$**: when the caching state of F-AP n is $s(t)$, the reward $r(t)$ will be obtained after performing the action $a(t)$. Based on (6), in order to minimize the total content request delay and maximize the system reward, we design the reward function as follows [13]:

$$r\left(t\right)=\begin{cases}p_{nf}e^{-\lambda_1 d_{n1}},\ \text{Local Service}\\ p_{nf}e^{-(\lambda_1 d_{n1}+\lambda_2 d_{n2})},\ \text{F-APs Cooperation}\\ p_{nf}e^{-(\lambda_1 d_{n1}+\lambda_3 d_{n3})},\ \text{Cloud Service},\end{cases} \tag{12}$$

where $\lambda_1+\lambda_2+\lambda_3=1$, $\lambda_1<\lambda_2\ll\lambda_3$, $p_{nf}e^{-\lambda_1 d_{n1}}$ is the reward that the user obtains the requested content f from the local F-AP. $p_{nf}e^{-(\lambda_1 d_{n1}+\lambda_2 d_{n2})}$ indicates the user is served by the neighbouring F-APs. If user fetches content f from the cloud server, the reward is $p_{nf}e^{-(\lambda_1 d_{n1}+\lambda_3 d_{n3})}$.

Based on (8), the neural network internally decomposes the action-state value function Q into the state value function V and the action advantage function A in the dueling deep Q-network. The state value function V is not related to the action. The action advantage function A is related to the action and It is the average return of performing action $a(t)$ under the state $s(t)$ to solve the reward bias problem. The action Q-value function is given by

$$Q\left(s\left(t\right),a\left(t\right);\theta\right)=V\left(s\left(t\right);\theta\right)+A\left(s\left(t\right),a\left(t\right);\theta\right). \tag{13}$$

In practice, the action advantage is generally set as a single action advantage function minus the average value of all action advantage functions in a certain state. Based on this, the action Q-value function is calculated as follows:

$$Q\left(s\left(t\right),a\left(t\right);\theta\right)=V\left(s\left(t\right);\theta\right)+$$
$$\left(A\left(s\left(t\right),a\left(t\right);\theta\right)-\frac{1}{|A|}\sum_{a(t+1)}A\left(s\left(t\right),a\left(t+1\right);\theta\right)\right). \tag{14}$$

The (14) can ensure that the relative order of the dominant functions of each action is unchanged in this state. The advantage of the method is that it reduces the Q-value range and removes excess degrees of freedom. The stability of the algorithm is improved. The dueling deep Q-network based content caching replacement process is shown in Algorithm 1. Each F-AP optimizes the caching strategy π^* by maximizing the reward function $r(t)$ based on local content popularity and content request information of users.

C. HFL-Based Algorithm Workflow

In order to improve the caching cooperation between F-APs and protect user data privacy, we apply horizontal federated learning in cooperative edge caching. Each F-AP uploads the local model learned by the dueling deep Q-network to the cloud server. The cloud server aggregates all local models into a global model through federated averaging. Then, the cloud server sends the updated global model to each F-AP, and each F-AP trains its local model according to the updated global model. The HFL-based content caching policy is shown in algorithm 2. For F-AP n, the learning problem is to find the objective parameters $\hat{\theta}_n^t$ with the loss function $f\left(\theta_n^t\right)$ according to (9), i.e.,

$$\hat{\theta}_n^t=\arg\min_{\theta_n^t}f\left(\theta_n^t\right). \tag{15}$$

The global model is obtained by

$$f\left(\theta_G^t\right)=\frac{1}{\sum_{n\in\mathcal{N}}\mathcal{D}_n}\sum_{n=1}^{N}\mathcal{D}_n f\left(\theta_n^t\right), \tag{16}$$

where $\mathcal{D}_n$ is the local dataset of F-AP n. The learning problem in (15) can be solved by the gradient descent algorithm [16]. For $t=1,2,...,T$, F-AP n updates its local parameter θ_n^t as follows:

$$\theta_n^{t+1}=\theta_n^t-\eta\nabla f\left(\theta_n^t\right) \tag{17}$$

where $\eta\geq 0$ is the learning step. After T iterations, the global parameter θ_G^{t+1} is updated in the cloud server:

$$\theta_G^{t+1}=\frac{1}{\sum_{n\in\mathcal{N}}D_n}\sum_{n}^{N}D_n\theta_n^{t+1} \tag{18}$$

The updated global model parameter θ_G^{t+1} will be sent to the each F-AP for the next round training.

IV. Simulation Results

In this section, the performance of the proposed FDRL-based cooperative caching policy is evaluated. We consider the

Algorithm 1 The dueling deep Q-network based content caching replacement algorithm

Initialize reply memory $\mathcal{D}$, the maximum value B of experience samples;
Initialize the prediction network Q with the weight parameters θ, the target network $\hat{Q}$ with the weight parameters $\theta^{-}=\theta$;
for time slot $t=1,2,...,T_s$ **do**
  for user $u_n=1,2,...,U_n$ **do**
    user u_n requests content f;
    if content f has been cached in F-AP n or the neighbouring F-APs **then**
      Fetch the content from the F-AP n or the neighbouring F-APs;
    else
      if the storage of F-AP n is not full **then**
        Fetch content f from the cloud server and cache content f in F-AP n;
      else
        Observe the caching state $s(t)$;
        According to the ε-greedy method, choose an action $a(t)=\arg\max_{a(t)} Q\left(s(t),a(t)\right)$;
        Execute action $a(t)$, get new state $s(t+1)$;
        Obtain reward $r(t)$ according to (12);
        Save $(s(t),a(t),r(t),s(t+1))$ in $\mathcal{D}$;
        Set $s(t)=s(t+1)$;
        Randomly sample a minibatch of experiences from $\mathcal{D}$;
        Update θ_n of the prediction network by using the gradient descent method in (9);
        Reset $\theta_n^{-}=\theta_n$ every M steps;
      end if
    end if
  end for
end for

Algorithm 2 The HFL-based cooperative caching algorithm

Initialize parameter θ_n^t from Algorithm 1; number of iterations T; step size η;
for time slot $t=1,2,...,T$ **do**
  for each F-AP n **do**
    Compute its local update;
    $\theta_n^{t+1}=\theta_n^t-\eta\nabla f\left(\theta_n^t\right)$;
    Return θ_n^{t+1};
  end for
  Send θ_n^{t+1} to the cloud server;
  The cloud receive θ_n^{t+1} from each F-AP n;
  Update global model according to (18);
  Set $\theta_n^{t+1}=\theta_G^{t+1}$;
  The cloud server disperses θ_n^{t+1} to each F-AP;
end for

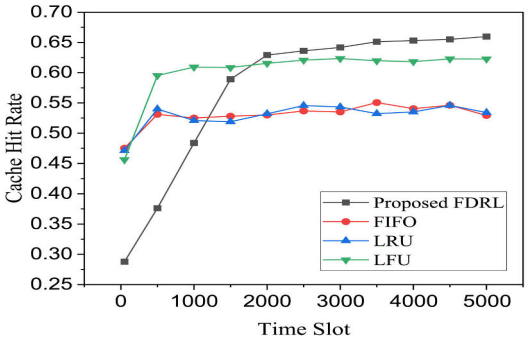

Fig. 2: Cache hit rate versus time slot for the proposed policy and three benchmark algorithms

size of the content library $F=500$ and the storage capacity of each F-AP $C=50$. The popularity of the content in the caching scenario is generated by M-Zipf distribution, which also represents the content request probability of the user. In time slot t, the parameter α_t of M-Zipf distribution are randomly generated, ranging from 0.5 to 1.5. In addition, d_{n1}, d_{n2}, d_{n3} caused by three different transmission methods are set to 10ms, 20ms, and 200ms, respectively. We compare and analyze the proposed FDRL-based cooperative caching policy with three traditional caching algorithms, including First In First Out (FIFO), the Least Recently Used (LRU) and the Least Frequently Used (LFU). Meanwhile, we use two reinforcement learning algorithms (Q-learning and DQN) as the benchmark caching algorithms.

Fig. 2 shows the performance comparison between the proposed cooperative caching policy and three traditional caching algorithms (FIFO, LRU, LFU) in improving the cache hit rate of content. We can observe that the cache hit rate of our proposed policy is lower than that of three traditional caching algorithms when time slot $t<2000$. When time slot $t>2000$, our proposed policy has a higher cache hit rate. The reason is that our proposed policy has few samples that can be learned in the reply memory $\mathcal{D}$ at the beginning of the algorithm iteration. With the continuous accumulation of learning samples, learning efficiency of the algorithm has been significantly improved. When cache size of each F-AP is set as $C=10\%F=50$, cache hit rate of our proposed policy can reach 0.65. As the number of iteration increases, the performance of our proposed policy in improving cache hit rate of the content is better than that of three traditional caching algorithms.

Fig. 3 shows the variation of the cache hit rate between the proposed cooperative caching policy and three traditional caching algorithms (FIFO, LRU, LFU) with different cache sizes. The cache size of each F-AP increases from $C=25$ to $C=250$ with $F=500$. It can be observed from Fig. 3 that the cache hit rate of our proposed cooperative caching policy gradually increases with the cache size. Meanwhile, it can also be observed that the cache hit rate of our proposed policy is always higher than that of three traditional algorithms.

Fig. 4 shows the performance comparison of the proposed cooperative caching policy and two reinforcement learning algorithms (Q-learning, DQN) in reducing the total request delay of users. We set the reward discount factor γ=0.9 and the learning step length α=0.001. It can be seen from Fig. 4 that our proposed policy has the lower request delay and

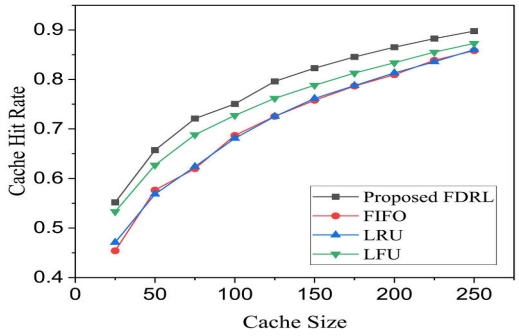

Fig. 3: Cache hit rate versus cache size for the proposed policy and three benchmark algorithms

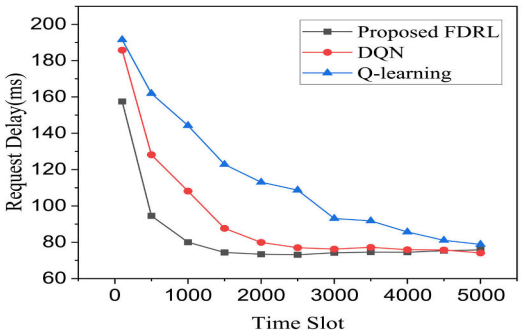

Fig. 4: Request delay versus time slot for the proposed policy and two benchmark algorithms

the faster convergence speed. In addition, we observe that the convergence speed of the Q-learning-based algorithm is significantly slower than the two algorithms with deep neural network (DQN, FDRL). The reason is that DRL uses deep neural network $Q\left(s\left(t\right),a\left(t\right),\theta\right)$ to approximate the action value function $Q^{*}(s(t),a(t))$, which greatly reduces the time of finding the optimal value.

V. Conclusions

In this paper, we have proposed an FDRL-based cooperative edge caching policy based on content popularity in F-RANs. In each F-AP, the dueling deep Q-network allows to learn a local caching model for optimal caching decision according to the user content request and the content popularity. HFL can enhance the caching cooperation between F-APs and improve the caching performance of the whole network by aggregating all local caching models from each F-AP into a global caching model in the cloud server. Simulation results have shown that our proposed cooperative caching policy outperforms traditional caching methods in reducing content request delay and improving the cache hit rate.

Acknowledgment

This work was supported in part by the Natural Science Foundation of China under grant 61971129, the Natural Science Foundation of Jiangsu Province under grant BK20181264, the Shenzhen Science and Technology Program under Grant KQTD20190929172545139 and JCYJ20180306171815699, and the National Major Research and Development Program of China under Grant 2020YFB1805005.

References

[1] C. Wang, Y. He, F. R. Yu, and et al., "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 1, pp. 7–38, 2018.

[2] Z. Piao, M. Peng, Y. Liu, and M. Daneshmand, "Recent advances of edge cache in radio access networks for internet of things: Techniques, performances, and challenges," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1010–1028, 2019.

[3] Y. Jiang, M. Ma, M. Bennis, and et al., "User preference learning-based edge caching for fog radio access network," *IEEE Trans. Wirel. Commun.*, vol. 67, no. 2, pp. 1268–1283, 2019.

[4] Y. Jiang, H. Feng, F. C. Zheng, and et al., "Deep learning-based edge caching in fog radio access networks," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 12, pp. 8442–8454, 2020.

[5] C. Xia, Y. Jiang, M. Peng, and et al., "Cooperative edge caching in fog radio access networks: A pigeon inspired optimization approach," in *IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[6] L. Li, Y. Xu, J. Yin, and et al., "Deep reinforcement learning approaches for content caching in cache-enabled D2D networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 544–557, 2020.

[7] X. Fang, T. Zhang, Y. Liu, and Z. Zeng, "Multi-agent cooperative alternating Q-learning caching in D2D-enabled cellular networks," in *IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[8] L. Lu, Y. Jiang, M. Bennis, and et al., "Distributed edge caching via reinforcement learning in fog radio access networks," in *IEEE Veh. Technol. Conf.*, 2019, pp. 1–6.

[9] J. Yan, Y. Jiang, F. Zheng, and et al., "Distributed edge caching with content recommendation in fog-RANs via deep reinforcement learning," in *IEEE Int. Conf. Commun. Workshops*, 2020, pp. 1–6.

[10] B. Guo, X. Zhang, Q. Sheng, and H. Yang, "Dueling deep-Q-network based delay-aware cache update policy for mobile users in fog radio access networks," *IEEE Access*, vol. 8, pp. 7131–7141, 2020.

[11] Z. Yang, Y. Liu, Y. Chen, and L. Jiao, "Learning automata based Q-learning for content placement in cooperative caching," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3667–3680, 2020.

[12] M. Dai, Z. Su, Q. Xu, and W. Chen, "A Q-learning based scheme to securely cache content in edge-enabled heterogeneous networks," *IEEE Access*, vol. 7, pp. 163 898–163 911, 2019.

[13] X. Wang, C. Wang, X. Li, and et al., "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, 2020.

[14] P. Kairouz, H. B. McMahan, B. Avent, and et al., "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[15] H. Zhu, Y. Cao, X. Wei, and et al., "Caching transient data for internet of things: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2074 – 2083, 2019.

[16] W. Y. B. Lim, N. C. Luong, D. T. Hoang, and et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 2031–2063, 2020.