CHINA COMMUNICATIONS

Deep Reinforcement Learning for Energy-efficient Edge Caching in Mobile Edge Networks

Deng Meng[1,4], Huan Zhou[1,2,4,*], Kai Jiang[5], Hantong Zheng[3,4], Yue Cao[5], Peng Chen[3,4]

[1] College of Economics and Management, China Three Gorges University, Yichang 443002, China
[2] College of Computer, Northwestern Polytechnical University, Xi'an 710000, China
[3] College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China
[4] The Hubei Key Laboratory of Intelligent Vision Based Monitoring for Hydroelectric Engineering (China Three Gorges University), Yichang 443002, China
[5] School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[*] The corresponding author, email: zhouhuan117@gmail.com

Cite as: D. Meng, H. Zhou, *et al.*, "Deep reinforcement learning for energy-efficient edge caching in mobile edge networks," *China Communications*, 2023. **DOI**: 10.23919/JCC.ea.YYYY-MMMM.NNNN

Abstract: Edge caching has emerged as a promising application paradigm in 5G networks, and by building edge networks to cache content, it can alleviate the traffic load brought about by the rapid growth of Internet of Things (IoT) services and applications. Due to the limitations of Edge Servers (ESs) and a large number of user demands, how to make the decision and utilize the resources of ESs are significant. In this paper, we aim to minimize the total system energy consumption in a heterogeneous network and formulate the content caching optimization problem as a Mixed Integer Non-Linear Programming (MINLP). To address the optimization problem, a Deep Q-Network (DQN)-based method is proposed to improve the overall performance of the system and reduce the backhaul traffic load. In addition, the DQN-based method can effectively solve the limitation of traditional reinforcement learning (RL) in complex scenarios. Simulation results show that the proposed DQN-based method can greatly outperform other benchmark methods, and significantly improve the cache hit rate and reduce the total system energy consumption in different scenarios.

Keywords: Edge Caching; Energy Consumption; Deep Reinforcement Learning; Markov Decision Process

Received:
Revised:
Published Online:
Editor:

I. INTRODUCTION

With the popularity of smart devices (such as smart phones and smart watches) and the growing demand of users for networks, especially for some multimedia services and applications, a large amount of cellular traffic will be generated every day, which is a serious problem for Mobile Users (MU) [1–4]. However, the current cloud-based centralized model has been unable to satisfy the Quality of Service (QoS) of large-scale content delivery [5, 6]. In fact, the additional energy consumption and communication delay of cloud processing have always been an urgent problem to be solved, which has attracted the extensive attention of many researchers [7, 8].

To efficiently deal with huge traffic requests and alleviate the pressure on cellular networks, edge caching technology has become a research hotspot in the industry. It makes the content closer to the user by using the Small Base Station (SBS) to provide the caching function, which will bring the following benefits: (1) reducing the traffic load and system energy consumption on the backhaul link; (2) dropping the delay for

users to obtain content; (3) saving network bandwidth; (4) alleviating network congestion. At the same time, recent studies have found that only a few popular content are frequently requested and the rest are rarely requested, which further accelerates the implementation of edge caching technology in mobile edge networks [9, 10]. Moreover, unlike macro base station (MBS) with rich and diverse resources, SBS has limited caching resources and can only cache a small part of popular content. Therefore, SBS's caching decisions for popular content will greatly enhance the performance of the system [11].

Meanwhile, with the rapid development of Deep Neural Network (DNN) technology, RL and Deep Reinforcement Learning are considered as suitable approaches to find approximate optimal solutions in dynamic environments [12]. RL does not rely on any prior information, it makes intelligent decisions by continuously interacting with the environment, and finally gets the best long-term goal. This feature makes it possible to apply RL for dynamic selection decisions under Mobile Edge Computing (MEC) environment [13, 14].

This paper focuses on the joint optimization of content caching and replacement decisions in edge caching. An edge caching system is first proposed, aiming to minimize the energy consumption, and the corresponding problem is formulated as Mixed Integer Non-Linear Programming (MINLP). To solve the problem, we propose a Deep Q-Network (DQN)-based method that can effectively improve cache utilization and overall system performance. In addition, the DQN-based method can efficiently solve the problem of the curse of dimensionality in complex environments.

The main contributions can be summarized as follows.

1. A three-layered edge caching architecture is proposed, in which the uncertain cache requirements, time-varying system conditions, and the bilateral synergy between SBSs are all considered.

2. We formulate the optimization problem as an MINLP problem, intending to minimize the total energy consumption of the system. In particular, the optimization process can be approximated as a Markov Decision Process (MDP), where the caching decision is determined by a centralized agent.

3. To address the MDP, we propose a DQN-based method to determine the optimal coordination of the caching decision, in which action-value functions are estimated with neural network-based parameter approximation.

The rest of this paper is organized as follows. Section II first reviews the related work. Then, Section III describes the system model. Next, In Section IV, the optimization problem is formulated as a MINLP to minimize the total energy consumption. In Section V, a DQN-based method is proposed. Furthermore, Section 6.1 performs simulations and analyzes its performance. Finally, Section VII concludes this paper.

II. RELATED WORK

2.1 Content Caching

In recent years, edge caching is a very important technology in MEC. The authors in [15, 16] employed mobility prediction to design optimal edge caching schemes. In [15], the authors described Wi-Cache, a prototype for caching-as-a-service and a software-defined networking (SDN)-based distributed content caching system that uses storage at the Access Points (APs) to cache content. The authors in [17] considered a three-tier architecture, including the smart device layer, edge cloud and blockchain-based distributed cloud. The authors in [18] proposed a cooperative caching scheme to reduce network delay and improve network throughput. Sun *et al.* in [19] considered two-hop edge caching in 6G networks, adopting blockchain and physical layer security technologies to prevent data from being tampered with and eavesdropping. Fang *et al.* in [20] presented a Mobile Edge Data Cooperative Cache Admission Based on Content Popularity (DCCCP) from the content provider's perspective to reduce system overhead. Zhou *et al.* in [21] consider the joint optimization of computation offloading and service caching in an edge computing-based smart grid, aiming to minimize the cost of the system. The authors in [22] first proposed a new Personalized Edge Caching System (PECS) architecture that utilizes big data analytics and mobile edge caching to provide personalized service access at the mobile edge network.

2.2 Deep Reinforcement Learning

In recent years, RL has been utilized to design efficient edge caching schemes in many researches. The authors in [23] utilized the RL methods to design a flexible edge caching strategy that aims to minimize the total cost by simultaneously considering energy consumption and latency. In [24], RL-based methods are also used to dynamically arrange edge computing and cache resources to improve the utility of the system. To effectively solve privacy issues in edge caches, the authors in [25] presented a privacy-preserving distributed deep deterministic policy gradient (P2D3PG) to maximize the cache hit rates of devices in the MEC networks. In [26], the authors proposed an RL-based self-adaptive edge caching architecture to reduce user access delay and background traffic burden. In order to improve cache utilization and reduce backhaul traffic load in edge vehicle network, Jiang *et al.* in [27] proposed several edge caching methods based on reinforcement learning. Anokye *et al.* in [28] proposed a user mobility model that utilizes human-centric features, random waypoints, and using Deep Reinforcement Learning to predict the location of the drone and what to cache on the drone, aiming to improve the Quality of Experience (QoE) for users.

To quickly converge to the optimal policy in the rapidly changing edge cache environment and avoid the cold start problem, Hu *et al.* in [29] proposed a transferable edge cache method based on A3C. The authors in in [30] presented an AC-based method in edge caching to minimize download latency for users. Ao *et al.* in [31] leveraged the idea of multi-point collaboration and design an efficient cooperative caching strategy to maximize the system throughput, while considering the available cache size and network topology. Saputra *et al.* in [32] introduced a novel MEC network structure that jointly optimizes content caching through cooperation among edge nodes. They then used an improved branch-and-bound algorithm and an interior-point approach to solve the optimization problem, aiming to find a joint caching and delivery strategy to reduce the average total latency across the network. To minimize the content access cost while satisfying the constraint on content delivery latency, Qiao *et al.* in [33] proposed a nature inspired method based on the deep deterministic policy gradient (DDPG) framework. the authors in [34] presented an A3C-based approach to jointly optimize computation offloading and service caching, aiming to minimize the task cost of the system.

Based on the above analysis, this paper considers the edge caching problem in a three-layer mobile heterogeneous network structure. In this structure, SBS pre-caches popular content in its available cache upon user request. When a mobile user initiates a content request, the user can directly obtain popular content from a nearby base station without downloading the content from a remote server. In addition, the SBSs are connected and can cooperate with each other to cache. Therefore, designing an effective caching strategy can not only improve the utilization of the cache, but also reduce the traffic load of the edge caching system. To solve the optimization problem, a DQN-based algorithm is proposed, and the SBS is used as the agent for training to quickly obtain the long-term optimal policy and reward.

III. SYSTEM MODEL

This section first presents the system model, and then introduces the communication model of the content delivery process.

3.1 Network Architecture

As shown in Figure 1, we investigate an edge caching system in a heterogeneous network and consider a three-layer structure including one MBS, several SBSs, and multiple MUs. The MBS can communicate with all SBSs through wired connections, and SBSs can communicate and cooperate to realize edge caching. At the same time, MUs connect to the SBSs through wireless links.

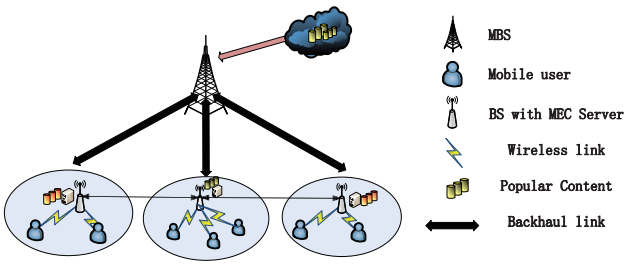


Figure 1. *System Architecture of Edge Caching.*

In addition, we use $\mathcal{T} = \{0, 1, 2, ..., t, ..., T\}$ to represent the time series of processing requests, MBS is

Table 1. *System Variable and Explanation*

Variable	Explanation
$\mathcal{I}$	The set of SBSs
$\mathcal{U}$	The set of MUs
$\mathcal{F}$	The set of Contents
C_i	The caching capacity of the ES i
P_f	The popularity of content f
$R_{i,u}(t)$	The transmission rate between MU_u and SBS_i
P_0	The transmission power of MBS
P_i	The transmission power of SBS_i
$a^t_{f,j,0}$	The transmission between MBS and of SBS_j
$a^t_{f,i,j}$	The transmission between SBS_j and SBS_i
$r^t_{f,i,i}$	The content replacement in SBS_i
s_f	The size of content f

represented by index 0, $\mathcal{I} = \{1, 2, ..., J\}$ means the set of SBSs, $\mathcal{C}$ denotes the cache capacity of SBS, $\mathcal{U} = \{1, 2, 3, ..., U\}$ represents the set of MUs, and $\mathcal{F} = \{1, 2, 3, ..., F\}$ represents the set of content that MUs can request. More system variables are shown in Table 1.

Meanwhile, we assume that the content requested by each user obeys the Zipf distribution, which is a well-known content popularity model [35], and the accuracy of Zipf's law distribution has been verified experimentally [36]. Therefore, it is easy to obtain the popularity P_f of the f-th content as:

$$P_f = \frac{(R_f + \tau)^{-\beta}}{\sum_{i \in \mathcal{F}} (R_i + \tau)^{-\beta}}, \forall f \in \mathcal{F} \,, \tag{1}$$

where R_f presents popularity index of content f in a descending order, τ and β denote the plateau factor and skewness factor, respectively.

3.2 Communication Model

In the process of communication, the type of transmission include MU to SBS, SBS to SBS, and MBS to SBS, and the u-th user is denoted as MU_u ($u \in \{1, 2, 3, ..., U\}$), the i-th SBS is denoted as SBS_i ($i \in \{1, 2, 3, ..., I\}$).

1) Mobile User to Edge Server: When MUs communicate with SBSs over a wireless link, similar to [37], we disregard the user's request delay and only assess the content delivery delay caused by SBS. Furthermore, we assume that each SBS contains numerous subchannels with the same bandwidth assigned to each channel. Therefore, the communication rate between the local SBS_i and MU_u at time slot t can be expressed by:

$$R_{i,u}(t) = w_{i,u} \log_2 \left(1 + \frac{q_i g_{i,u}}{\sigma^2 + \sum_{v \in U \setminus \{u\}: a_v = a_u} q_v g_{v,i}}\right) , \tag{2}$$

where $w_{i,u}$ indicates the bandwidth allocated to MU_u by SBS_i and $\sum_{u=1}^{U} w_{i,u} \leq W$, W denotes the bandwidth of SBS_i. q_i is the communication power from SBS_i to MU_u, $g_{i,u}$ presents the channel gain between SBS_i and the MU_u, σ^2 denotes the power of background noise, and $g_{i,u}$ is only related to the distance between SBS_i and MU_u.

According to Eq. (2), we can obtain the data transmission delay from the local SBS_i to MU_u at time slot t as:

$$D_{f,i,u}(t) = \frac{s_f}{R_{i,u}(t)} \,, \tag{3}$$

where $s_f (f \in \mathcal{F})$ denotes the size of content f.

Considering the QoE of the MUs, the request delay of MU_u cannot exceed the maximum tolerable delay σ_u^{max}.

When the local SBS_i stores the content f required by MU_u at time slot t, $a^t_{f,i,j} = 1$ $(i = j)$, which means that MU_u can directly obtain the relevant content from the local SBS_i through the wireless link. At this time, the communication energy consumption generated at time slot t can be calculated by:

$$E^{local}_{f,u}(t) = q_i \cdot D_{f,i,u}(t) \,, \tag{4}$$

where q_i means the communication power of the SBS_i.

Moreover, MUs will also produce energy consumption when requesting and receiving content from the local SBS. Since the energy consumption generated by request is negligible and can be ignored, we use $E^{rece}_{f,u}$ to describe the energy consumption when MUs receive content.

When MU_u receives content f from the local SBS_i, similar to [38], the receiving power can be expressed by:

$$p_{f,u}^{receive}(t) = \frac{\lambda \cdot q_i}{d_{i\to u}^2} , \tag{5}$$

where λ denotes a constant (path loss factor), $d_{i\to u}^2$ represents the distance between SBS_i and MU_u. Therefore, the receiving energy consumption of MU_u at time slot t can be defined as:

$$\begin{aligned} E_{f,u}^{rece}(t) &= p_{u,f}^{receive}(t) \cdot D_{f,i,u}(t) \\ &= \frac{\lambda \cdot q_i}{d_{i\to u}^2} \cdot D_{f,i,u}(t) . \end{aligned} \tag{6}$$

2) Edge Server to Edge Server / the Cloud Server: Firstly, when the content f is stored in SBS_j, here $a_{f,i,j}^t = 1$ and $i \neq j$, $j \neq 0$, which means that the content f is stored in the collaborative SBS_j. Therefore, the SBS_i first needs to obtain the corresponding content f from the cooperative SBS_j, and then transmit the content f to the designated MU_u. And the energy consumption from SBS_j to SBS_i can be calculated by:

$$E_{f,u}^{co}(t) = P_j \cdot s_f , \tag{7}$$

where P_j represents the communication power of each bit from SBS_i to SBS_j. Similar to [39], we set $P_j = 1 \times 10^{-8}(J/bit)$.

Then, when the content f exists in MBS, now $a_{f,i,0}^t = 1$, which indicates that MU_u cannot obtain the content f from the local SBS_i or the cooperative SBS_j. At this time, MU_u needs to receive the content f from the MBS. The local SBS_i first obtains the content f from the MBS, and then transmits f to MU_u. The energy consumption from MBS to SBS_i can be be expressed by:

$$E_{f,u}^{MBS}(t) = P_0 \cdot s_f , \tag{8}$$

where P_0 means the communication power of each bit from MBS to SBS_i.

In summary, when MU_u makes a content request, according to the data transmission process, the communication energy consumption of the system at time slot t is:

$$\begin{aligned} E_{f,u}^{tran}(t) =& E_{f,u}^{local}(t) + a_{f,i,j}^t \cdot E_{f,u}^{co}(t) \\ &+ a_{f,i,0}^t \cdot E_{f,u}^{MBS}(t) . \end{aligned} \tag{9}$$

3.3 Replacement Model

In addition to content caching, SBS also needs to implement a replacement strategy for the cache considering the more efficient use of the system's cache and the limitation of cache capacity. When a user makes a content request, if the required content is not stored in the local cache SBS, it must be requested from the collaborative SBS or MBS. In addition, to improve the utilization and effectiveness of the cache system, it is also necessary to adopt a corresponding strategy to replace the cached content, store the currently requested content in the cache, and update the cache.

Here, we define, $\boldsymbol{r_{\mathcal{F},i}^t} = \{r_{1,i}^t, r_{2,i}^t, ..., r_{F,i}^t\}$, where $r_{f,i}^t \in \{0, 1\}$ means whether the content in local SBS_i should be replaced by the current content f.

When the local SBS_i needs to perform cache replacement, the SBS_i first obtains the specified content f from the MBS and then uses the replacement strategy to store the content f in the cache, and the corresponding communication energy consumption can be calculated by: $E_f^{rep}(t) = P_0 \cdot s_f$.

Through the previous analysis, considering the content requests of MUs, the total system energy consumption of the entire system in time slot t can be expressed as:

$$\begin{aligned} E_{total}(t) = \sum_{f=1}^{F} \sum_{i=0}^{I} \Big(& E_{f,u}^{tran}(t) + E_{f,u}^{rece}(t) \\ & + r_{f,i}^t \cdot E_f^{rep}(t) \Big) . \end{aligned} \tag{10}$$

IV. PROBLEM FORMULATION

This section studies the joint optimization problem in edge caching. SBS can collaboratively implement edge caching to improve the effectiveness of the entire caching system. By jointly optimizing the SBSs to minimize the total energy consumption of the system, the optimization problem is formulated as:

$$\min \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E_{total}(t) \tag{11}$$

$$
\begin{aligned}
s.t. \quad & C1: a^t_{f,i,j} \in \{0,1\}, \quad \forall f, \forall i, \forall j \\
& C2: r^t_{f,i} \in \{0,1\}, \quad \forall f, \forall i \\
& C3: \sum_{j=0}^{N} a^t_{f,i,j} = 1, \quad \forall f, \forall i, \forall j \\
& C4: \sum_{f=1}^{F} c^t_{f,i} \cdot s_f \le C_i, \quad \forall i \\
& C5: D_{f,i,u}(t) \le \sigma_u^{\max} .
\end{aligned} \tag{12}
$$

Here, constraint C1 represents the types of transmission, constraint C2 represents the replacement decision of SBS, and constraint C3 means that the content requested by the user can only be one of the local SBS, cooperative SBS, and MBS. Constraint C4 represents the limitation of the cache capacity to ensure the stored content will not exceed the cache capacity and limit the maximum delay. Constraint C5 represents the maximum delay that the user can tolerate.

To solve the optimization problem in Eq. (11), it is necessary to find the optimal caching decision variables. However, since the state and actions of the system change dynamically, each SBS could not collect the massive state information of the caching system in real-time and make the content replacement strategy according to the current state. Therefore, the proposed problem of minimizing energy consumption is NP-hard [21, 32]. Furthermore, due to the non-convex nature of the feasible set, it is hard to optimize our objective with enumeration or traditional methods.

V. PROPOSED DQN-BASED ALGORITHM

This section introduces a DQN-based method to solve the above optimization problem.

5.1 State, Action and Reward Definition

During the solution process, the optimization problem is formulated as an MDP. The state space, action space, and reward function for the problem are defined as follows.

1. **State space (S):** In the system, we assume that before users start to request content, at time slot t, SBSs have an initial state $\boldsymbol{S_t} = \{\boldsymbol{z^t_{\mathcal{F},\mathcal{I}}}, \boldsymbol{c^t_{\mathcal{F},\mathcal{I}}}\}$. For the arrived content request state, we define $F \times I$ matrix $\boldsymbol{z^t_{\mathcal{F},\mathcal{I}}}$, where each element $z^t_{f,i}$ represents the request state of content f at SBS_i and $z^t_{f,i} \in \{0,1\}$. Here, $z^t_{f,i} = 1$ means that at least one UE within local SBS_i requests for content f in time slot t, $z^t_{f,i} = 0$ is the opposite. Similarly, for the cache state, we define $F \times I$ matrix $\boldsymbol{c^t_{\mathcal{F},\mathcal{I}}}$, where each element $c^t_{f,i}$ represents the cache state of content f in SBS_i and $c^t_{f,i} \in \{0,1\}$.
2. **Action space (A):** When MUs start to request from the SBSs, SBSs need to perform a series of operations, according to the current cache state in SBSs, the content f should be sent to the MU_u. At this time, the action space of SBSs are $\boldsymbol{A_t} = \left\{\boldsymbol{a^t_{\mathcal{F},\mathcal{I}}}, \boldsymbol{r^t_{\mathcal{F},\mathcal{I}}}\right\}$, where $F \times I$ matrix $\boldsymbol{a^t_{\mathcal{F},i,j}}(i \in \mathcal{I}, j \in \mathcal{I}, i \neq j)$ represents the processing action respecting to various contents in each SBS, and $F \times I$ matrix $\boldsymbol{r^t_{\mathcal{F},\mathcal{I}}}$ indicates the content replacement control in each SBS. Here, element $r^t_{f,i} \in \{0,1\}$ indicates whether and which content in SBS_i should be replaced by the current content.
3. **Reward function:** According to the cached state $\boldsymbol{S_t}$, when MUs request content, SBSs take the corresponding action $\boldsymbol{A_t}$ and obtain different reward values R_t according to different results, and transfers to the next state $\boldsymbol{S_{t+1}}$. To minimize the total energy consumption of the system and guarantee that the result converges to an ideal value, a negative exponential function is used as a long-term reward, and the R_t can be written as:

$$
R(\boldsymbol{S_t}, \boldsymbol{A_t}) = e^{-E_{total}(t)} . \tag{13}
$$

5.2 Markov Decision Process

MDP is often used to describe reinforcement learning, the process of stochastic control of discrete-time states and actions. Meanwhile it can solve many problems related to stochastic control. Usually, in linear programming or dynamic programming, it is an efficient solution. However, it is difficult for MDP to deal with these problems when it is difficult to determine the change of transfer probability and immediate reward over time.

For the above problem, RL is an effective solution, and the agent can continuously interact with the environment, learn from experience, and take the action with the greatest probability to obtain the maximum cumulative expectation.

The ultimate goal of the agent is to find an optimal

strategy $\pi^*(s_t) = a_t^* \in \boldsymbol{A}$. For the MDP method, under the conditions of any initial state, the agent can obtain a long-term expectation through s_t and the optimal strategy π, and the expected value $V^\pi(s_t)$ can be expressed as:

$$V^\pi(s_t) = \mathbb{E}_\pi \left[\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{\infty} \alpha^{t-1} \cdot R(s_t, a_t) | s = s_1 \right]. \tag{14}$$

Among them, $\alpha \in (0, 1]$ is the discount factor. it means that the current reward value has a greater impact on the entire state value function, while the later reward value has less impact. the state-value function is more focused on immediate rewards.

To maximize long-term returns, each SBS needs to adaptively learn the optimal strategy, and maximize the system average expectation:

$$R_{long} = \max V^\pi(s_t). \tag{15}$$

Under the initial conditions that the state s_t and the action a_t have been determined. From the Bellman equation, we can get that s_{t+1} passes through s_t, then s_{t+1} will also satisfy the initial conditions. The optimal state value function can be expressed as:

$$V^\pi(s_t) = R(s_t, a_t) + \alpha \cdot \sum_{s_{t+1}} Pr\{s_{t+1}|s_t, a_t\} \cdot V(s_{t+1}). \tag{16}$$

To sum up, the RL agent learns an optimal policy according to the environment, and its goal is to obtain a long-term cumulative reward value. Therefore, the optimization problem can be transformed into a function that seeks the optimal state value, expressed as:

$$\begin{aligned} V^{\pi^*}(s_t) = &\max_\pi [R(s_t, a_t) \\ &+ \alpha \cdot \sum_{s_{t+1}} Prs_{t_1}|s_t, a_t \cdot V(s_{t+1})]. \end{aligned} \tag{17}$$

The optimal control strategy is equivalent to maximizing our state-value function:

$$a^* = arg \max_\pi V^\pi(s_t, a_t). \tag{18}$$

5.3 DQN-based Edge Caching Method

RL is different from supervised or unsupervised learning, and it maximizes long-term rewards by learning and making optimal decisions about the state of the network. As a typical representative of reinforcement learning, Q-Learning can obtain the optimal policy by continuously recording and updating the Q value in the Q table. However, Due to the complex state and action space of the problem, traditional RL methods can no longer meet the requirements.

In order to solve the above problems, we propose a DQN-based method that utilizes DNN to measure the action-value function of RL. Compared with Q-learning, the DQN method has shown excellent performance in solving complex dynamic MDP problems.

Figure 2 shows the training process of DQN. The environment will first give an initial state s_t, then, according to the strategy π and s_t, the agent will get the optimal action a_t, and finally obtain the corresponding reward value and next state s_{t+1}. At the same time, in order to improve the sample efficiency and speed up the convergence, DQN stores information (s_t, a_t, r_t, s_{t+1}) in the Experience Replay Buffer. Finally, during training in each iteration, the DQN network extracts data from the Experience Replay Buffer and updates the network parameters by minimizing loss value between Target-Net and Main-Net. With the continuous iteration of the network, the optimal neural network parameters can be finally obtained.

In the following, we propose a DQN-based algorithm for content caching, aiming to address the proposed MDP. According to the environment state s_t, the action a_t can be given by:

$$a_t = \begin{cases} a, & \epsilon \\ arg \max_a Q(s_t, a_t), & 1 - \epsilon \end{cases}. \tag{19}$$

Therefore, according to Eq. (19), the the value function is defined as:

$$Q(s_t, a_t) = r_t + \gamma \max_{a_t} Q(s_t, a_t), \tag{20}$$

where $\gamma \in$ [0,1] denotes the discount factor. In addition, we utilize the DNN as an approximator of the Q-value $Q(s, a)$, the $Q(s, a)$ in DQN can be expressed as:

$$Q(s_t, a_t) \approx Q(s_t, a_t; \theta), \tag{21}$$

where θ is the weight of the DNN.

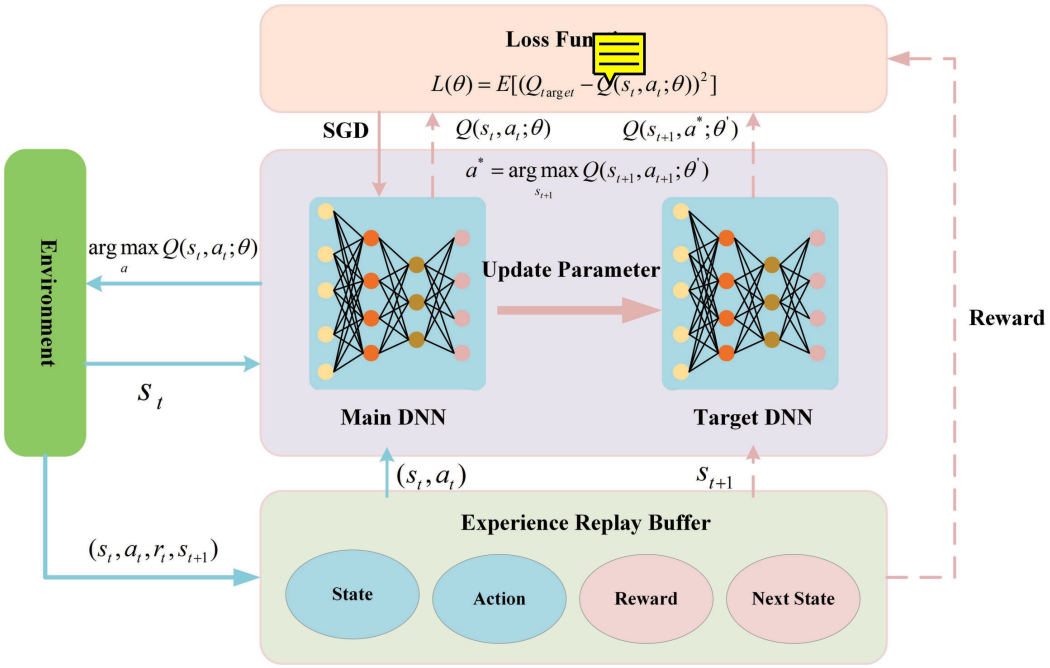


Figure 2. *The training process of DQN.*

According to the current state s_t, the agent can adopt the optimization strategy to get the caching strategy a_t, next, the reward value r_t, and the next state s_{t+1} can be obtained by a_t. Then, the Q_{target} can be expressed by:

$$Q_{target} = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta') \,, \tag{22}$$

where $\gamma \in [0,1]$ denotes the discount factor, θ represents the DNN parameters.

According to Eq. (21) and Eq. (22), the loss function L of the DQN network can be given by:

$$L(\theta) = E[(Q_{target} - Q(s_t, a_t; \theta))^2]. \tag{23}$$

After the loss function value is obtained, the stochastic gradient descent algorithm is used to obtain the parameters of the network model. The value of θ can be calculated by:

$$\theta = \theta - \alpha \frac{\partial L(\theta)}{\partial \theta} \,. \tag{24}$$

where $\alpha = 0.1$ denotes the learning rate.

Finally, we save the trained network model parameters into the Target-Net.

The details of the DQN-based algorithm are shown in Algorithm 1. First is the initialization of the environment (Lines 1-4). The iteration based on the DQN algorithm will be run until the end (Line 5). In each episode, the environment will initialize the state s_t (Line 6). According to the s_t, the DQN agent selects the action a_t to obtain the reward value r_t and the next state s_{t+1}. Then, we can calculate the energy consumption cost of the system and store the samples (s_t, a_t, r_t, s_{t+1}) in the Experience Replay Buffer (Lines 8-16). When the Experience Replay Buffer is full, the NN starts training. By extracting data from the Replay Buffer, Main Net and Target Net calculate $Q(s_t, a_t; \theta)$ and Q_{target} respectively to minimize the loss function. Finally, at the end of the iteration, the parameters will be updated by the Stochastic Gradient Descent algorithm (SGD) (Lines 17-22).

In Algorithm 1. the complexity depends on $|E|$ and the number of $|T|$. For the outer loop, its time complexity is $O(|E|)$. For the inner loop, SBS will determine its cache decision and get a reward, the complexity is $O(|T|)$. Therefore, the complexity of the DQN algorithm is $O(|E||T|)$.

Algorithm 1. *DQN-based Algorithm for Content Caching.*

Set action A
Initialize the Experience Replay Buffer with the fixed capacity $|D|$
Set the Main-Net with weights θ
Set the Target-Net with weights $\theta^{'} = \theta$
for $episode = 1, 2, 3, ..., E$ **do**
  Observe the initial status s_1 of the system environment
  for $t = 1, 2, 3, ..., T$ **do**
    Choose an action with a probability Φ in the current s_t via ϵ-greedy policy as
    if $\Phi \leqslant \epsilon$ **then**
      randomly select an action a_t
    else
      select $a_t = argmax_a Q(s_t, a_t)$
    end if
    Execute action a_t, observe the reward R_t and the next state s_{t+1}
    Evaluate the energy cost via Eq. (10)
    Store the experience (s_t, a_t, r_t, s_{t+1}) into the Experience Replay Buffer
    Sample random minibatch of transitions from experience replay buffer
    Calculate the target Q value via Eq. (22)
    Update the weights θ of network Q via Eq. (24) by utilizing SGD algorithm
    Every C steps: $\theta^{'} \leftarrow \theta$
    Let $s_t \leftarrow s_{t+1}$
  end for
end for

VI. NUMERICAL RESULTS

In this section, simulation results are presented to verify the performance of the DQN-based method and compares it with other benchmarks.

6.1 Simulation Settings

In the simulation, we consider an MBS, 4 SBSs, and several MUs, the MBS can communicate with all SBSs, and the users are randomly distributed under the SBSs. The communication power of the MBS is 20W, and the communication power of the SBS is 5W. Similar to [40], we use $g_{n,u} = 127 + 30 \cdot \log_{10}(L)$ as the channel power gain, here L represents the distance between MUs and SBSs. In addition, we use 3000 contents for experimental simulation, the size of the content is in the range of 3-8 Mbit, and the content popularity obeys the Zipf distribution with parameter $\beta = 0.45$. Finally, We use $C = 100$ MB as the cache capacity. Table 2 shows the detailed parameter settings. For the proposed DQN-based methods, we use $\varphi = 0.1$ as the learning rate and $\alpha = 0.9$ as the discount factor. It should be noted that in each slot t, all request processing time cannot exceed the maximum tolerable delay.

Table 2. *Simulation parameters and assumptions.*

Parameter	Definition	Value
F	The number of Content	3000
s_f / (Mbit)	The size of each content	[3, 8]
W/(MHz)	The transmission bandwidth	20
C /(MB)	The cache capacity of each SBS	100
$P_1/(J \cdot bit)$	The Energy consumption between SBS and SBS	1×10^-8
$P_2/(J \cdot bit)$	The energy consumption between SBS and MBS	2×10^-8
$\sigma^2/(dBm)$	The power of White Gaussian Noise	-95
φ	Exploration probability	0.1
α	The discount factor	0.9
τ	The plateau factor	-0.34
β	The skewness factor	0.45

6.2 Evaluation Results

In this part, we evaluate the performance of the DQN-based method by comparing them with four other benchmark methods, which are presented as follows:

1. Q-learning: Q-learning is an efficient reinforcement learning algorithm to solve part of dynamic decision process problems.
2. FIFO: Replace the first cached content first. When the SBS cache is full, it needs to replace the first stored content in the cache with the new content.
3. LFU: Replace the least used content first. When the SBS cache is full, it will replace the longest unused content with new content in the SBS.
4. LRU: Replace the least recently used content first. When the cache capacity of the SBS is full and the SBS needs to replace content, it will replace the content with the least number of requests in a certain period with new content in the SBS.

Figure 3 depicts the convergence performance of the

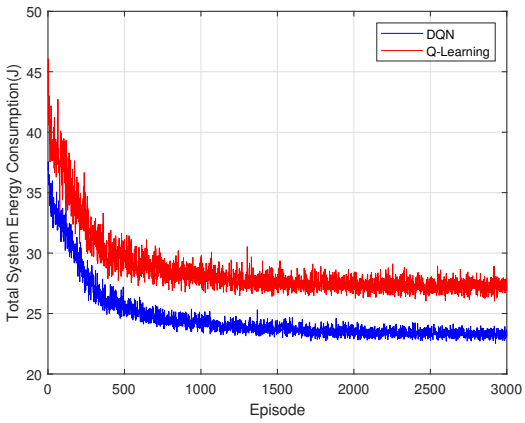


Figure 3. *Convergence of the proposed DQN-based method.*

proposed DQN-based method. From Figure 3, we can observe that in both RL-based methods, the total system energy consumption per episode decreases rapidly as the training set increases, and with continuous training, both methods can successfully learn effective content caching strategy. However, the DQN-based converges faster than the Q-learning-based method, reaching convergence at around 800 episodes. Furthermore, although the energy consumption of both methods is decreasing, the Q-learning-based is consistently higher than that based on DQN in most episodes, implying that the DQN-based is more efficient in the content replacement process.

Figure 4 compares the cache hit ratios of the five methods in different scenarios (i.e., different number of users, cache capacity, and Zipf parameter).

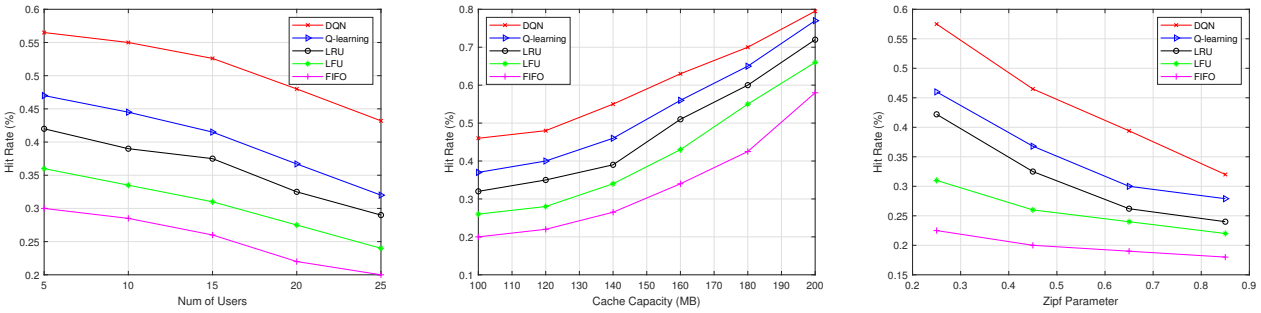
As can be seen from the Figure 4a, When the user is fixed, the methods based on DQN and Q-learning have better performance, and the proposed DQN-based method has the highest cache hit rate, this result fully shows that the DQN-based method can better improve the utilization of the cache. With the increase of users, the edge caching methods based on DQN and Q-Learning proposed in this paper always shows better performance than other baseline methods in terms of cache hit ratio. In addition, the cache hit ratio of all caching methods tends to decrease as the number of users increases. In this case, the increase in the number of users makes them request more popular contents, resulting in more requests that are not processed on SBSs.

Figure 4b represents the performance comparison of the five methods in terms of the cache hit ratio when the cache capacity of SBS is different. First, when the cache capacity is fixed, the methods based on DQN and Q-learning have better effects, and the DQN-based method has the best performance. Then, the cache hit ratios of all algorithms increase when the cache capacity keeps increasing, these results illustrate that a larger cache capacity can store more content with higher hit ratios. In addition, when the cache capacity increases to a range, the cache hit rates of the five methods gradually approach, which shows that the impact of the replacement strategy on the hit rate is greatly reduced as the cache capacity increases.

Similarly, the Zipf parameter also has an impact on cache hit ratios. Figure 4c depicts the performance comparison of the five methods in terms of the cache hit ratios when the Zipf parameter is different. When the parameters are fixed, it can also be seen from the figure that the methods based on DQN and Q-learning have better performance. Next, we can observe that the cache hit ratio of all methods decreases as the Zipf parameter increases. The simulation results indicate the larger the Zipf parameter is, the more popular content will be requested by users, which also leads to more content requests not being processed on SBSs. Furthermore, when the Zipf parameter is gradually increased, the cache hit rate of all five methods decreases. It can be seen that the increase of the Zipf parameter has a greater impact on the methods based on DQN, Q-learning, and LRU.

Figure 5 displays the performance comparison of the five methods in terms of total system energy consumption as the number of users increases. The cache capacity of SBS is 100MB. First, when the number of users is fixed and small, the system energy consumption of the five methods is relatively close. Then, with the increase in the number of users, the energy consumption value of the five methods also increases gradually. But the energy consumption of the DQN-based and the Q-learning-based increases less, indicating that the DQN-based and the Q-learning-based proposed in this paper can make full use of the cache capacity. In addition, compared with Q-learning, the DQN-based method consumes less energy, which shows that with the increase of users, the calculation becomes more complicated, and the DQN-based has a greater advantage in training.

Furthermore, when the number of users is large, the



(a) (b) (c)

Figure 4. *(a) Edge hit rate versus the number of users. (b) Edge hit rate versus the cache capacity of SBSs. (c) Edge hit rate versus the Zipf Parameter.*

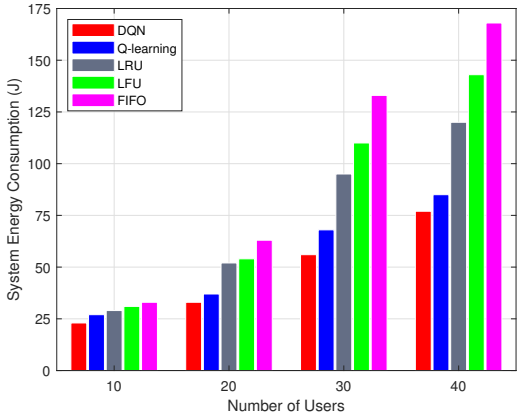


Figure 5. *Total system energy consumption with different number of users*

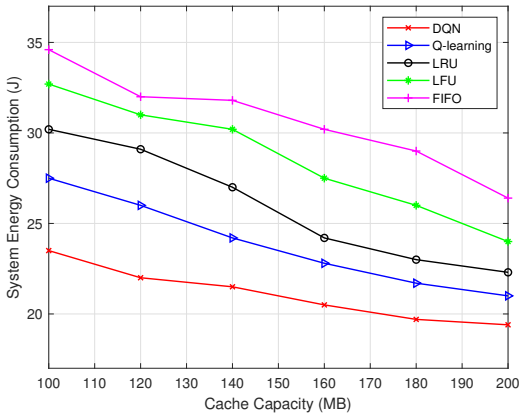


Figure 6. *Total system energy consumption with different cache capacities.*

energy consumption based on DQN and Q-learning is less compared to the other three methods. The experimental results show that the more the number of users, the higher the frequency of content replacement by SBS, and the method proposed in this paper can effectively improve cache utilization.

In the following, simulation results for the total system energy consumption of different methods under different SBS cache capacities are shown in Figure 6. the cache capacity of SBS changes from 100 to 200 (unit MB). When the cache capacity is fixed and small, increasing the cache capacity has a positive effect on the total energy consumption of the system. The smaller the cache capacity of the SBS, the less content is stored, and content replacement is more likely to occur, resulting in high system energy consumption.

In addition, as the cache capacity increases, the number of content replacements and energy consumption decrease. When the cache capacity is fixed, the proposed DQN-based and Q-learning-based methods consistently have less energy consumption compared to the three baseline methods, indicating that DQN-based and Q-learning-based method have more efficient caching strategies, can improve system utilization. Finally, as the SBS cache capacity increases, the energy consumption of the five methods gradually approaches. The experimental results show that with the increase of the cache capacity, the content stored in the SBS increases, the number of content replacements, and the energy consumption decreases.

VII. CONCLUSION

This paper investigated the joint optimization problem of content caching for three-layer heterogeneous networks. First, we transformed the optimization problem into MINLP, aiming to minimize the total energy consumption of the system. Then, the DQN-based method was proposed to solve the optimization problem. Finally, experiments showed that the proposed DQN-based method can effectively reduce the total energy consumption and improve the utilization rate of the system.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62172255; in part by the Outstanding Youth Program of Hubei Natural Science Foundation under Grant 2022CFA080, and the Wuhan AI Innovation Program (2022010702040056). Part of this work appeared in IEEE INFOCOM Workshop ICCN 2021 [41].

References

[1] K. Jiang, C. Sun, *et al.*, "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Network*, vol. 35, no. 5, pp. 74–82, 2021.

[2] H. Zhou, T. Wu, *et al.*, "Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 6144–6159, 2023.

[3] Z. Wang, K. Liu, *et al.*, "Attrleaks on the edge: Exploiting information leakage from privacy-preserving co-inference," *Chinese Journal of Electronics*, vol. 32, no. 1, pp. 1–12, 2023.

[4] H. Zhou, M. Li, *et al.*, "Accelerating deep learning inference via model parallelism and partial computation offloading," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 475–488, 2022.

[5] H. Zhou, Z. Wang, *et al.*, "Uav-aided computation offloading in mobile-edge computing networks: A stackelberg game approach," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6622–6633, 2022.

[6] S. He, K. Shi, *et al.*, "Collaborative sensing in internet of things:a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1435–1474, 2022.

[7] H. Zhang, M. Huang, *et al.*, "Capacity maximization in risuav networks: a ddqn-based trajectory and phase shift optimization approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2583–2591, 2022.

[8] H. Zhou, K. Jiang, *et al.*, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1517–1530, 2021.

[9] W. Jiang, G. Feng, *et al.*, "Multi-agent reinforcement learning for efficient content caching in mobile d2d networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1610–1622, 2019.

[10] H. Zhou, V. C. Leung, *et al.*, "Predicting temporal social contact patterns for data forwarding in opportunistic mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 372–10 383, 2017.

[11] Q. Li, X. Wang, *et al.*, "Mobility-aware caching strategy in an sdn-based cooperative caching network," *China Communications*, vol. 20, no. 9, pp. 196–214, 2023.

[12] E. Li, L. Zeng, *et al.*, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[13] X. Wang, Y. Han, *et al.*, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[14] H. Zhou, T. Wu, *et al.*, "Incentive-driven deep reinforcement learning for content caching and d2d offloading," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2445–2460, 2021.

[15] A. H. Ngu, M. Gutierrez, *et al.*, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2016.

[16] G. Yu, Z. Cai, *et al.*, "Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1294–1308, 2019.

[17] H. Wang, Y. Li, *et al.*, "An algorithm based on markov chain to improve edge cache hit ratio for blockchain-enabled iot," *China Communications*, vol. 17, no. 9, pp. 66–76, 2020.

[18] L. Guo, M. Dong, *et al.*, "A secure mechanism for big data collection in large scale internet of vehicle," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 601–610, 2017.

[19] W. Sun, S. Li, *et al.*, "Edge caching in blockchain empowered 6g," *China Communications*, vol. 18, no. 1, pp. 1–17, 2021.

[20] J. Fang, S. Chen, *et al.*, "Mobile edge data cooperative cache admission based on content popularity," in *2021 IEEE International Conference on Edge Computing (EDGE)*, pp. 111–118. IEEE, 2021.

[21] H. Zhou, Z. Zhang, *et al.*, "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1122–1132, 2023.

[22] M. Yan, W. Li, *et al.*, "Pecs: Towards personalized edge caching for future service-centric networks," *China Communications*, vol. 16, no. 8, pp. 93–106, 2019.

[23] A. Sadeghi, F. Sheikholeslami, *et al.*, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2267–2281, 2019.

[24] Y. Dai, D. Xu, *et al.*, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 12–18, 2019.

[25] S. Liu, C. Zheng, *et al.*, "Distributed reinforcement learning

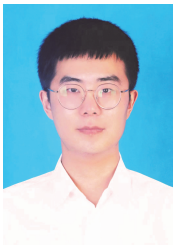
for privacy-preserving dynamic edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 749–760, 2022.
[26] D. Li, Y. Han, *et al.*, "Deep reinforcement learning for cooperative edge caching in future mobile networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6. IEEE, 2019.
[27] H. Zhou, K. Jiang, *et al.*, "Distributed deep multi-agent reinforcement learning for cooperative edge caching in internet-of-vehicles," *IEEE Transactions on Wireless Communications*, vol. PP, no. 99, pp. 1–1, 2023.
[28] S. Anokye, D. Ayepah-Mensah, *et al.*, "Deep reinforcement learning-based mobility-aware uav content caching and placement in mobile edge networks," *IEEE Systems Journal*, vol. 16, no. 1, pp. 275–286, 2021.
[29] L. Hu, S. Fan, *et al.*, "A transferable edge caching method based on reinforcement learning for dense small cell network," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6. IEEE, 2020.
[30] W. Jiang, D. Feng, *et al.*, "Proactive content caching based on actor–critic reinforcement learning for mobile edge networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 1239–1252, 2021.
[31] W. C. Ao and K. Psounis, "Fast content delivery via distributed caching and small cell cooperation," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1048–1061, 2017.
[32] Y. M. Saputra, D. T. Hoang, *et al.*, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 19–31, 2019.
[33] G. Qiao, S. Leng, *et al.*, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 247–257, 2019.
[34] H. Zhou, Z. Wang, *et al.*, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An a3c-based approach," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1326–1338, 2023.
[35] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on networking*, vol. 16, no. 6, pp. 1447–1460, 2008.
[36] X. Li, X. Wang, *et al.*, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1768–1785, 2018.
[37] X. Chen, L. Jiao, *et al.*, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 5, pp. 2795–2808, 2015.
[38] Y. An and X. Luo, "An in-network caching scheme based on energy efficiency for content-centric networks," *IEEE Access*, vol. 6, pp. 20 184–20 194, 2018.
[39] M. Yan, C. A. Chan, *et al.*, "Assessing the energy consumption of proactive mobile edge caching in wireless networks," *IEEE Access*, vol. 7, pp. 104 394–104 404, 2019.
[40] L. Li, Y. Xu, *et al.*, "Deep reinforcement learning approaches for content caching in cache-enabled d2d networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 544–557, 2019.
[41] H. Zheng, H. Zhou, *et al.*, "Reinforcement learning for energy-efficient edge caching in mobile edge networks," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6. IEEE, 2021.

Biographies

Deng Meng received the M.S. degree from China Three Gorges University, Yichang, China, in 2022. He is currently pursuing the Ph.D. degree in Management Science and Engineering at China Three Gorges University. His research interests are mobile edge computing, deep reinforcement learning, and edge caching.

Huan Zhou (M'14) received his Ph.D. degree from the Department of Control Science and Engineering at Zhejiang University. He was a visiting scholar at the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoc fellow at the University of British Columbia from Nov. 2016 to Nov. 2017. Currently, he is a professor at the College of Computer, Northwestern Polytechnical University. He was a Lead Guest Editor of Pervasive and Mobile Computing, TPC Chair of EAI BDTA 2020, Local Arrangement Chair of I-SPAN 2018, Special Session Chair of the 3rd International Conference on Internet of Vehicles (IOV 2016), and TPC member of IEEE Globecom, ICC, ICCCN, etc. He has published more than 50 research papers in some international journals and conferences, including IEEE JSAC, TPDS, TWC and so on. His research interests include Opportunistic Mobile Networks, VANETs, Mobile Data Offloading, and Mobile Edge Computing. He receives the Best Paper Award of I-SPAN 2014 and I-SPAN 2018, and is currently serving as an associate editor for IEEE ACCESS and EURASIP Journal on Wireless Communications and Networking.

Kai Jiang (S'20) received the M.S. degree from China Three Gorges University, China, in 2021. He is currently pursuing the Ph.D. degree in cyberspace security at Wuhan University, China. His research interests include Mobile Edge Computing, Deep Reinforcement Learning, and Internet of Vehicles.



Hantong Zheng received the M.S. degree from China Three Gorges University in 2022. His main research interests are mobile edge computing, deep reinforcement learning and edge caching.

Yue Cao (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the Institute for Communication Systems, University of Surrey, Guildford, U.K., in 2013.

He was a Research Fellow with the University of Surrey and an Academic Faculty with Northumbria University, Newcastle upon Tyne, U.K., Lancaster University, Lancaster, U.K., and Beihang University, Beijing, China. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His research interests include intelligent transport systems, including E-mobility, vehicle-to-everything, and edge computing.

Peng Chen received the Ph.D. degree in system analysis and integration from the Huazhong University of Science and Technology. Now, he has been a Professor with Computer and Information Institute of China Three Gorges University. His research interests include Artificial Intelligence, Big Data.