Attention-Weighted Federated Deep Reinforcement Learning for Device-to-Device Assisted Heterogeneous Collaborative Edge Caching

Xiaofei Wang, *Senior Member, IEEE,* Ruibin Li, *Student Member, IEEE,* Chenyang Wang, *Student Member, IEEE,* Xiuhua Li, *Member, IEEE,* Tarik Taleb, *Senior Member, IEEE,* and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—In order to meet the growing demands for multimedia service access and release the pressure of the core network, edge caching and device-to-device (D2D) communication have been regarded as two promising techniques in next generation mobile networks and beyond. However, most existing related studies lack consideration of effective cooperation and adaptability to the dynamic network environments. In this paper, based on the flexible trilateral cooperation among user equipment, edge base stations and a cloud server, we propose a D2D-assisted heterogeneous collaborative edge caching framework by jointly optimizing the node selection and cache replacement in mobile networks. We formulate the joint optimization problem as a Markov decision process, and use a deep Q-learning network to solve the long-term mixed integer linear programming problem. We further design an attention-weighted federated deep reinforcement learning (AWFDRL) model that uses federated learning to improve the training efficiency of the Q-learning network by considering the limited computing and storage capacity, and incorporates an attention mechanism to optimize the aggregation weights to avoid the imbalance of local model quality. We prove the convergence of the corresponding algorithm, and present simulation results to show the effectiveness of the proposed AWFDRL framework in reducing average delay of content access, improving hit rate and offloading traffic.**

***Index Terms*—Edge caching, device to device, attention-weighted federated learning, deep reinforcement learning.**

Part of this work was presented at the IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, Apr. 6-8, 2020, which is cited as [1]. This work is supported in part by the National Key R & D Program of China through Grants No. 2018YFC0809803, 2019YFB2101901 and 2018YFF0214700, China NSFC GD Joint Fund U1701263, National NSFC through Grants No. 62072332, 61902044, 62002260, 61672117 and 62072060, Chongqing Research Program of Basic Research and Frontier Technology (Grant No. cstc2019jcyj-msxmX0589), Fundamental Research Funds for the Central Universities (Grant No. 2020CDJQY-A022), the European Union's Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant No. 871780, the Academy of Finland 6Genesis project under Grant No. 318927, the Academy of Finland CSN project under Grant No. 311654, Chinese National Engineering Laboratory for Big Data System Computing Technology at Shenzhen University, and Canadian Natural Sciences and Engineering Research Council. *(Corresponding author: Xiuhua Li.)*

X. Wang, R. Li and C. Wang are with the College of Intelligence and Computing, Tianjin University, Tianjin, 300072 China (e-mail: {xiaofeiwang, leeruibin, chenyangwang}@tju.edu.cn).

X. Li is with the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China, and with the School of Big Data & Software Engineering, Chongqing University, Chongqing, 401331 China (e-mail: lixiuhua1988@gmail.com).

T. Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland, Information Technology and Electrical Engineering, Oulu University, 90570 Oulu, Finland, and the Department of Computer and Information Security, Sejong University, Seoul, 05006 South Korea (e-mail: Tarik.Taleb@aalto.fi).

V. C. M. Leung is with the College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, 518060 China, and the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, V6T 1Z4 Canada (e-mail: vleung@ieee.org).

I. Introduction

Currently, the wide utilization of Internet of Things (IoT) has brought rocket-increasing service requirements for existing mobile networks (i.e., 4G), which results in the rapid growth of mobile data traffic [1], [2]. Advances in sensing and artificial intelligence (AI) techniques have enabled innovative intelligent applications for improving people's daily life [3]. However, these applications are highly dependent on the computation, storage and communication resources [4]. The low latency for content access and diverse application requirements may not be satisfied if the contents are fetched from remote data centers (e.g., cloud server). To address these issues, it is necessary to introduce advanced networking architectures and new data transmission techniques towards next generation mobile networks and beyond (i.e., 5G/B5G). Particularly, mobile edge caching (MEC) has been regarded as a promising technique to relieve the burden of backhaul traffic for network operators [5]. In the MEC system, popular contents can be cached in proximity to the edges of networks, e.g. base stations (BSs) and user equipment (UE) (or mobile devices)[1], which reduces massive duplicated traffic of content deliveries via backhaul networks and shortens the transmission delay. Meanwhile, by combining device to device (D2D) communications [6], the network performances on traffic offloading and delay reduction can be further improved.

In terms of MEC, two key issues, i.e., node selection and cache replacement, need to be properly investigated. Particularly, when UEs generate content requests, we should first decide which nodes (i.e., UEs, BSs or the cloud server) are responsible for their requests and how to fetch the contents [7]. Then we should consider to design a proper cache replacement framework to adapt to the dynamic network environments [8]. Most existing caching systems rely on such rule-based cache

[1]In this paper, we use the terms mobile device and user equipment interchangeably.

replacement schemes as First In First Out (FIFO) [9], Least Recently Used (LRU), Least Frequently Used (LFU), or their variants [10]–[12]. Meanwhile, to tackle the cache replacement problem, some learning techniques have been employed such as long short-term memory (LSTM) [13] and reinforcement learning methods [14] like deep Q-learning network (DQN) [15] [16]. However, most of them are in lack of effective cooperation and adaptability to the dynamic network environments. Moreover, some existing learning-based methods require users to send their personal data to the central server, which may cause serious issues of user privacy and security. Indeed, even transmitting the anonymous data can still put user privacy at risk, considering that the attackers can recover the anonymous information by comparing it with other data [17]. Federated learning (FL) is a promising technique to address this issue[2].

Thus, in this paper, we are motivated to exploit the framework design of heterogeneous collaborative edge caching by jointly optimizing the node selection and cache replacement in D2D assisted mobile networks, and consider the flexible trilateral collaboration among UEs, BSs and the cloud server. We formulate the joint optimization problem as a Markov decision process (MDP), and propose an attention-weighted federated deep reinforcement learning (AWFDRL) framework to address the problem. Particularly, it uses DQN to address the formulated long-term mixed integer linear programming (LT-MILP) problem [19], and employs FL to improve the training process by considering the limited computing capacity as well as the user privacy. Most importantly, we employ an attention mechanism [20] to control the model weights in the FL aggregation step, which can address the imbalance issue of local model quality. It distributes different aggregation weights to different quality models. There are many factors that can affect the attention weights and model quality. We divide them into three categories as: 1) user-related factors such as user preference and personal habit; 2) device-related factors such as CPU capacity and RAM size which may effect the training batch size and replay memory size; 3) model-related factors such as model staleness and performance loss which can be calculated during the training process. The main contributions of this paper can be summarized as:

- We investigate the issue of D2D assisted heterogeneous collaborative edge caching in mobile networks. Particularly, we model the whole edge caching process by formulating an LT-MILP problem, and use the DQN model to control the decision process of joint node selection and cache replacement dynamically based on the network state and historical information.
- We propose the AWDFRL framework, an improved FL framework which can train the DQN model in a distributed manner through keeping the data in the local UEs and address the issue of model aggregation among heterogeneous UEs. Most importantly, we employ an attention mechanism to control the model weights in the FL aggregation step, which can solve the imbalance problem of local model quality. In addition, we derive the expectation convergence of AWFDRL.
- Simulation results show that compared with existing methods, the proposed AWFDRL framework can effectively reduce average delay, improve hit rate, and offload traffic. Moreover, it also outperforms the existing FL framework in term of average reward.

The remainder of this paper is organized as follows. Sec. II summarizes the related work. We introduce the system model in Sec. III, and formulate the problem in Sec. IV. We propose the AWFDRL framework and provide the convergence analysis in Sec. V. Simulation results are provided in Sec. VI. Finally, Sec. VII concludes this paper.

II. Related Work

FL is a promising method to train the neural network parameters while keeping the training data in the local devices [21]. There are several studies focusing on FL for edge computing. For instance, in [22], FL and DQN were used to address the issue of the computation task offloading. The study in [23] used a FL model to optimize the content caching problem. In [24], the FL model and DQN model were combined to solve the problem of joint computing, caching and communication in Edge-AI. However, there still exists an important issue in these models, i.e., aggregation efficiency. Particularly, these models only considered simple average aggregation of local model parameters, which was not effective since different devices might have different model performances. Besides, since some local devices might train an abnormal model, giving these models with the same aggregation weight would cause serious bias to the global model.

Moreover, there are also some studies about asynchronous FL in recent years. In [25], the authors generalized a normal FL method by allowing the episode to vary according to the characteristics of the network, which can improve the robustness of the FL. Also, considering the user-related factors such as user privacy, transmission delay and network status, the studies in [26], [27] adopted a differentially private asynchronous FL scheme to address the security issue. In [28], [29], model-related factors like staleness were considered to optimize the system design and training process of FL. The study in [30] showed that the parameters in the shallow and deep layers in the models could be updated asynchronously in order to decrease the amount of data to be transmitted between the server and clients. The blockchain technique was considered in [31] to improve the secure data sharing in FL. Meanwhile, the study in [32] proposed the asynchronous FL with dual-weight correction. Particularly, the dual weights were calculated based on the device-related factors like data size and model-related factors like parameter weights, and using this model could better adjust to the unrestricted characteristics of edge nodes.

[2]Privacy is a potential advantage of FL as it transmits the local model information (like model parameters) rather than the source data. These parameters do not contain privacy information of the raw training data, and the updates of the global model can be executed without transmitting the metadata over the complex network [18]. However, privacy is not the focus of this paper.

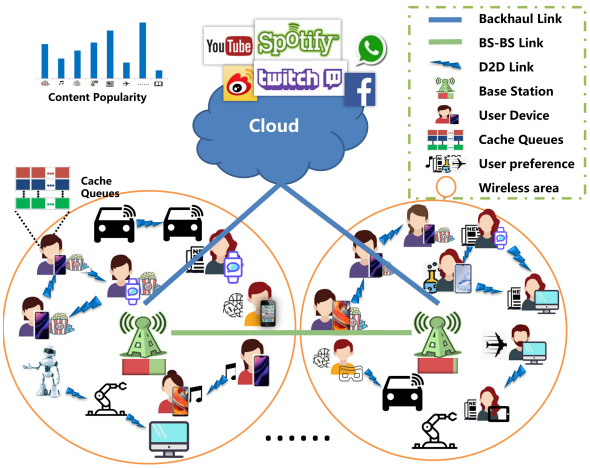


Fig. 1. Illustration of the network architecture of D2D assisted heterogeneous collaborative edge caching.

III. System Model

In this section, we introduce the system modeling of D2D assisted heterogeneous collaborative edge caching. Particularly, we introduce the corresponding network architecture in Sec. III-A. Sec. III-B models the content popularity and user preferences. Sec. III-C and Sec. III-D introduce the D2D sharing pattern, and the content transmission and delay model, respectively. Some key modeling parameters and notations are summarized in Table I.

A. Network Architecture

An illustration of the network architecture of D2D assisted heterogeneous collaborative edge caching is shown in Fig. 1, which includes three types of heterogeneous cache nodes, i.e., the cloud server, BSs and UEs. In the considered network, there are $\mathcal{F} = \{1, 2, ..., \mathfrak{F}\}$ popular contents in total that users may access, and their sizes are denoted by $(s_f)^{1\times\mathfrak{F}}$. Let $\mathcal{U} = \{1, 2, ..., U\}$ denote the index set of UEs, and each UE is equipped with a cache of limited storage c_u. To simplify the model, we consider the cooperation case of two neighboring BSs in the network: the local BS (denoted by BS_s) serving the UEs, and a neighboring BS (denoted by BS_n) which is close to and connected with BS_s. Particularly, both BSs are equipped with a cache of limited storage c_b, and BS_n can deliver the cached contents to BS_s through optical cables. We assume that the cache size of the cloud server is sufficiently large so that it can cache all the contents in the $\mathcal{F}$. The cloud server connects with all the BSs via the backhaul links. For each UE, we denote $x_{u,f} = 1$ if UE u has content f in its cache list; otherwise, $x_{u,f} = 0$. If content f is cached in BS_s or BS_n, we denote $x_{s,f} = 1$ or $x_{n,f} = 1$, respectively; otherwise, we denote $x_{s,f} = 0$ or $x_{n,f} = 0$. Denote $\mathbf{X}_u, \mathbf{X}_s, \mathbf{X}_n$ as the cache states of UE u, BS_s and BS_n, respectively.

TABLE I
Modeling Parameters and Notations.

Notation	Definition
$\mathfrak{F}, \mathcal{F}$	Total number and set of contents
$U, \mathcal{U}$	Number and set of UEs
s_f	Size of content f
BS_s	Local BS
BS_n	Neighboring BS
c_u	Cache size of UEs
c_b	Cache size of BSs
$\mathbf{X}_u, \mathbf{X}_s, \mathbf{X}_n$	Cache state
$\mathbf{\Omega}$	Content popularity
$p_u^f, \vec{\mathbf{p}}$	Preference of user u for content f
$\boldsymbol{R}_u$	D2D communication rate of user u
$\mathbf{Req}_t$	Request state as time t
$r_{u,t}^D, r_{u,t}^B, r_{Co}, r_C$	Data rate between different link
$d_{u,t}^L, d_{u,t}^D, d_{u,t}^B, d_{u,t}^{Co}, d_{u,t}^C$	Transmission delay
B^D, B^B	Bandwidth between UE and nodes
L, μ, σ_u, G	Variables introduced by assumptions
η_t	Learning rate
Γ	Term to quantify the aggregation operation
F, F_u	Loss function of global model and local model
w_u	Attention weight during aggregation phase
$\mathbf{K}_u$	Evaluation indicators
$\widetilde{\mathcal{M}}_u, m_u$	Mini-batch from replay memory buffer and its size
$\mathcal{M}_u, M_u$	Replay memory buffer of u and its size
$\boldsymbol{\theta}_t^u, \boldsymbol{\Theta_t}$	Local model parameter and global model parameter
$\mathbf{s}_t^u$	The state of UE u at time t
$\mathbf{a}_t^u$	The action of UE u at time t
π	Cache policy
$\boldsymbol{\beta}_{u,t}^f$	Indicator on that UE u fetches content f from node v
g_t^D, g_t^B	Transmit power between UE and other UE, BS

B. Content Popularity and User Preference Model

1) Content Popularity: Generally, the content popularity is modeled as the MZipf distribution [33]

$$\omega_f = \frac{(O_f + \tau)^{-\beta}}{\sum_{i\in\mathcal{F}}(O_i + \tau)^{-\beta}}, \ \forall f \in \mathcal{F}, \tag{1}$$

where O_f, $\tau \geq 0$, and β denote the rank of content f in the descending order of content popularity, plateau factor, and skewness factor, respectively. Denote $\mathbf{\Omega} = (\omega_f)^{1\times\mathfrak{F}}$. Besides, we assume that the content popularity changes slowly during a relatively long period.

2) User Preference: Denote p_u^f as the probability that UE u prefers to access content f, which can be expressed as

$$p_u^f = \frac{R_{u,f}}{R_u}, \ \forall u \in \mathcal{U}, \forall f \in \mathcal{F}, \tag{2}$$

where $R_{u,f}$ denotes the number of requests for content f from UE u, and R_u denotes the total number of content requests from UE u during the considered time period. Particularly, $\sum_{f\in\mathcal{F}} p_u^f = 1$ for $\forall u \in \mathcal{U}$ holds.

C. D2D Sharing Pattern

We model the D2D sharing pattern mainly based on the physical domain and social domain as shown in Fig. 2. The details of these domains are discussed as below.

1) Physical Domain: Due to the physical constraints such as signal attenuation [34], only a subset of UEs that are sufficiently close (e.g., with the detectable signal strength) can be feasible relay candidates for UE u in order to establish

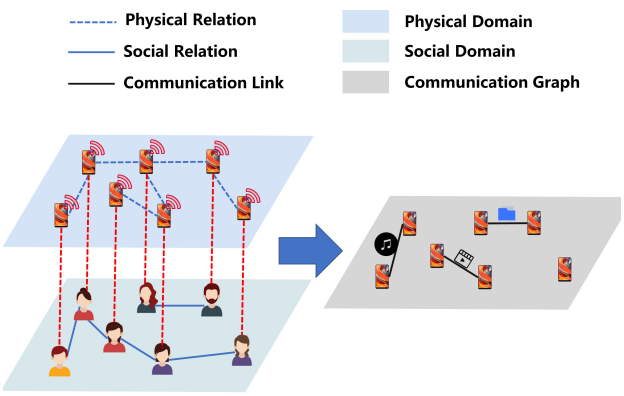


Fig. 2. Illustration of communication graph based on social domain and physical domain.

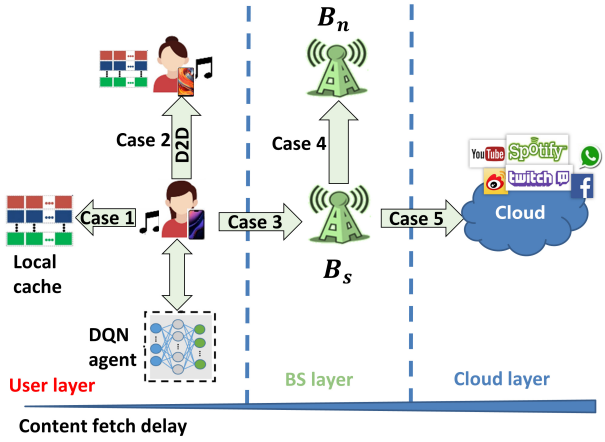


Fig. 3. Illustration of content request path through different cache nodes.

the corresponding D2D links. Thus, we introduce the physical graph $\mathcal{G}_P \triangleq \{\mathcal{U}, \mathcal{E}_P\}$, where $\mathcal{U}$ is the vertex set of all UEs and $\mathcal{E}_P \triangleq \{(u,v) : e^p_{uv} = 1, \forall u, v \in \mathcal{U}\}$ is the edge set. Here, $e^p_{uv} = 1$ holds if and only if UE v is a feasible relay for UE u; otherwise, we set $e^p_{uv} = 0$.

2) Social Domain: Intuitively, considering the selfish nature of human beings, mobile users with stronger social relationship are more willing to share their own content directly. This phenomenon is called social trust [35]. Based on this, we introduce the social graph $\mathcal{G}_S$ which also contains the vertex set $\mathcal{U}$ and edge set $\mathcal{E}_S$. Here, the edge set is given by $\mathcal{E}_S \triangleq \{(u,v) : e^s_{uv} = 1, \forall u, v \in \mathcal{U}\}$. When two users have close social relationship such as friends and family members, we set $e^s_{uv} = 1$; otherwise, we set $e^s_{uv} = 0$. We can get this relationship from the social network, and assume that UE u can communicate with UE v only when $e^s_{uv} = 1$.

3) Communication Graph: Based on the physical graph and the social graph, we can get the communication graph $\mathcal{G}_C \triangleq \{\mathcal{U}, \mathcal{E}_C\}$, where $\mathcal{E}_C \triangleq \{(u,v) : e^c_{uv} = 1, \forall u, v \in \mathcal{U}\}$ is the edge set and we calculate $e^c_{uv} = e^p_{uv} \cdot e^s_{uv}$. Particularly, $e^c_{uv} = 1$ holds if and only if there is a path (the length of which is less than a threshold l) between UE u and and UE v in the physical graph as well as the social graph.

4) D2D Sharing Probability: We use tanimoto coefficients [36] among different users to measure the D2D sharing probability. Based on the user preference, we can calculate the tanimoto coefficient between UE u and UE v as

$$r^{\tan}_{uv} = \frac{\vec{\mathbf{p}}_u \cdot \vec{\mathbf{p}}_v}{||\vec{\mathbf{p}}_u||^2 + ||\vec{\mathbf{p}}_v||^2 - \vec{\mathbf{p}}_u \cdot \vec{\mathbf{p}}_v}, \forall u, v \in \mathcal{U}, \tag{3}$$

where $\vec{\mathbf{p}}_u = (p^f_u)^{1\times\mathfrak{F}}, \forall u \in \mathcal{U}$. We define $r^{\tan}_{uu} = 0$ for the pair (u,u). After getting the $r^{\tan}_{uv}$, we calculate the D2D sharing probability between u and v as $R_{uv} = r^{\tan}_{uv} e^c_{uv}$, and denote $\boldsymbol{R}_u = (R_{uv})^{1\times U}$. Finally, we normalize $\boldsymbol{R}_u$ by setting $\frac{R_{uv}}{\sum_{v\in\mathcal{U}} R_{uv}} \rightarrow R_{uv}, u \in \mathcal{U}, \forall v \in \mathcal{U}$.

D. Content Transmission and Delay Model

There are several ways to fetch contents as shown in Fig. 3. The concept of time slots is used to divide the considered time period into T episodes of δ (in seconds), denoted by $\mathcal{T} = \{1, 2, \ldots, T\}$. For each time slot $t \in \mathcal{T}$, UE u will send a request $r_{u,t}$ to access a content (say content $f_{u,t}$), and we denote $\mathbf{Req}_t = \{r_{1,t}, r_{2,t}, ..., r_{U,t}\}$ to describe the request state at time t. Each way to fetch contents is shown as follows:

- Case 1: If the content requested by UE u is cached in its cache list, then the content can be satisfied locally. As a result, the corresponding delay of fetching the content, denoted by $d^L_{u,t}$, can be regarded as zero.
- Case 2: If the request cannot be satisfied through the local cache list, UE u can seek help from surrounding users that can establish direct D2D links. According to [6], the corresponding transmission rate can be calculated as $r^D_{u,t} = B^D \log(1 + g^D_t |h^D_{u,t}|^2)$, where B^D, g^D_t, and $h^D_{u,t}$ denote the corresponding channel bandwidth, transmit power, and a ratio function of the channel gain to background noise power σ_b, respectively. Thus, the D2D transmission delay for UE u to fetch the content through D2D communication at time slot t can be calculated as $d^D_{u,t} = \frac{s_{f_{u,t}}}{r^D_{u,t}}$, where $s_{f_{u,t}}$ is the size of content $f_{u,t}$.
- Case 3: UE u can also send the request to the local BS (i.e., BS_s) for fetching the content. The corresponding data rate can be calculated as $r^B_{u,t} = B^B \log(1 + g^B_t |h^B_{u,t}|^2)$, where B^B, g^B_t, and $h^B_{u,t}$ denote the corresponding channel bandwidth, transmit power, and a ratio function of the channel gain to background noise power σ_b, respectively. The corresponding delay of fetching the content can be calculated as $d^B_{u,t} = \frac{s_{f_{u,t}}}{r^B_{u,t}}$.
- Case 4: The neighboring BS_n is also a candidate node, and UE u can fetch the content from BS_n with some additional delay if the content is in the cache list of B_n. Suppose that the average data rate between the BSs is a constant, denoted by r_{Co}. The total delay of fetching the content through BS cooperation can be calculated $d^{Co}_{u,t} =$

$d^B_{u,t} + \frac{s_{f_{u,t}}}{r_{Co}}$.

- Case 5: Finally, if the request cannot be satisfied in the above ways, BS_s will forward the request to the cloud and fetch the content with a long delay. Suppose that the average data rate between BS_s and the cloud server is a constant, denoted by r_C. The total delay of fetching the content through the cloud can be calculated as $d^C_{u,t} = d^B_{u,t} + \frac{s_{f_{u,t}}}{r_C}$.

Note that $d^L_{u,t} < d^D_{u,t} < d^B_{u,t} < d^{Co}_{u,t} < d^C_{u,t}$ generally holds in practical systems [37].

IV. Problem Formulation

In this section, we formulate the joint problem of node selection and cache replacement in a UE as a Markov decision process (MDP). Particularly, the UE's state and action are introduced in Sec. IV-A and Sec. IV-B, respectively. Sec. IV-C calculates the system reward. Finally in Sec. IV-D, the objective of UEs is to find an optimal policy to optimize the expected long-term reward which is defined as a value function.

A. UE State

The state of UE u at time slot t can be described as $\mathbf{s}^u_t = [\mathbf{\Omega}_t, \vec{\mathbf{p}}_u, r^D_{u,t}, \mathbf{X}_{u,t}, \mathbf{I}_f]$, where $\mathbf{\Omega}_t$, $r^D_{u,t}$ and $\mathbf{X}_{u,t}$ denote the content popularity, D2D link rate and cache state at time slot t, respectively. Besides, $\mathbf{I}_f$ is the indicator on which node caches the requested content f.

B. UE Action

After receiving the state $\mathbf{s}^u_t$, UE u will decide where to fetch the content through the above methods and which content should be replaced in its cache list. The action of UE u at time slot t can be described as $\mathbf{a}^u_t = [\boldsymbol{\beta}^f_{u,v,t}, \mathbf{X}^f_{u,t}]$, where $\boldsymbol{\beta}^f_{u,v,t}$ is the indicator that whether UE u obtains content f from node v (e.g., other UEs, local or neighboring BSs, or the cloud server), and $\mathbf{X}^f_{u,t}$ denotes that whether content f is replaced from the cache list of UE u.

C. System Reward

Our objective is to maximize the D2D sharing traffic while minimizing the delay to fetch the content. Thus, two aspects are taken into account to formulate the system reward function, namely, the gain function of D2D sharing traffic and the cost function of content access delay.

1) D2D Sharing Gain: For each pair of UEs, if UE u does not have the content (i.e., $x_{u,f} = 0$) and UE v does (i.e., $x_{v,f} = 1$), UE u can get the requested content f from UE v with the probability $R_{u,v}$. Thus, the content sharing gain via D2D communication between UE u and UE v at time slot t can be calculated as $s_{f_{u,t}} R_{u,v}$. Then the total D2D sharing gain of UE u at time slot t can be calculated as

$$C^1_{u,t} = \sum_{v \in \mathcal{U}} s_{f_{u,t}} \, R_{u,v} \, x_{v,f}(1 - x_{u,f}), \forall u \in \mathcal{U}, \forall t \in \mathcal{T}. \quad (4)$$

2) Content Fetch Gain: Since UE u can only be shared with a content by another UE at one time slot, we introduce an average queue delay d^Q_u, which is directly proportional to the corresponding served UEs. Then we define the content fetch gain [37] as

$$C^2_{u,t} = \begin{cases} \psi e^{-d^L_{u,t}}, & Local\ Cache \\ \psi e^{-(d^Q_u + d^D_{u,t})}, & D2D\ Communication \\ \psi e^{-d^B_{u,t}}, & Communication\ to\ B_s \\ \psi e^{-d^{Co}_{u,t}}, & BS - BS\ Cooperation \\ \psi e^{-d^C_{u,t}}, & Cloud\ Service \end{cases}, \quad (5)$$

where ψ is an introduced parameter, and the negative exponential function with respect to the delay is adopted to realize the objective of minimizing the content fetch delay.

Thus, based on the above definitions, we calculate the system reward as

$$C(\mathbf{s}^u_t, \mathbf{a}^u_t) = \lambda_1 C^1_{u,t} + \lambda_2 C^2_{u,t}, \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \quad (6)$$

where $\lambda_1 + \lambda_2 = 1$, $0 \leq \lambda_1, \lambda_2 \leq 1$ are two introduced weight factors.

D. Value Function

We define the cache policy π as the mapping from the current state to a series of actions, e.g., $\pi(\mathbf{a}|\mathbf{s}^u_t)$ is the possibility of taking action $\mathbf{a}$ when the state is $\mathbf{s}^u_t$ under the policy π, and $\mathbf{a}^u_t = \max_{\mathbf{a} \in \mathbf{A}} \pi(\mathbf{a}|\mathbf{s}^u_t)$. The objective of UEs is to find an optimal policy to optimize the expected long-term reward (defined as a value function) [15], which can be expressed as

$$V(\mathbf{s}) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^{t-1} C(\mathbf{s}^u_t, \mathbf{a}^u_t) | \mathbf{s}^u_0 = \mathbf{s} \Big], \quad (7)$$

where $\gamma \in [0, 1]$ is a discounted factor. According to the Bellman Equation [15], the value function in (7) can also be expressed as

$$V(\mathbf{s}) = \sum_{\mathbf{a} \in \mathbf{A}} \pi(\mathbf{a}|\mathbf{s}) \Big\{ C(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}'} Pr\{\mathbf{s}'|(\mathbf{s}, \mathbf{a})\} \cdot V(\mathbf{s}') \Big\}, \quad (8)$$

where $\mathbf{a}$ is the action we take at state $\mathbf{s}$ and $\mathbf{s}'$ is the possible state after we execute action $\mathbf{a}$. Each UE is expected to learn an optimal cache policy $\pi^* \in \Pi$. The D2D assisted heterogeneous edge caching problem in terms of joint node selection and cache replacement can be formulated to maximize the value function as

$$\begin{aligned} &\max_{\pi \in \Pi} V(\mathbf{s}) \\ &s.t. \sum_{f \in \mathcal{F}} x_{u,f} \, s_f \leq c_u, \forall u \in \mathcal{U}, \\ &x_{u,f} \in \{0, 1\}, \forall u \in \mathcal{U}, \forall f \in \mathcal{F}. \end{aligned} \quad (9)$$

The above problem in (9) is a long-term mixed integer linear programming (LT-MILP), which has been proven to be NP-hard [38]. Particularly, if the numbers of UEs and contents are large-scale in the system, the dimension of the state space will be very high. To address this problem, traditional iterative approaches [19] generally have extremely high computation

complexity, which will be a significant disadvantage limiting their practical applications in caching systems. Therefore, it is necessary to introduce intelligent learning-based methods in the system.

V. Framework Design of AWFDRL

To address the above problem, we propose the AWFDRL framework including three main phases, i.e., model release phase, local DQN model training phase and weighted federated aggregation phase. Particularly, the details of the i-th round training and the whole flow in the proposed AWFDRL framework are introduced in Sec. V-A. Meanwhile, Sec. V-B and Sec. V-C describe the details of the local training and model aggregation process, respectively. Finally in Sec. V-D, we derive the expectation convergence of the AWFDRL.

A. Whole Process

The whole process of the i-th round training in the proposed AWFDRL framework is shown in Fig. 4.

1) Model release phase: In the model release phase, there contains two steps which is the left part of round i in Fig. 4. In the step 1, UEs associate with the local BS and report their state like whether they can participate in the training or not. After receiving these information, the local BS executes the model release step by sending the global model of the last round to local agents.

2) Local training phase: In the local DQN model training phase, each local agent receives the global model parameters $\mathbf{\Theta}_t$ from the local BS. Then each UE u starts to train its local DQN model $\boldsymbol{\theta}_t^u$ according to the global model parameters and its local data. The details of the training process are shown in the Sec. V-B.

3) Aggregation phase: There are two steps in this phase which is the right part of round i in Fig. 4. Firstly, after E local training rounds, each UE collects its training evaluation indicators (e.g., average reward, average loss, and hit rate) during the training phase, and sends them to the local BS with the local model parameter $\boldsymbol{\theta}_t^u$ which is step 4. Then, in step 5, the BS calculates each agent's aggregation weights based on the training evaluation indicators. Particularly, we employ the attention mechanism to provide different devices with different aggregation weights. Then we aggregate different local DQN parameters with the attention weights, instead of aggregating them equally or calculating weights simply based on the data size. The details of the aggregation process are discussed in the Sec. V-C.

B. Local DQN Model Training

The local DQN model is utilized to evaluate the policies based on the action-value function $Q(\mathbf{s}_t^u, \mathbf{a}_t^u)$. According to the relationship between the value function and action-value function, we have

$$Q(\mathbf{s}_t^u, \mathbf{a}_t^u) = C(\mathbf{s}_t^u, \mathbf{a}_t^u) + \gamma \sum_{\mathbf{s}_{t+1}^u \in \mathbf{S}} Pr\{\mathbf{s}_{t+1}^u | \mathbf{s}_t^u, \mathbf{a}_t^u\} V(\mathbf{s}_{t+1}^u). \tag{10}$$

Since $Q(\mathbf{s}_t^u, \mathbf{a}_t^u)$ cannot be directly obtained due to continuous action space, we apply deep neural network to approximate $Q(\mathbf{s}_t^u, \mathbf{a}_t^u)$ and update parameters by using the history experience stored in replay memory buffer $\mathcal{M}_u$. So we have $Q(\mathbf{s}_t^u, \mathbf{a}_t^u; \boldsymbol{\theta}_t^u) \approx Q(\mathbf{s}_t^u, \mathbf{a}_t^u)$. The iterative formula of Q function can be expressed as

$$\begin{aligned} Q(\mathbf{s}_t^u, \mathbf{a}_t^u; \boldsymbol{\theta}_t^u) \quad = \quad & Q(\mathbf{s}_t^u, \mathbf{a}_t^u; \boldsymbol{\theta}_t^u) + \alpha \Big(C(\mathbf{s}_t^u, \mathbf{a}_t^u) \\ & + \gamma \cdot \max_{\mathbf{a}_{t+1}^u} Q(\mathbf{s}_{t+1}^u, \mathbf{a}_{t+1}^u; \widehat{\boldsymbol{\theta}_t^u}) \\ & - Q(\mathbf{s}_t^u, \mathbf{a}_t^u; \boldsymbol{\theta}_t^u) \Big), \end{aligned} \tag{11}$$

where $\alpha \in [0, 1)$ is an introduced parameter, $\boldsymbol{\theta}_t^u$ is a parameter of evaluation network, and $\widehat{\boldsymbol{\theta}_t^u}$ is a parameter of target network. Two networks are utilized to reduce the relevance between action choosing and model training. We employ the gradient descent (GD) method for updating the parameters. The loss function of DQN model is expressed as

$$F_u(\boldsymbol{\theta}_t^u) = \sum_{(\mathbf{s}_i^u, \mathbf{a}_i^u) \in \widetilde{\mathcal{M}}_u} (y_i - Q(\mathbf{s}_i^u, \mathbf{a}_i^u; \boldsymbol{\theta}_t^u))^2, \tag{12}$$

where $y_i = C(\mathbf{s}_i^u, \mathbf{a}_i^u) + \gamma \max_{\mathbf{a}_{i+1}^u} Q(\mathbf{s}_{i+1}^u, \mathbf{a}_{i+1}^u; \widehat{\boldsymbol{\theta}_t^u})$, and $\widetilde{\mathcal{M}}_u$ is a mini-batch of $\mathcal{M}_u$. Then the update of parameter $\boldsymbol{\theta}_t^u$ can be expressed as

$$\boldsymbol{\theta}_{t+1}^u = \boldsymbol{\theta}_t^u - \eta_t \nabla(F_u(\boldsymbol{\theta}_t^u)), \tag{13}$$

where η_t is a learning rate.

The whole process of the local DQN training is shown in Algorithm 1 and Fig. 5. For a UE, it runs E episodes during the local DQN training phase. In each episode, UE sends a request $r_{u,t}$. If the requested content is in the local cache list, the request can be satisfied immediately, and this episode ends (Lines 1-4). Otherwise, the UE collects the current state $\mathbf{s}_t^u$ (Label 1 in Fig. 5) and chooses an action $\mathbf{a}_t^u$ based on the evaluation Q-Network (Labels 2-3). After executing the selected action, the DQN agent can obtain the immediate reward and new state (Lines 5-11). Then it constructs a transition $\mathbf{T}_t$ and stores it into the replay memory buffer (Lines 12-13, Label 4-5). Finally, the DQN agent samples a random mini-batch of transitions from $\mathcal{M}_u$, and uses it to update the evaluation Q-network. The target $\hat{Q}$ network will be updated by assigning $\boldsymbol{\theta}_t^u$ to $\widehat{\boldsymbol{\theta}_t^u}$ periodically (Lines 14-16, Labels 6-7). Besides, we calculate the size of replay memory of UE u as $M_u = |\mathcal{M}_u|$, and the mini-batch size of UE u as $m_u = |\widetilde{\mathcal{M}}_u|$, respectively. The computation complexity for UE u mainly includes two parts: collecting transitions and back propagation. For UE u, the complexity of collecting transitions is $O(M_u)$; and the complexity of training parameters with back propagation and gradient descent is $O(abE\, m_u)$, where a and b denote the layer number and the number of units in each layer, respectively. Therefore, we can calculate the total computation complexity for all UEs as $O\big(\sum_{u \in \mathcal{U}} (M_u + abE\, m_u)\big)$.

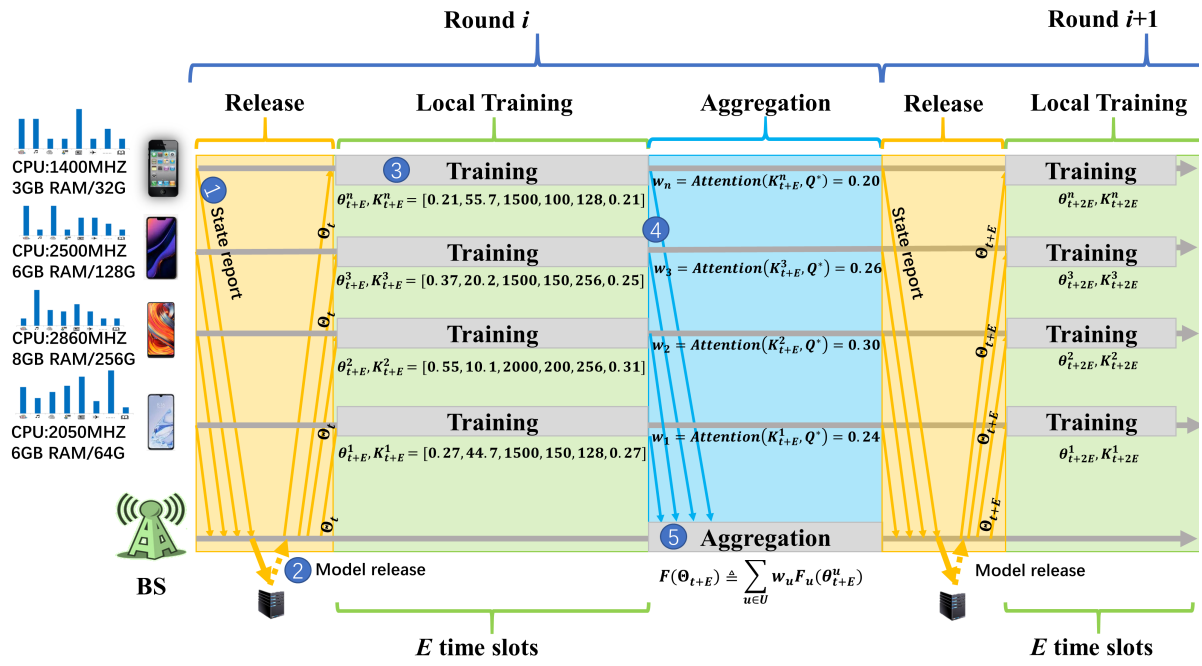


Fig. 4. The whole process of the t-th round training in the proposed AWFDRL framework.

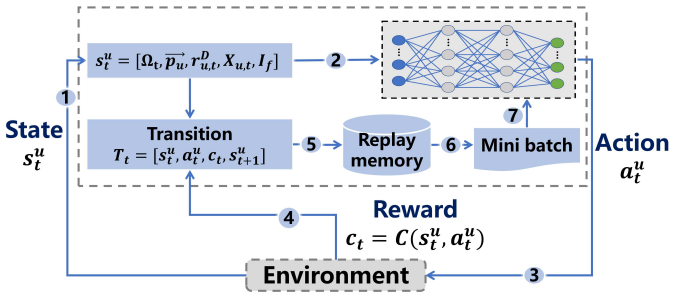


Fig. 5. Illustration of the local DQN Training Process.

C. Weighted Federated Aggregation

As a leading method in FL, Federated Averaging (FedAvg) [18] performs gradient descent algorithm in parallel on a small subset of the total devices, and then averages the sequences only once in a while. It employs simple weight average integration based on the data size of the distributed trained model parameters. However, by considering the difference among UEs' computing capacity, data quality as well as the model performance, it is not reasonable to integrate each local model equally and we need to consider its weighted form (i.e., weighted federated aggregation). In this model, we use the reward $C(\mathbf{s}_t^u, \mathbf{a}_t^u)$ and some device-related indicators as the measurement to evaluate the local model's contribution to the global model. The update process of weighted federated aggregation is shown in Fig. 6.

Algorithm 1 Local DQN Model Training Process.

Iteration: (Training Process)

for episode $t = 1$ **to** E **do**
  UE u send a request $r_{u,t}$ for content f.
  if The cache state $x_{u,f} = 1$ **then**
    End episode.
  else
    Receive the caching state $\mathbf{s}_t^u = [\mathbf{\Omega}_t, \vec{\mathbf{p}}_u, r_{u,t}^D, \mathbf{X}_{u,t}, \mathbf{I}_f]$.
    Select action $\mathbf{a}_t^u = \arg\max_{\mathbf{A}} Q(\mathbf{s}_t^u, \mathbf{a}_t^u; \boldsymbol{\theta}_t^u)$.
    Execute action $\mathbf{a}_t^u$.
    Obtain immediate reward $c_t = C(\mathbf{s}_t^u, \mathbf{a}_t^u)$.
    Observe the new state $\mathbf{s}_{t+1}^u$.
    Construct $\mathbf{T}_t = [\mathbf{s}_t^u, \mathbf{a}_t^u, c_t, \mathbf{s}_{t+1}^u]$.
    Store the transition $\mathbf{T}_t$ into $\mathcal{M}_u$.
    Randomly sample a mini-batch $\widetilde{\mathcal{M}}_u$ from $\mathcal{M}_u$.
    Update evaluation Q network $\boldsymbol{\theta}_t^u$ with $\nabla(F_u(\boldsymbol{\theta}_t^u))$.
    Update target $\hat{Q}$ network parameter $\widehat{\boldsymbol{\theta}}_t^u$ periodically.
  end if
end for

Problem formulation: The corresponding problem of weighted federated aggregation can be formulated as

$$\min_{\boldsymbol{\theta}, p} \left\{ F(\mathbf{\Theta}_t) \triangleq \sum_{u \in \mathcal{U}} w_u F_u(\boldsymbol{\theta}_t^u) \right\}, \tag{14}$$

where w_u is an introduced weight factor for UE u to measure the contribution of local model to the global model. For UE u, we calculate the weight w_u with **average reward, average**

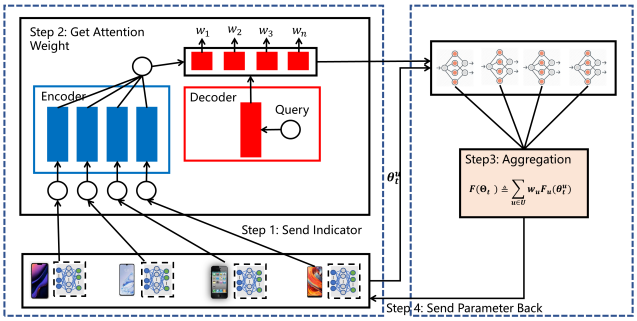


Fig. 6. The weighted federated aggregation process.

loss, training data, episode number, batch size as well as **hit rate.**

- **Average reward**: The average reward $\overline{C}_u$ for UE u is calculated by averaging all the local reward $C(\mathbf{s}_t^u, \mathbf{a}_t^u)$ during E local updates.
- **Average loss**: Average loss $\overline{L}_u$ for each UE is calculated by averaging the loss function $F_u(\boldsymbol{\theta}_t^u)$ output during E local training process.
- **Training data size**: M_u is the size of the replay memory, for those devices which have more memory resources, they can store more training data into the replay memory.
- **Episode number**: Different UEs may train different batch numbers of local episodes (denoted by $(E_u)^{1\times U}$) during one local training step. Besides, each E_u is a multiple of E.
- **Batch size**: m_u is mini-batch size of u, for those which have more computing capacity, they can train more data during one local training process.
- **Hit rate**: The average hit rate $\overline{h}_u$ during E time slots.

These above evaluation indicators can be described as $\mathbf{K}_u = [\overline{C}_u, \overline{L}_u, M_u, E_u, m_u, \overline{h}_u]$. In our model, we employ the attention mechanism which has received great success in some fields such as natural language processing [39] and computer vision [40]. Particularly, the evaluation indicator vector $\mathbf{K}_u$ and the UE u's local model parameters $\boldsymbol{\theta}_t^u$ are modeled as the key and the value in the attention mechanism, respectively. The target of our model is to get a more powerful agent who can get more reward, less loss and higher hit rate, so we design the query as

$$\mathbf{Q} = [\max_u(\overline{C}_u), \min_u(\overline{L}_u), \max_u(M_u), \max_u(E_u), \max_u(m_u), \max_u(\overline{h}_u)]. \quad (15)$$

The inputs of the BS consist of query $\mathbf{Q}$, keys $\mathbf{K}_u$ with dimension of d_k, and values $\boldsymbol{\theta}_t^u$. We calculate the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and use a softmax function for obtaining the weights on the values. For the weight factor w_u, we have

$$w_u = Attention(\mathbf{Q}, \mathbf{K}_u) = softmax(\frac{\mathbf{Q}\mathbf{K}_u^T}{\sqrt{d_k}}), \forall u \in \mathcal{U}. \quad (16)$$

D. Convergence Analysis

Assumptions: We make the following assumptions on the functions $(F_u)^{1\times U}$. Assumptions 1 and 2 are standard; typical examples are the l_2-norm regularized linear regression. It can also be used in neural network model. And Assumptions 3 and 4 have been made based on [41].

1) $(F_u)^{1\times U}$ are all L-smooth: for all x_1 and x_2, $F_u(x_1) \leq F_u(x_2) + (x_1 - x_2)^T \nabla F_u(x_2) + \frac{L}{2}||x_1 - x_2||_2^2$.
2) $(F_u)^{1\times U}$ are all μ-strongly convex: for all x_1 and x_2, $F_u(x_1) \geq F_u(x_2) + (x_1 - x_2)^T \nabla F_u(x_2) + \frac{\mu}{2}||x_1 - x_2||_2^2$.
3) With mini-batch $\widetilde{\mathcal{M}}_u$, the variance of stochastic gradients in each UE is bounded by $\mathbb{E}||\nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u) - \nabla F_u(\boldsymbol{\theta}_t^u)||^2 \leq \sigma_u^2$ for $u \in \mathcal{U}$.
4) The expected squared norm of stochastic gradients is uniformly bounded by $\mathbb{E}||\nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u)||^2 \leq G^2, \forall u \in \mathcal{U}, \forall t \in \mathcal{T}$.

Robustness of aggregation: We use F^* and F_u^* to represent the optimal value of $F(\boldsymbol{\Theta}_t)$ and $F_u(\boldsymbol{\theta}_t^u)$, respectively. The term Γ is used to quantify the effect of aggregation operation as

$$\Gamma = F^* - \sum_{u\in\mathcal{U}} w_u F_u^* + \frac{1}{6L} \sum_{u\in\mathcal{U}} w_u^2 \sigma_u^2. \quad (17)$$

For those who have more computing capacity, experience pool, training batch and better training data, the difference between F_u^* and F^* is theoretically less than other UEs, and the variance of stochastic gradients σ_u^2 is also smaller than others. By considering these divergences, we model a more complicated weight factor w_u in (16). The UEs with better optimal value will get more weight in the aggregation process, so the term Γ will be less than those who just average local parameter or those who just consider the data size of local data. With a less Γ, we will get a better convergence bound. Actually, we can consider Γ as a function of the aggregation weight w_u.

To derive our results, it is necessary to use the useful lemmas as follow. The proofs of these lemmas can be seen in the Appendix B.

Lemma 1 (Convergence of one step local training). Assumptions 1 and 2 holds. If $\eta_t \leq \frac{1}{4L}$, we have

$$\mathbb{E}||\overline{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\Theta}^*||^2 \leq (1 - \eta_t\mu)\mathbb{E}||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2 + \eta_t^2 \mathbb{E}||\mathbf{g}_t - \hat{\mathbf{g}}_t||^2 + 6L\eta_t^2\kappa + 2\mathbb{E}\sum_{u\in\mathcal{U}} w_u ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2, \quad (18)$$

where $\boldsymbol{\Theta}^*$ is the optimal global parameters and $\kappa = F^* - \sum_{u=1}^U w_u F_u^*$ which is the first term in Γ. With this lemma holds, we can guarantee that the difference between the locally trained parameters and the optimal parameters will not be too large. Meanwhile, it also ensures that with the increase of the number of training, the local parameters converge to the optimal parameters.

Lemma 2 (Stability of local training sampling). If Assumption 3 holds, we can obtain

$$\mathbb{E}||\mathbf{g}_t - \hat{\mathbf{g}}_t||^2 \leq \sum_{u\in\mathcal{U}} w_u^2 \sigma_u^2. \quad (19)$$

With this lemma holds, we can ensure the stability and effectiveness of local training sampling. It means that we can achieve the same optimal result as training with the all original data.

Lemma 3 (Boundness of the aggregation parameter). If Assumption 4 holds, η_t is non-increasing and $\eta_t \leq 2\eta_{t+E}$ for all $t \geq 0$. We have

$$\mathbb{E}\Big[\sum_{u\in\mathcal{U}} w_u||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2\Big] \leq 4\eta_t^2(E-1)^2G^2. \tag{20}$$

With this lemma, we can guarantee the boundness of the aggregation phase, which means that the difference between the local parameters and aggregation parameters will decrease as the training period increase. Finally, we can get stable aggregation parameters.

Convergence Analysis: Let the AWFDRL algorithm terminate after T iterations and return $\boldsymbol{\Theta}_T$ as the global model. We assume that T can be divisible by E so that we can always get the output $\boldsymbol{\Theta}_T$.

Theorem 1. Let Assumptions 1 to 4 hold and L, μ, σ_u, G be defined therein. Choose $\vartheta = \max\{8\frac{L}{\mu}, E\}$ and the learning rate $\eta = \frac{2}{\mu(\vartheta+t)}$. Then AWFDRL satisfies

$$\begin{aligned}\mathbb{E}[F(\boldsymbol{\Theta}_T)] - F^* \leq H(\frac{1}{T})\Big(&12L\Gamma + 16(E-1)^2G^2 \\ &+ \frac{(\vartheta+1)\mu^2}{2}||\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}^*||^2\Big),\end{aligned} \tag{21}$$

where

$$H(\frac{1}{T}) = \frac{L}{\mu^2(\vartheta+T)}. \tag{22}$$

Proof of Theorem 1: Please see the Appendix C.

In addition to the problem-related constants (L, μ, σ_u, G) in (21), the two most important variables are E and Γ. Particularly E can control the **convergence** rate of the AWFDRL, neither over-small nor over-large setting of E is good for the convergence. Γ show the **robustness** of our model as we set different weight coefficient w_u. We can infer that as the local training environment becomes more heterogeneous, the difference among the models trained by different equipments will become increasingly larger. Aggregating these various levels of local model with equal weight will cause the good-performance local model polluted by the poor-performance model. With the attention weights in (16), we can improve the model's robustness Γ under heterogeneous training equipment environment.

VI. Simulation Results

In this section, we evaluate the performance of the proposed AWFDRL framework based on simulation results.

A. Parameter Setting

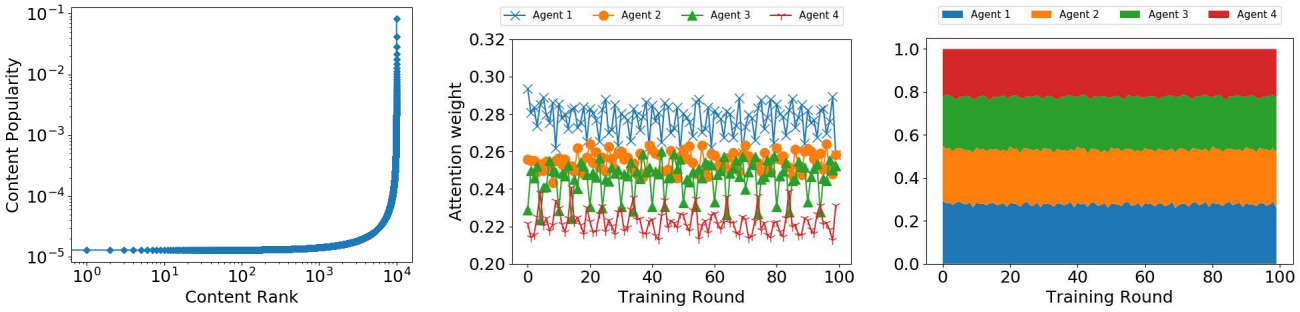
For simulation purpose, we consider 10,000 contents in the network, and the content sizes range from 5MB to 10MB. The number of UEs varies from 100 to 450. Besides, we set the cache sizes of each UE and BS as 300MB and 2GB for default, respectively. Meanwhile, the cloud server stores all the contents. Each UE runs a DRL agent with a two-layer neural network locally. We consider different scenarios in terms of computing instances and data sizes of UEs by deploying different replay memory capacities, ranging from 1000 to 2000. Moreover, for one aggregation step, different UEs train different episodes which range from 100 to 200. We consider the channel model as: the path loss (dB) is $36.8 + 36.7\log(d)$, where d is the distance in meter; the log-normal shadowing parameter and antenna gain is 7 dB and 5 dBi, respectively; the small-scale fading is Rayleigh fading with unit variance. The channel bandwidth is 20 MHz and the background noise power is -95 dBm. Besides, we set the replay memory capacity as [1000,2000], aggregation step as 100, local episode time as [100,200], mini-batch size [128,256], learning rate as 0.05, and reward decay as 0.9. According to the MZipf distribution [33], the content popularity is set with the corresponding factors $(\tau, \beta) = (-0.95, 0.5)$, and shown in Fig. 7(a).

B. Baseline Schemes and Performance Metrics

To evaluate the performance of the proposed AWFDRL framework under different user preferences, we consider the following baseline schemes as:

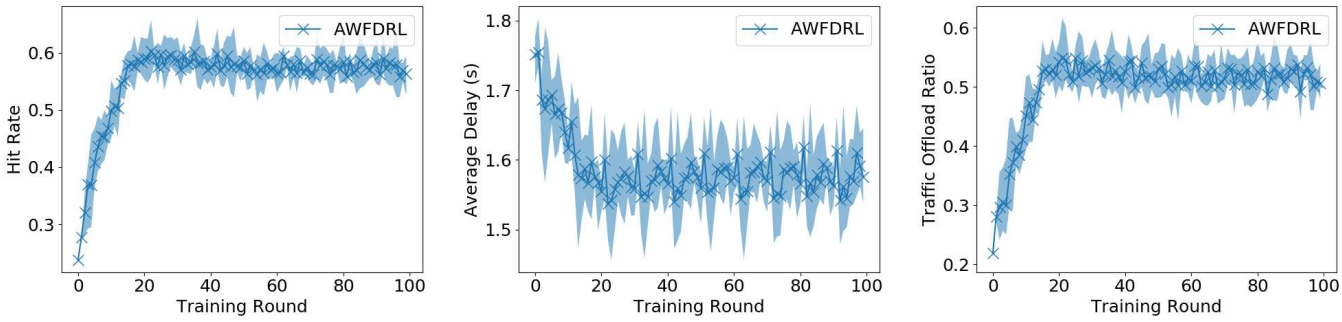
1) **FIFO** [9]: The earliest stored content is replaced by the new one.
2) **LRU** [10]: The least recently used content is replaced by the new one.
3) **LFU** [10]: The least frequently used content is replaced firstly.
4) **LFU with Dynamic Aging (LFUDA)** [42]: A variant of LFU that aims to address the cache pollution issue by employing a cache age counter to punish the access frequency of old contents.
5) **Greedy Dual Size with Frequency (GDSF)** [42]: Replace the content with the smallest key value for a certain utility (cost) function. The utility function is calculated based on the content size and access frequency.
6) **Federated Averaging (FedAvg)** [41]: Run the DQN model in each UE and aggregate the local model with aggregation weight calculated by data size during the trainig process.
7) **FedProx**: [25] A generalization and re-parametrization of FedAvg. It improves FedAvg by allowing the episode to vary according to the characteristics of the network.
8) **Optimal method (OPT)** [37]: Evict the content which has the largest next-access time. However, the OPT algorithm is not implementable since it needs future information in adavance.

To evaluate the schemes, we use the following performance metrics as: 1) **hit rate** (satisfied by the local cache, D2D sharing or BS_s); 2) **average delay**; and 3) **traffic offload ratio** (satisfied by the local cache, D2D sharing or BS_s).



(a) Content popularity (b) Line graph of attention weights (c) Stack graph of attention weights

Fig. 7. Content popularity and the changes of different agents' attention weights.



(a) Hit rate (b) Average delay (c) Traffic offload ratio

Fig. 8. Performance comparison in the case study.

C. Case Study

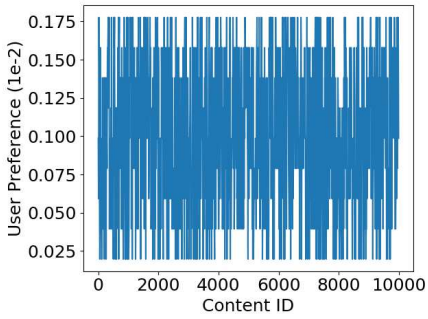
We give a simple case here to reveal the changes of attention weights and model performance. In this case, we consider four UEs, and set the experience pool sizes of these UEs as $[2000, 1500, 1500, 1500]$, the episode numbers as $[200, 150, 150, 100]$, and the batch sizes as $[256, 256, 128, 128]$.

Based on the parameter settings and (16), we can obtain each UE's attention weight in the aggregation phase. The changes of different DRL agents' attention weights are shown in Fig. 7(b). During the training process, Agent 1 takes the highest weight (around 28%) in the aggregation phase, while Agent 4 takes the least weight (around 22%). It is mainly because Agent 1 has the most powerful computing capacity. Meanwhile, we can see that the results fluctuate due to different sizes of the training data. In order to show different weights intuitively, we also put them into a stack graph as shown in Fig. 7(c) where the maximum value of the y-coordinate is 1. From Fig. 7(c), we can see that the blue part (Agent 1) is the widest and the red part (Agent 4) is the narrowest, which means that Agent 1 has the greatest impact on the global model. In Fig. 8, the solid line means the average hit rate, average delay, traffic offload ratio of all agents, respectively; the upper and lower boundaries of the shadow region represent the maximum and minimum of different indicators in different sub-graphs. We can see that after 20 training rounds, all agents converge to a relative stable hit rate (59%), average delay (1.55 second) and traffic offload ratio (53%).

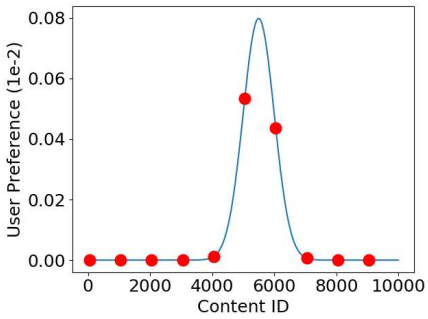
D. Effects of Different User Preferences

We consider Random and Gaussian preferences of users as shown in Fig. 9. Particularly in Fig. 9(a), the user preference is set randomly, which indicates that the user has no preference for a specific content and would like to access all the contents. In Fig. 9(b), we initialize a list based on the Gaussian distribution $X \sim \mathcal{N}(1, 1)$ and then assign it as the user preferences for contents to simulate that users prefer content 4500-6500.

Fig. 10 shows the effects of different user preferences on the AWFDRL scheme and the aforementioned baseline schemes. From Fig. 10(a)-(c), when users have random preferences, the proposed AWFDRL scheme outperforms the first five baseline schemes by the improvements of 10%-15%, 50-80ms and 5%-18% in terms of hit rate, average delay and traffic offload ratio, respectively. It is mainly because that the proposed AWFDRL scheme chooses the evicted contents more intelligently based on the user preference and global content popularity. Meanwhile, our method also outperforms other learning-based methods such as FedAvg and FedProx in all performance metrics. Specially, as the cache size of UEs increases, the input state size of the local DQN agent also increases, which makes it hard to train and aggregate the global model for simple FL methods like FedAvg. FedProx improves FedAvg by allowing the agent to run the different



(a) Random preference



(b) Gaussian preference

Fig. 9. Random and Gaussian preferences of users.

numbers of local episodes. Actually, we can regard FedProx as a special case of our proposed AWFDRL since we also take the local episodes into the consideration when we aggregate the global model. Moreover, compared with the OPT scheme, AWFDRL has only a 10% loss gap of hit rate, 50ms gap of average delay and 8% gap of traffic offload ratio. From Fig. 10(d)-(f), when users have Gaussian preferences, the proposed AWFDRL scheme outperforms the first five baseline schemes by the improvements of 10%-20%, 80-120ms and 5%-18% in terms of hit rate, average delay and traffic offload ratio, respectively. Also, the hit rate and traffic offload ratio are larger whereas the average delay is less compared with those with random preferences. That is mainly because the evicted contents can be more easily chosen when the user preference is known in advance. Besides, it is observed that the proposed AWFDRL scheme greatly outperforms other schemes, which can be explained by that the AWFDRL scheme can choose the offload node based on the network state and UE state.

E. Effects of Different Numbers of UEs

Fig. 11 shows the performance comparison of different caching schemes under different numbers of UEs. Here, the number of UEs ranges from 100 to 450 (which is the maximum value that a general 4G BS can serve simultaneously). From Fig. 11(a), the proposed AWFDRL outperforms the first five baseline schemes by the improvements of 10%-20% on hit rate and has only a 5%-12% loss gap of hit rate compared with OPT scheme. From Fig. 11(b), the proposed AWFDRL scheme achieves the lowest average delay compared with all the baseline schemes. Meanwhile, from Fig. 11(c), it is observed that the traffic offload ratio increases along with the increase of the number of UEs since the UEs have more desirable selected nodes to fetch the requested contents. Moreover, when the number of UEs is sufficiently large, the AWFDRL scheme offloads 80% traffic in the network. From Fig. 11, AWFDRL always outperforms FedAvg and FedProx in all performance metrics. Moreover, the gap between AWFDRL and FedAvg increases as the number of UEs increases. It is because when there are a large number of UEs, the difference between the various local models is very huge and aggregating these various local models with simple weights calculated by data size cannot get a well-performed global model.

F. Comparison of Different Training Methods

As the rewards $C(\mathbf{s}_t^u, \mathbf{a}_t^u)$ are calculated based on the D2D sharing gain and content delay cost, the larger average reward value is, the more traffic offload and the lower average delay we will get during the training process. So we compare AWFDRL, FedAvg and FedProx in term of average reward under different episode numbers, experience pool sizes, batch sizes and cache sizes, which shows the model performance under different parameter settings. The corresponding simulation results are shown in Fig. 12(a)-12(d). From Fig. 12(a), as the episode number increases, the average reward in each considered method increases since the training model is more intelligent with a larger episode number. From Fig. 12(b), when the experience pool size of the agent increases, the average reward in each considered method also increases as the agent can store more history transitions and learn more from the history actions. Meanwhile, from Fig. 12(c), as the batch size increases from 32 to 256, the agent gets more data during the training process and gets more average reward. However, it will take more time to finish the local training phase with larger batch sizes and episode numbers. Besides, from Fig. 12(d), when the user cache size increases, the average reward in each considered method increases. It is caused by the improvement of hit rate, average delay and traffic offload. Particularly, it is observed that the AWFDRL scheme always outperforms the baseline training methods on the average reward under all user cache sizes and episode numbers.

Moreover, to evaluate the model performance in different heterogeneous environments, we compare AWFDRL, FedAvg and FedProx in four cases. In each case, we consider four UEs. In case 1, we set the same parameters (i.e., experience pool size, episode number, batch size for all UEs) in these two training methods. In case 2, we set different experience pool sizes $[2500, 2000, 2000, 1500]$ for the UEs. In case 3, we set different experience pool sizes as the same in case 2, and different episode numbers $[200, 150, 150, 100]$. In case 4, we also set different batch sizes for different UEs as $[256, 256, 128, 128]$. The simulation environment is increasingly heterogeneous from case 1 to case 4. The corresponding simulation results are shown in Fig. 12(e). We can see that as the simulation environment becomes increasingly heteroge-

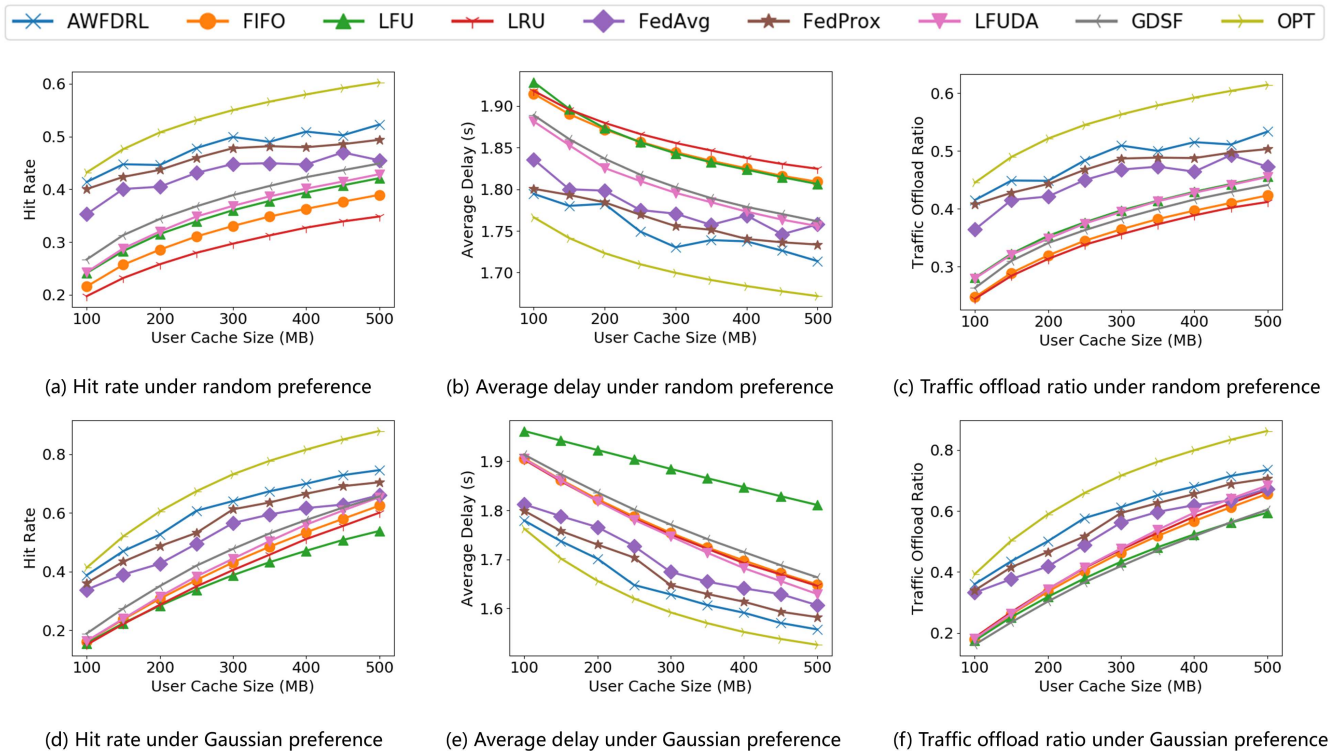


Fig. 10. Performance comparison of different caching schemes under different user preferences.

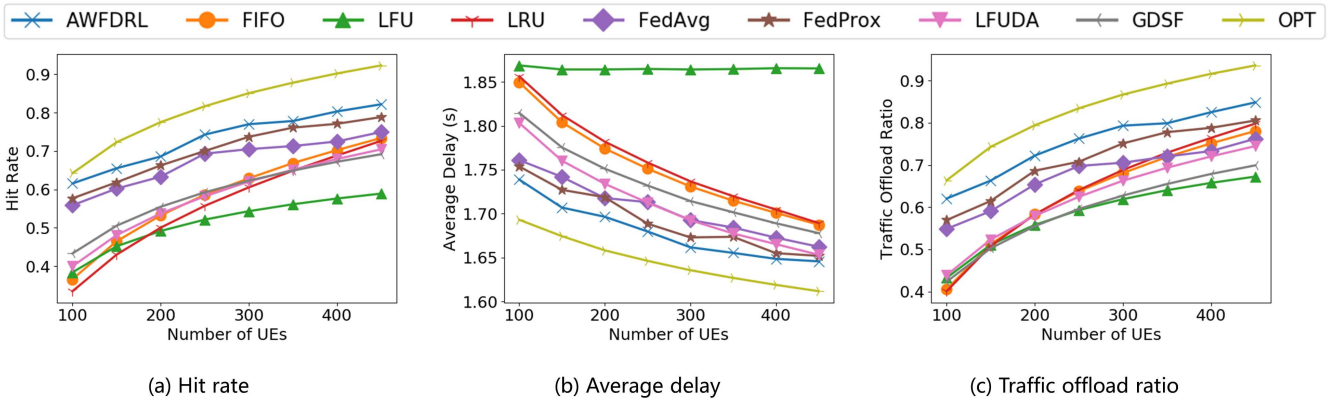
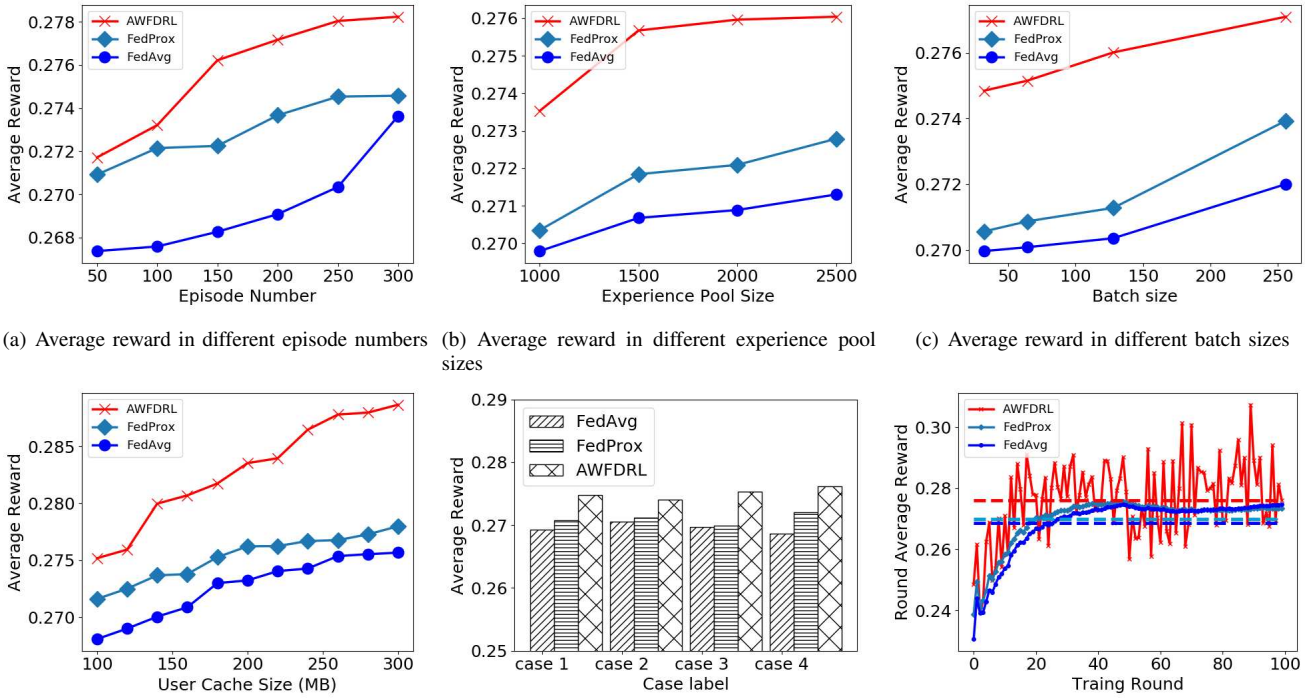


Fig. 11. Performance comparison of different caching schemes under different numbers of UEs.

neous, the FedAvg method will get less average reward (0.267 in case 4). The Average Reward of FedProx is stable which is about 0.27. However, compared with the others, the proposed AWFDRL is less affected and achieves more average reward (0.275 in case 4). Moreover, the proposed AWFDRL scheme outperforms the FedAvg and FedProx methods in all four cases, and the gaps between FedAvg and AWFDRL become larger when the simulation environment tends to be more heterogeneous. It is because that with the attention weights, UEs with more powerful computing capacity in the AWFDRL scheme can get larger weights in the aggregation phase; then after the training, AWFDRL reduces the impacts of poorly trained model on the global model and guarantees that the global model can achieve the high average award. For instance, Fig. 12(f) shows the training round average reward of case 4, and it demonstrates that during the training process, the proposed AWFDRL achieves the faster convergence of average reward (10 training rounds) than FedAvg (25 training rounds) and FedProx (20 training rounds).

VII. Conclusion

In this paper, we have investigated the issue of D2D-assisted heterogeneous collaborative edge caching in mobile networks, and formulated the problem of joint node selection



(a) Average reward in different episode numbers (b) Average reward in different experience pool sizes (c) Average reward in different batch sizes

(d) Average reward in different cache sizes (e) Average reward in different cases (f) Round Average reward of case 4

Fig. 12. Performance demonstration of average reward.

and cache replacement as an LT-MILP. In particular, we have employed the DQN model to control the joint decision process dynamically based on the network state and historical information. To efficiently solve this problem, we have proposed the AWDFRL framework, an improved FL framework which trains the DQN model in a distributed manner by keeping the data in the local UEs while addressing the issue of the model aggregation among heterogeneous UEs. Most importantly, we have employed the attention mechanism to control the model weights in the FL aggregation step, which can address the imbalance issue of local model quality. We have derived the expectation convergence of AWFDRL. Simulation results have been presented to show that compared with the existing schemes, the proposed AWFDRL framework can effectively improve hit rate, reduce average delay and offload traffic. Moreover, the proposed AWFDRL framework has been shown to outperform the existing FL methods such as FedAvg and FedProx in term of average reward, due to the attention mechanism.

APPENDIX

A. Definition

Let $\mathbb{T}_E$ be the set of aggregation steps ($\mathbb{T}_E = \{nE | n = 1, 2, ...\}$). If $t+1 \in \mathbb{T}_E$, the system run aggregation step. Then we define

$$\boldsymbol{\theta}_{t+1}^u = \begin{cases} \boldsymbol{\theta}_t^u - \eta_t \nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u) & if\ t+1 \notin \mathbb{T}_E, \\ \sum_{u \in \mathcal{U}} w_u \boldsymbol{\theta}_t^u & if\ t+1 \in \mathbb{T}_E. \end{cases} \tag{23}$$

We define $\overline{\boldsymbol{\theta}}_t = \sum_{u \in \mathcal{U}} w_u \boldsymbol{\theta}_t^u$ for all t. If $t \in \mathbb{T}_E$, we can get $\overline{\boldsymbol{\theta}}_t = \boldsymbol{\Theta}_t$, otherwise, we cannot get $\overline{\boldsymbol{\theta}}_t$. Moreover, we have $\hat{\mathbf{g}}_t = \sum_{u \in \mathcal{U}} w_u \nabla F_u(\boldsymbol{\theta}_t^u)$ and $\mathbf{g}_t = \sum_{u \in \mathcal{U}} w_u \nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u)$. Therefore, $\overline{\boldsymbol{\theta}}_{t+1} = \overline{\boldsymbol{\theta}}_t - \eta_t \mathbf{g}_t$ and $\mathbb{E}\mathbf{g}_t = \hat{\mathbf{g}}_t$.

B. Proof of Lemma

1) Proof of Lemma 1: Notice that we have $\overline{\boldsymbol{\theta}}_{t+1} = \overline{\boldsymbol{\theta}}_t - \eta_t \mathbf{g}_t$, then

$$\begin{aligned} ||\overline{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\Theta}^*||^2 &= ||\overline{\boldsymbol{\theta}}_t - \eta_t \mathbf{g}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t + \eta_t \hat{\mathbf{g}}_t||^2 \\ &= \underbrace{||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t||^2}_{A_1} + \eta_t^2 ||\hat{\mathbf{g}}_t - \mathbf{g}_t||^2 \\ &\quad + \underbrace{2\eta_t \langle \overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t, \hat{\mathbf{g}}_t - \mathbf{g}_t \rangle}_{A_2}. \end{aligned} \tag{24}$$

Note that $\mathbb{E}A_2 = 0$. We next focus on bounding of A_1. Again we split A_1 into three terms as

$$||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t||^2 = ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2 \underbrace{-2\eta_t \langle \overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*, \hat{\mathbf{g}}_t \rangle}_{B_1} + \underbrace{\eta_t^2 ||\hat{\mathbf{g}}_t||^2}_{B_2}. \tag{25}$$

From the L-smoothness of $F_u(\cdot)$, if follows

$$||\nabla F_u(\boldsymbol{\theta}_t^u)||^2 \leq 2L(F_u(\boldsymbol{\theta}_t^u) - F_u^*). \tag{26}$$

By the convexity of $||\cdot||^2$ and (26), we have

$$\begin{aligned} B_2 = \eta_t^2 ||\hat{\mathbf{g}}_t||^2 &\leq \eta_t^2 \textstyle\sum_{u=1}^U w_u ||\nabla F_u(\boldsymbol{\theta}_t^u)||^2 \\ &\leq 2L\eta_t^2 \textstyle\sum_{u=1}^U w_u (F_u(\boldsymbol{\theta}_t^u) - F_u^*). \end{aligned} \tag{27}$$

Note that

$$\begin{aligned} B_1 &= -2\eta_t \textstyle\sum_{u=1}^{U} w_u \langle \overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u, \nabla F_u(\boldsymbol{\theta}_t^u)\rangle \\ &\quad -2\eta_t \textstyle\sum_{u=1}^{U} w_u \langle \boldsymbol{\theta}_t^u - \boldsymbol{\Theta}^*, \nabla F_u(\boldsymbol{\theta}_t^u)\rangle. \end{aligned} \tag{28}$$

By Cauchy-Schwarz inequality and AM-GM inequality, we have

$$-2\langle \overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u, \nabla F_u(\boldsymbol{\theta}_t^u)\rangle \leq \frac{1}{\eta_t}||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2 + \eta_t ||\nabla F_u(\boldsymbol{\theta}_t^u)||^2. \tag{29}$$

Based on the μ-strong convexity of $f_i(\cdot)$, we obtain

$$-2\langle \boldsymbol{\theta}_t^u - \boldsymbol{\Theta}^*, \nabla F_u(\boldsymbol{\theta}_t^u)\rangle \leq -(F_u(\boldsymbol{\theta}_t^u) - F_u(\boldsymbol{\Theta}^*)) - \frac{\mu}{2}||\boldsymbol{\theta}_t^u - \boldsymbol{\Theta}^*||^2. \tag{30}$$

Based on (25), (26), (28), (29), (30), we have

$$\begin{aligned} A_1 &= ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t||^2 \\ &\leq ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2 + 2L\eta_t^2 \textstyle\sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F_u^*) \\ &\quad +\eta_t \textstyle\sum_{u\in\mathcal{U}} w_u \big(\frac{1}{\eta_t}||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2 + \eta_t ||\nabla F_u(\boldsymbol{\theta}_t^u)||^2\big) \\ &\quad -2\eta_t \textstyle\sum_{u=1}^{U} w_u \big(F_u(\boldsymbol{\theta}_t^u) - F_u(\boldsymbol{\Theta}^*) + \frac{\mu}{2}||\boldsymbol{\theta}_t^u - \boldsymbol{\Theta}^*||^2\big) \\ &= (1-\mu\eta_t) \textstyle\sum_{u\in\mathcal{U}} w_u ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2 \\ &\quad + \textstyle\sum_{u\in\mathcal{U}} w_u ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2 + C, \end{aligned} \tag{31}$$

where $C = 4L\eta_t^2 \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F_u^*) - 2\eta_t \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F_u(\boldsymbol{\Theta}^*))$.

Then we need to find the bound of C. We define $\vartheta_t = 2\eta_t(1-2L\eta_t)$. Since $\eta_t \leq \frac{1}{4L}, \eta_t \leq \vartheta_t \leq 2\eta_t$. Then we can divide C into two terms as

$$\begin{aligned} C &= -2\eta_t(1-2L\eta_t) \textstyle\sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F_u^*) \\ &\quad +2\eta_t \textstyle\sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\Theta}^*) - F_u^*) \\ &= -\vartheta_t \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F^*) \\ &\quad +(2\eta_t - \vartheta_t) \sum_{u\in\mathcal{U}} w_u (F^* - F_u^*) \\ &= \underbrace{-\vartheta_t \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F^*)}_{D} + 4L\eta_t^2\kappa. \end{aligned} \tag{32}$$

To bound the term D, we have

$$\begin{aligned} \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F^*) &= \sum_{u\in\mathcal{U}} w_u (F_u(\boldsymbol{\theta}_t^u) - F_u(\overline{\boldsymbol{\theta}}_t)) \\ &\quad + \sum_{u\in\mathcal{U}} w_u (F_u(\overline{\boldsymbol{\theta}}_t - F^*)) \\ &\geq \sum_{u\in\mathcal{U}} w_u \langle \nabla F_u(\overline{\boldsymbol{\theta}}_t), \overline{\boldsymbol{\theta}}_t^u - \overline{\boldsymbol{\theta}}_t\rangle \\ &\quad +(F(\overline{\boldsymbol{\theta}}_t) - F^*) \\ &\geq -\sum_{u\in\mathcal{U}} w_u \Big[\eta_t L(F_u(\overline{\boldsymbol{\theta}}_t) - F_u^*) \\ &\quad +\tfrac{1}{2\eta_t}||\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_t||^2\Big] + (F(\overline{\boldsymbol{\theta}}_t) - F^*), \end{aligned} \tag{33}$$

where these two inequality are derived from the convexity of $F_u(\cdot)$, and from (26) and AM-GM inequality, respectively.

Therefore, we have

$$\begin{aligned} C &= \vartheta_t \sum_{u\in\mathcal{U}} w_u \Big[\eta_t L(F_u(\overline{\boldsymbol{\theta}}_t) - F_u^*) + \tfrac{1}{2\eta_t}||\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_t||^2\Big] \\ &\quad -\vartheta_t(F(\overline{\boldsymbol{\theta}}_t) - F^*) + 4L\eta_t^2\kappa \\ &= \vartheta_t(\eta_t L - 1) \sum_{u\in\mathcal{U}} w_u (F_u(\overline{\boldsymbol{\theta}}_t) - F_u^*) \\ &\quad +(4L\eta_t^2 + \vartheta_t\eta_t L)\kappa + \tfrac{\vartheta_t}{2\eta_t} \sum_{u\in\mathcal{U}} w_u ||\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_t||^2 \\ &\leq 6L\eta_t^2\kappa + \sum_{u\in\mathcal{U}} w_u ||\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_t||^2, \end{aligned} \tag{34}$$

where in the last inequality, we utilize the following facts: (1) $\eta_t L - 1 \leq -\frac{3}{4} \leq 0$ and $\sum_{u\in\mathcal{U}} w_u (F_u(\overline{\boldsymbol{\theta}}_t) - F_u^*) = F(\overline{\boldsymbol{\theta}}_t) - F^* \geq 0$ (2) $\kappa \geq 0$ and $4L\eta_t^2 + \vartheta_t\eta_t L \leq 6\eta_t^2 L$ and (3) $\frac{\vartheta_t}{2\eta_t} \leq 1$.

By substituting C into A_1, we get

$$\begin{aligned} A_1 &= ||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^* - \eta_t \hat{\mathbf{g}}_t||^2 \\ &\leq (1-\mu\eta_t)||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2 + 2\sum_{u\in\mathcal{U}} w_u ||\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_t||^2 + \\ &\quad 6L\eta_t^2\kappa. \end{aligned} \tag{35}$$

Using (35) and taking expectation on both sides of (24), we can erase the randomness from stochastic gradients. The whole proof is complete.

2) Proof of Lemma 2: If Assumption 3 holds, the variance of the stochastic gradients in UE u is bounded by σ_u^2, then we have $\mathbb{E}||\mathbf{g}_t - \hat{\mathbf{g}}_t||^2 = \mathbb{E}\Big\| \sum_{u\in\mathcal{U}} w_u (\nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u) - \nabla F_u(\boldsymbol{\theta}_t^u))\Big\|^2 = \sum_{u\in\mathcal{U}} w_u^2 \mathbb{E}||\nabla F_u(\boldsymbol{\theta}_t^u, \widetilde{\mathcal{M}}_u) - \nabla F_u(\boldsymbol{\theta}_t^u)||^2 \leq \sum_{u\in\mathcal{U}} w_u^2\sigma_u^2$.

3) Proof of Lemma 3: Since AWFDRL requires information exchange every E steps. Therefore, for any $t \geq 0$, there exists a $t_0 \leq t$, such that $t - t_0 \leq E-1$ and $\boldsymbol{\theta}_{t_0}^u = \overline{\boldsymbol{\theta}}_{t_0}$ for all $u = 1, 2, ..., U$. Also, we use the fact that η_t is non-increasing and $\eta_t \leq 2\eta_t$ for all $t - t_0 \leq E-1$, then we have $\sum_{u\in\mathcal{U}} w_u \mathbb{E}||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t^u||^2 = \sum_{u\in\mathcal{U}} w_u \mathbb{E}||(\boldsymbol{\theta}_t^u - \overline{\boldsymbol{\theta}}_{t_0}) - (\overline{\boldsymbol{\theta}}_t - \overline{\boldsymbol{\theta}}_{t_0})||^2 \leq \sum_{u\in\mathcal{U}} w_u (E-1)^2\eta_t^2 G^2 \leq 4\eta_t^2(E-1)^2G^2$.

C. Proof of Theorem 1

Proof. Let $\Delta_t = \mathbb{E}||\overline{\boldsymbol{\theta}}_t - \boldsymbol{\Theta}^*||^2$. With Lemmas 1-3, we have

$$\Delta_{t+1} \leq (1-\eta_t\mu)\Delta_t + \eta_t^2 H(\Gamma, E), \tag{36}$$

where $H(\Gamma, E) = 6L\Gamma + 8(E-1)^2G^2$.

For a diminishing step size, $\eta_t = \frac{\beta}{t+\vartheta}$ for some $\beta \geq \frac{1}{\mu}$ and $\vartheta \geq 0$ such that $\eta_1 \leq min\{\frac{1}{\mu}, \frac{1}{4L}\}$ and $\eta_t \leq 2\eta_{t+E}$. We will prove $\Delta_t \leq \frac{v}{\vartheta+t}$ where $v = max\{\frac{\beta^2 H(\Gamma,E)}{\beta\mu - 1}, (\vartheta+1)\Delta_1\}$.

We prove it by induction. At first, the definition of v guarantees that it holds for $t = 1$. When the conclusion also holds for time slot t, we have

$$\begin{aligned} \Delta_{t+1} &\leq (1-\eta_t\mu)\Delta_t + \eta_t^2 H(\Gamma, E) \\ &\leq \left(1 - \frac{\beta\mu}{t+\vartheta}\right)\frac{v}{t+\vartheta} + \frac{\beta^2 H(\Gamma,E)}{(t+\vartheta)^2} \\ &= \frac{t+\vartheta-1}{(t+\vartheta)^2}v + \left[\frac{\beta^2 H(\Gamma,E)}{(t+\vartheta)^2} - \frac{\beta\mu - 1}{(t+\vartheta)^2}v\right] \leq \frac{v}{t+\vartheta+1}. \end{aligned} \tag{37}$$

Then by considering the strong convexity of $F(\cdot)$, we have

$$\mathbb{E}[F(\overline{\boldsymbol{\theta}}_t)] - F^* \leq \frac{L}{2}\Delta_t \leq \frac{L}{2}\frac{v}{\vartheta + t}. \tag{38}$$

Specifically, if choosing $\beta = \frac{2}{\mu}$, we have $\eta_t = \frac{2}{\mu}\frac{1}{\vartheta+t}$ and

$$\begin{aligned} \mathbb{E}[F(\overline{\boldsymbol{\theta}}_t)] - F^* &\leq \frac{L}{2(\vartheta+t)} \cdot \left(\frac{4H(\Gamma,E)}{\mu^2} + (\vartheta+1)\Delta_1 \right) \\ &\leq \frac{L}{\mu^2(\vartheta+t)}(2H(\Gamma,E) + \frac{\vartheta+1}{2}\Delta_1), \end{aligned} \tag{39}$$

where $\overline{\boldsymbol{\theta}}_t$ is $\boldsymbol{\Theta}_T$ in (21) when the algorithm terminates.

REFERENCES

[1] R. Li, Y. Zhao, C. Wang, X. Wang, V. C. M. Leung, X. Li, and T. Taleb, "Edge Caching Replacement Optimization for D2D Wireless Networks via Weighted Distributed DQN," in *Proc. IEEE WCNC*, pp. 1-6, May 2020.

[2] H. Sheng, Y. Zheng, W. Ke, D. Yu, X. Cheng, W. Lv, and Z. Xiong, "Mining Hard Samples Globally and Efficiently for Person Re-identification," *IEEE Internet of Things Journal*, pp. 1-1, Mar. 2020.

[3] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 2, pp. 869-904, Second Quarter 2020.

[4] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: Spatial-Temporal Collaborative Sampling in Flow-aware Software Defined Networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 999-1013, Jun. 2020.

[5] S. Deng, H. Zhao, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge Intelligence: the Confluence of Edge Computing and Artificial Intelligence," *IEEE Jounal of Internet of Things*, pp. 1-1, Apr. 2020.

[6] H. Tang, and Z. Ding, "Mixed Mode Transmission and Resource Allocation for D2D Communication," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 1, pp. 162-175, Jan. 2016.

[7] G. Cao, L. Yin, and R. Das, "Cooperative cache-based data access in ad hoc networks," *Computer*, vol. 37, no. 2, pp. 32-39, Aug. 2004.

[8] C. Zhang, H. Pang, J. Liu, et al., "Toward Edge-Assisted Video Content Intelligent Caching With Long Short-Term Memory Learning," *IEEE Access*, vol. 7, pp. 152832 - 152846, Oct. 2019.

[9] L. Tang, Q. Huang,W. Lloyd, S.Kumar, and K. Li, "Ripq: Advanced photo caching on flash for Facebook," in *Proc. ACM FAST*, Feb. 16-19, 2015, pp. 373-386.

[10] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of Facebook photo caching," in *Proc. ACM SOSP*, Nov. 2013, pp. 167-181.

[11] S. Podlipnig and L. Böszörmenyi, "A survey of Web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374-398, Dec. 2003.

[12] J. Wang, "A survey of Web caching schemes for the Internet," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 36-46, Oct. 1999.

[13] H. Pang, J. Liu, X. Fan and L. Sun, "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *Proc. IEEE/ACM IWQoS*, Jun. 2018, pp. 1-6.

[14] F. Xu, F. Yang, S. Bao and C. Zhao, "DQN Inspired Joint Computing and Caching Resource Allocation Approach for Software Defined Information-Centric Internet of Things Network," *IEEE Access*, vol. 7, pp. 61987-61996, May 2019.

[15] H. Van, A. Guez, D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI*, Mar. 2016.

[16] W. Jiang, G. Feng, S. Qin, and Y. Liu, "Multi-agent reinforcement learning based cooperative content caching for mobile edge networks," *IEEE Access*, vol. 7, pp. 61856-6167, May 2019.

[17] L. Sweeney. "Simple demographics often identify people uniquely," 2000.

[18] B. McMahan, E. Moore, D. Ramage, et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, vol. 54, pp. 1273-1282, 2016.

[19] G. Qiao, S. Leng, S. Maharjan, et al., "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247-257, Jan. 2019.

[20] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," in *Proc. NIPS*, Dec. 2017, pp. 5998-6008.

[21] J. Konečný, B. McMahan, X. Yu, et al. "Federated learning: Strategies for improving communication efficiency," *arXiv preprint* arXiv:1610.05492, 2016.

[22] S. Shen, Y. Han, X. Wang, et al., "Computation Offloading with Multiple Agents in Edge-Computing–Supported IoT," *ACM Trans. Sens. Netw.*, vol. 16, no. 1, pp.1-27, Dec. 2019.

[23] Z. Yu, J. Hu, G. Min, et al., "Federated learning based proactive content caching in edge computing," in *Proc. IEEE GLOBECOM*, Dec. 2018, pp. 1-6.

[24] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Netw. Mag.*, vol.33, no.5, pp.156-165, 2019.

[25] T. Li, A. Sahu, M. Zaheer, et al., "Federated optimization in heterogeneous networks," *arXiv preprint* arXiv:1812.06127, 2020.

[26] Y. Li, S. Yang, X. Ren, et al., "Asynchronous Federated Learning with Differential Privacy for Edge Intelligence," *arXiv preprint* arXiv:1912.07902. Dec. 2019.

[27] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 2134-2143, Mar. 2020.

[28] K. Bonawitz, H. Eichner, W. Grieskamp, et al., "Towards Federated Learning at Scale: System Design," *arXiv preprint* arXiv:1902.01046. Feb. 2019.

[29] C. Xie, S. Koyejo, I. Gupta, "Asynchronous Federated Optimization," *arXiv preprint* arXiv:1903.03934. Mar. 2019.

[30] Y. Chen, X. Sun and Y. Jin, "Communication-Efficient Federated Deep Learning With Layerwise Asynchronous Model Update and Temporally Weighted Aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1 - 10, Dec. 2019.

[31] Y. Lu, X. Huang, K. Zhang, S. Maharjan and Y. Zhang, "Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298-4311, Apr. 2020.

[32] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing," *IEEE Access*, vol. 8, pp. 48970-48981, Mar. 2020.

[33] H. Mohamed, and S. Osama, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE-ACM Trans. Netw.*, vol. 16, no. 6, pp. 1447-1460, Dec. 2008.

[34] X. Li, X. Wang, J. Wan, et al., "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1768-1785, Aug. 2018.

[35] T. Govier, Social trust and human communities. *McGill-Queen's Press-MQUP*, 1997.

[36] C. Selvi, E. Sivasankar, "A Novel Singularity Based Improved Tanimoto Similarity Measure for Effective Recommendation Using Collaborative Filtering," in *International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 256-262, Noida, 2018.

[37] X. Wang, C. Wang, X. Li, V. C. M. Leung, T. Taleb, "Federated Deep Reinforcement Learning for Internet of Things with Decentralized Cooperative Edge Caching," *IEEE Internet Things J.*, pp. 1-1, April 2020.

[38] G. Kleinberg and E. Tardos, "Algorithm Design," *Cornell Univ.*, Ithaca, NY, USA, 2004.

[39] B. Dzmitry, C. Kyunghyun, and B. Yoshua, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv preprint* arXiv:1409.0473, 2014;

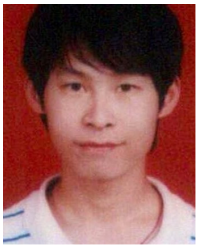
[40] K. Xu, J. Ba, R. Kiros, et al., "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. ICML*, Jun. 2015, pp. 2048-2057.

[41] X. Li, K. Huang, W. Yang, et al., "On the convergence of fedavg on non-iid data," *arXiv preprint* arXiv:1907.02189, 2019.

[42] M. Arlitt, L. Cherkasova, J. Dilley, et al., "Evaluating content management techniques for web proxy caches," in *Proc. ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3-11, Mar. 2000.

Xiaofei Wang (S'06, M'13, SM'18) is currently a Professor with the Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, China. He got master and doctor degrees in Seoul National University from 2006 to 2013, and was a Post-Doctoral Fellow with The University of British Columbia from 2014 to 2016. Focusing on the research of social-aware cloud computing, cooperative cell caching, and mobile traffic offloading, he has authored over 100 technical papers in the IEEE JSAC, the IEEE TWC, the IEEE WIRELESS COMMUNICATIONS, the IEEE COMMUNICATIONS, the IEEE TMM, the IEEE INFOCOM, and the IEEE SECON. He was a recipient of the National Thousand Talents Plan (Youth) of China. He received the "Scholarship for Excellent Foreign Students in IT Field" by NIPA of South Korea from 2008 to 2011, the "Global Outstanding Chinese Ph.D. Student Award" by the Ministry of Education of China in 2012, and the Peiyang Scholar from Tianjin University. In 2017, he received the "Fred W. Ellersick Prize" from the IEEE Communication Society.

Ruibin Li (S'20) received his B.S degrees in School of Computer Science and Technology, College of Intelligence and Computing at Tianjin University, Tianjin, China, in 2019. He is currently pursuing the M.S. degree in the School of Computer Science and Technology, College of Intelligence and Computing at Tianjin University, Tianjin, China. His current research interests include edge caching, distributed federated learning optimization, reinforcement learning, edge intelligence and deep learning.



Chenyang Wang (S'18) received his B.S and M.S. degrees in computer science and technology from Henan Normal University, Xinxiang, Henan province, in 2013 and 2017, respectively. He is currently pursuing the Ph.D. degree in the School of Computer Science and Technology, College of Intelligence and Computing at Tianjin University, Tianjin, China. His current research interests include edge computing, big data analytics, reinforcement learning, and deep learning. He received the "Best Student Paper Award" of the 24th International Conference on Parallel and Distributed Systems by IEEE Computer Society in 2018.

Xiuhua Li (S'12, M'19) received the B.S. degree from the Honors School, Harbin Institute of Technology, Harbin, China, in 2011, the M.S. degree from the School of Electronics and Information Engineering, Harbin Institute of Technology, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, in 2018. He joined Chongqing University through One-Hundred Talents Plan of Chongqing University in 2019. He is currently a tenure-track Assistant Professor with the School of Big Data & Software Engineering, and the Dean of the Institute of Intelligent Software and Services Computing associated with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Education Ministry, China. His current research interests are 5G/6G mobile Internet, mobile edge computing and caching, big data analytics and machine learning.

Tarik Taleb received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is the founder and the Director of the MOSA!C Lab, Espoo, Finland. He is a part-time Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a laboratory fully funded by KDDI until 2009. He was a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. He was then leading the NEC Europe Labs Team, involved with research and development projects on carrier cloud platforms, an important vision of 5G systems. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has also been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include architectural enhancements to mobile core networks (particularly 3GPP's), network softwarization and slicing, mobile cloud networking, network function virtualization, software defined networking, mobile multimedia streaming, intervehicular communications, and social media networking.

Prof. Taleb was a recipient of the 2017 IEEE ComSoc Communications Software Technical Achievement Award in 2017 for his outstanding contributions to network softwarization and the Best Paper Awards at prestigious IEEE-flagged conferences for some of his research work. He was a corecipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize in 2017, the 2009 IEEE ComSoc Asia–Pacific Best Young Researcher Award in 2009, the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation in 2008, the 2007 Funai Foundation Science Promotion Award in 2007, the 2006 IEEE Computer Society Japan Chapter Young Author Award in 2006, the Niwa Yasujirou Memorial Award in 2005, and the Young Researcher's Encouragement Award from the Japan Chapter of the IEEE Vehicular Technology Society in 2003. He is a member of the IEEE Communications Society Standardization Program Development Board. He is/was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE Wireless Communications Magazine, the IEEE JOURNAL ON INTERNET OF THINGS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals.

Victor C. M. Leung (S'75, M'89, SM'97, F'03) is a Distinguished Professor of Computer Science and Software Engineering at Shenzhen University. He is also an Emeritus Professor of Electrical and Computer Engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems at the University of British Columbia (UBC). His research is in the broad areas of wireless networks and mobile systems. He has co-authored more than 1300 journal/conference papers and book chapters. Dr. Leung is serving on the editorial boards of the IEEE Transactions on Green Communications and Networking, IEEE Transactions on Cloud Computing, IEEE Access, IEEE Network, and several other journals. He received the IEEE Vancouver Section Centennial Award, 2011 UBC Killam Research Prize, 2017 Canadian Award for Telecommunications Research, and 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award. He co-authored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, 2017 IEEE Systems Journal Best Paper Award, 2018 IEEE CSIM Best Journal Paper Award, and 2019 IEEE TCGCC Best Journal Paper Award. He is a Fellow of IEEE, the Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada. He is named in the current Clarivate Analytics list of Highly Cited Researchers.