IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 December 2023, accepted 22 January 2024, date of publication 13 February 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365578

RESEARCH ARTICLE

Deep Q-Learning Aided Energy-Efficient Caching and Transmission for Adaptive Bitrate Video Streaming Over Dynamic Cellular Networks

JUNFENG XIE
School of Information and Communication Engineering, North University of China, Taiyuan 030051, China
e-mail: xiejunfeng@nuc.edu.cn

This work was supported in part by the Research Project supported by the Shanxi Scholarship Council of China under Grant 2022-147, and in part by the Research Project supported by the Fundamental Research Program of Shanxi Province under Grant 202203021212151 and Grant 202203021221117.

ABSTRACT Adaptive bitrate video streaming (ABRVS) and edge caching are two techniques that hold the potential to improve user-perceived video viewing experience. In this paper, we investigate the content caching, transcoding and transmission for ABRVS in cache-enabled cellular networks. Considering the dynamic characteristics of video popularity distribution and wireless network environment, to improve energy efficiency and minimize system energy consumption, we begin by formulating a long-term optimization problem that focuses on both video caching and user association (UA). The problem is then transformed into a Markov decision process (MDP), which is solved by designing a deep Q-learning network (DQN)-based algorithm. Using this algorithm, we can obtain the optimal video caching and UA solutions. Since the action space of the MDP is huge, to cope with the ''curse of dimensionality'', linear approximation is integrated into the designed algorithm. Finally, the proposed algorithm's convergence and effectiveness in reducing long-term system energy consumption are demonstrated through extensive simulations.

INDEX TERMS Adaptive bitrate video streaming, edge caching, energy efficiency, deep Q-learning.

I. INTRODUCTION

In recent years, driven by mobile network technologies' rapid advance and smart devices' popularization, mobile data traffic is experiencing an explosive growth. According to the report from Ericsson [1], the total global mobile data traffic will reach 325EB every month in 2028, which is nearly four times of 2022. Meanwhile, the unprecedented increase of multimedia applications results in video service being one of the most popular services. It is estimated that video traffic will account for 80 percent of all mobile data traffic in 2028 [1]. Thus, enhancing user-perceived video viewing experience in cellular networks becomes very important.

Edge caching [2], [3], [4] and adaptive bitrate video streaming (ABRVS) are two techniques that hold the potential to improve user-perceived video viewing experience. Edge caching brings videos much closer to users by caching a part of high-popular videos at the mobile network edge (e.g., base station (BS)) in advance during off-peak hours. Once the video requested by a mobile user has already been cached, it will be transmitted to the user directly, thereby reducing the end-to-end content delivery latency, mitigating duplicate content transmissions, saving backhaul resources, alleviating network congestions, as well as improving mobile users' quality of experience (QoE).

Considering the heterogeneity of mobile users' devices and the time-varying wireless channel conditions, ABRVS [5], [6], [7] has been proposed. ABRVS divides a video into small chunks and encodes each chunk into multiple versions with different resolutions and bitrates. This allows ABRVS to adjust the video quality in real-time based on users' preferences and network conditions, providing an adaptive and seamless viewing experience to mobile users. For example, when a mobile user's device is highly capable and the wireless network condition is good, he can receive high quality videos, whereas the mobile user can receive

The associate editor coordinating the review of this manuscript and approving it for publication was Stefan Schwarz.

low quality videos when the wireless network condition is poor.

Based on the above analysis, combining edge caching and ABRVS can improve user-perceived video viewing experience. However, it is particularly challenging due to the following two major reasons. First, with ABRVS, both different videos and different quality versions of the same video will compete for edge nodes' limited caching capacity. Second, if a mobile user requests a certain quality version of a video that has not been cached, but a higher quality version of the same video has already been cached, the edge node will first transcode the cached higher quality version to the requested quality version before transmitting it to the mobile user.

In general, video delivery in cache-enabled cellular networks consists of video caching and video transmission. For video caching, BSs or other mobile edge nodes prefetch high-popular videos and cache them in advance. Considering the conflict between mobile edge nodes' limited caching capacity and the massive number of videos, mobile edge nodes can only cache a part of videos. Therefore, in order to satisfy as many users' requests as possible, it is a critical problem to make caching policy decision to select the proper videos for each mobile edge node to cache. For video transmission, considering the densification trend of radio access networks (RANs), it is high-probability for mobile users to be covered by multiple edge nodes [8], [9]. Therefore, it is a critical problem to make user association (UA) strategy decision to obtain the appropriate association relationship between mobile users and edge nodes. The UA strategy in cache-enabled cellular networks should consider not only the wireless channel conditions, but also the mobile users' requirements and edge nodes' cache status. For example, to reduce content delivery latency, the mobile user may prefer to associate with the edge node that has cached the requested video instead of the edge node with the best wireless channel condition. Apparently, caching policy and UA strategy are naturally coupled. In cache-enabled cellular networks, caching policy strongly depends on the video popularity distribution (VPD), while UA strategy is largely affected by the wireless channel state information (WCSI). Considering the dynamic time-varying characteristics of VPD and WCSI, as well as the strong coupled relationship between the video caching stage and the video transmission stage, the caching policy and UA strategy should be optimized jointly. As a result, it is of great significance to design efficient caching policy and UA strategy for ABRVS in dynamic cache-enabled cellular networks.

A. RELATED WORKS

Due to the potential to enhance video delivery efficiency and mobile users' QoE, cache-enabled ABRVS has been widely investigated. Literature [10] proposes a two-step approach to optimize the transcoding configuration and caching space allocation. In [11], the authors propose a JCCPA algorithm to minimize the delivery delay and energy consumption by optimizing caching, computing and power allocation. The authors in [12] optimize the two time-scale video caching and transmission to maximize the backhaul saving and video quality. In [13], to improve the system energy efficiency, video caching and scheduling are optimized jointly. The authors in [6] optimize the video caching and processing to minimize the expected delay cost. Literature [14] exploits a cooperative transfer learning-based algorithm to balance the content quality and hit ratio. In order to improve the ABRVS services, a flexible transcoding strategy is presented in [15]. However, these works studied in static scenarios, whereas the dynamic characteristics of the system states in practical scenarios were largely ignored. Considering the time-varying VPD and WCSI, the optimal system performance at a certain time slot cannot guarantee the optimal system performance over a long time period.

There have been studies that specifically focus on video caching and transmission in dynamic scenarios. In [16], the cache hit rate is maximized by utilizing a DQN-based content caching algorithm. Literature [17] proposes a learning-based algorithm to predict the future content popularity and optimize the edge caching policy. The authors in [18] focus on the two time-scale caching placement and UA problem, and propose a BP-based UA algorithm and DDPG-based caching placement algorithm. Literature [19] uses Q-learning algorithm to optimize caching placement and resource allocation. Literature [20] adopts the Stackelberg game to optimize UA, power allocation of non-orthogonal multiple access (NOMA), unmanned aerial vehicle (UAV) deployment and caching placement to minimize the content delivery delay. In [21], the authors improve the content caching and sharing of D2D networks by a CAQL-based caching placement algorithm. Taking into account Coordinated MultiPoint (CoMP) joint transmission technique, a reinforcement learning (RL)-based algorithm is presented in [22] to maximize the delay reduction. However, these works just considered multiple single videos, whereas the video contents with multiple quality versions and the video transcoding between different quality versions were not taken into account.

Recently, as artificial intelligence (AI) and machine learning algorithms continue to advance rapidly, many researchers have studied on the integration of intelligence technology and wireless communication system optimization [23]. RL-based algorithms, as one critical category of AI algorithms, have been widely used in many domains, such as blockchain, edge caching, computation offloading and resource allocation. Considering the immersive VR video services, literature [24] provides an asynchronous advantage actorâĂŞcritic (A3C)-based algorithm to minimize the long-term energy consumption of Terahertz wireless networks. In [25], the authors propose an A3C-based algorithm to maximize the computation rate and the transaction throughput of blockchain-enabled Mobile Edge Computing (MEC) systems. Another notable work is [26], which proposes a RL-based energy-aware resource management scheme for

wireless VR streaming in industrial Internet of Things (IIoTs). In [27], quantum collective learning and many-to-many matching game are adopted to solve the spectrum resource allocation problem and the distributed vehicles selection problem respectively. The authors in [28] aim at obtaining the optimal intelligence sharing policy by a collective deep reinforcement learning algorithm. Literature [29] improves the scalability of a service-oriented blockchain system by considering consensus protocols selection, block producers selection and network bandwidth allocation jointly.

B. MOTIVATION AND CONTRIBUTION

As discussed above, most of the existing research on the video caching and transmission optimization problem rarely took into account the time-varying VPD and WCSI. Some research contributions considered the dynamic scenarios, but they just considered multiple single videos and ignored the video contents with multiple quality versions. To fulfill this gap, this article focuses on optimizing video caching and UA for ABRVS in dynamic cache-enabled cellular networks with time-varying VPD and WCSI. Due to the surging energy cost of information industry, developing green communication becomes very urgent and important, which makes the energy efficiency a key performance indicator in current 5G and future 6G networks [30], [31], [32]. Thus, we adopt energy efficiency as the performance metric and aim at minimizing the long-term system energy consumption. Deep Q-learning network (DQN) is proposed to solve the problem. More specifically, the main contributions of this article are summarized as follows:

- We focus on the content caching and transmission of ABRVS in cache-enabled cellular networks. Considering the time-varying VPD and WCSI, a joint video caching and UA optimization problem is formulated to enhance the energy efficiency by minimizing the long-term system energy consumption, which is composed of video transmission energy consumption, video transcoding energy consumption and caching energy consumption.
- The formulated problem is then transformed into a discrete Markov decision process (MDP), which is solved by designing a DQN-based algorithm. Using this algorithm, we can obtain the optimal video caching and UA solutions. Considering the large state space and action space of the MDP, to cope with the "curse of dimensionality", linear approximation is integrated into the designed algorithm.
- Finally, during the simulations, the proposed algorithm's convergence is evaluated and its effectiveness is verified. The results show that the proposed algorithm outperforms benchmark algorithms in terms of energy efficiency and system energy consumption reduction over a long period.

C. ORGANIZATION

The remainder of this paper is organized as follows. Section II introduces the system model and formulates the optimization problem for video caching and UA. In Section III, we propose a DQN-based algorithm to solve the problem. The simulation settings, results, analysis and discussions are presented in Section IV. Finally, we conclude our work in Section V.

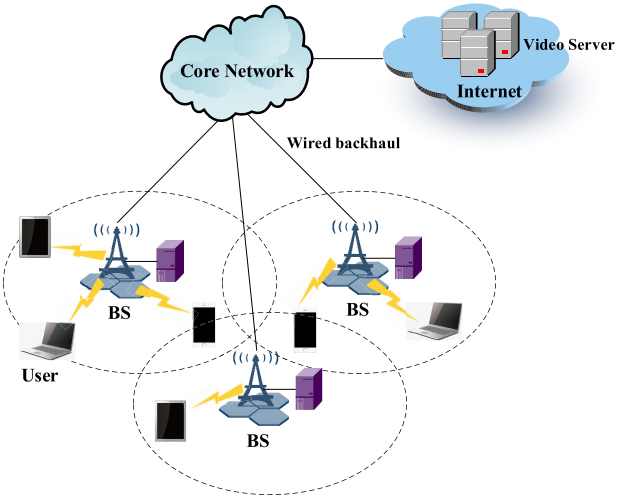

FIGURE 1. The cache-enabled cellular network architecture.

II. SYSTEM DESCRIPTION

In this section, we first present the system model, including network model, video request and caching model, transmission model and energy consumption model. Then the joint video caching and UA optimization problem for ABRVS is formulated.

A. SYSTEM MODEL

1) NETWORK MODEL

As illustrated in Figure 1, we are interested in delivering video content in a cache-enabled cellular network that consists of M ground BSs and N users. Let $\mathcal{M} = \{1, 2, \ldots, m, \ldots M\}$ and $\mathcal{N} = \{1, 2, \ldots, n, \ldots N\}$ denote the set of M ground BSs and the set of N users, respectively. Each ground BS is equipped with a MEC server with limited computing and caching resources, which are denoted as C_m^{comp} and C_m^{cache} respectively. The ground BSs are connected to the core network using wired backhaul links, which have a limited capacity. Meanwhile, users communicate with the ground BSs through radio access links, the bandwidth of which is assumed to be B. This bandwidth is shared among all the ground BSs.

Suppose that there are total F video chunks in the network, the set of which is denoted as $\mathcal{F} = \{1, 2, \ldots, f, \ldots F\}$. Each video chunk has K different quality versions with different bitrates and resolutions. Let $\mathcal{K} = \{1, 2, \ldots, k, \ldots K\}$ denote the set of K quality versions. We assume that quality version 1 has the lowest bitrate and resolution, while quality version K has the highest bitrate and resolution. Let $v_{f,k}$ denote the video chunk f with quality version k.

In general, caching policy in cache-enabled cellular networks strongly depends on the VPD, while UA strategy

is largely affected by the WCSI. It is assumed that the VPD and WCSI vary in different time slots. Suppose that the equal-sized time series is represented as $\mathcal{T} = \{1, 2, \ldots, t, \ldots\}$, where t denotes a time slot. Therefore, at the beginning of each time slot, the caching policy decision and UA strategy decision are made and updated.

2) VIDEO REQUEST AND CACHING MODEL

Note that the VPD is based on users' video interest and preference. In this article, the VPD, i.e., the request probabilities of different video chunks are modeled as a Zipf-like distribution [33], [34]. Let $p_{n,f}^{(t)}$ be the probability that user n requests video chunk f at time slot t and sort all F video chunks in a descending order based on their corresponding request probabilities, i.e., $p_{n,1}^{(t)} > p_{n,2}^{(t)} > \cdots > p_{n,f}^{(t)} > \cdots > p_{n,F}^{(t)}$. Thus, $p_{n,f}^{(t)}$ can be expressed as

$$p_{n,f}^{(t)} = \frac{f^{-\eta_n^{(t)}}}{\sum_{i=1}^{F} i^{-\eta_n^{(t)}}} \tag{1}$$

where $\eta_n^{(t)}$ is the skewness factor of user n at time slot t.

For each video chunk, the request probabilities of different quality versions are modeled as a normal distribution. The mean of this distribution is denoted by ϑ, which represents the dominant quality version [35]. Thus, the request probability of quality version k can be expressed as

$$p_k = e^{-(k-\vartheta)^2/2\sigma^2} \Big/ \sqrt{2\pi}\sigma \tag{2}$$

where σ^2 is the variance of the request probabilities of quality versions. A smaller σ leads to more concentrated requests of the dominant quality version ϑ. Based on $p_{n,f}^{(t)}$ and p_k, the probability of user n requesting $v_{f,k}$ at time slot t can be obtained by

$$p_{n,f,k}^{(t)} = p_{n,f}^{(t)} \cdot p_k \tag{3}$$

Denote $p_n^{(t)} = \left\{p_{n,1,1}^{(t)}, \ldots, p_{n,1,K}^{(t)}; \ldots; p_{n,F,1}^{(t)}, \ldots, p_{n,F,K}^{(t)}\right\}$ as user n's preference for all video chunks with different quality versions at time slot t. The time-varying VPD is modeled as a finite state Markov chain (FSMC). The corresponding state set is represented as $\mathcal{P} = \{p(\eta_1), p(\eta_2), \ldots, p(\eta_H)\}$, where $p(\eta_H)$ is the H-th state of VPD with skewness factor η_H. The total number of states is H and each state is a probability distribution of users requesting video chunks. The VPD transfers over time slots and the transition probability from one state to another state is denoted as $\mathbb{P}\left(p_n^{(t+1)} \middle| p_n^{(t)}\right)$, where $p_n^{(t)} \in \mathcal{P}, p_n^{(t+1)} \in \mathcal{P}$.

As for the video caching model, we denote the caching policy decision at time slot t as $\mathcal{X}^{(t)} = \left\{x_{m,f,k}^{(t)} | m \in \mathcal{M}, f \in \mathcal{F}, k \in \mathcal{K}\right\}$, where $x_{m,f,k}^{(t)} \in \{0, 1\}$. If BS m caches $v_{f,k}$ at time slot t, $x_{m,f,k}^{(t)} = 1$; otherwise, $x_{m,f,k}^{(t)} = 0$. Due to the limited caching resources, each BS can only store a part of video chunks. Here, we assume that BS m can store at most C_m^{cache} video chunks and $C_m^{cache} < F \cdot K$. Under the caching capacity constraint, $x_{m,f,k}^{(t)}$ satisfies

$$\sum_{f=1}^{F}\sum_{k=1}^{K} x_{m,f,k}^{(t)} \leq C_m^{cache}, \quad \forall m \in \mathcal{M} \tag{4}$$

Given the caching policy decision, the caching energy consumption at time slot t can be represented as

$$E_{cache}^{(t)} = \sum_{m=1}^{M}\sum_{f=1}^{F}\sum_{k=1}^{K} w_{cache} x_{m,f,k}^{(t)} s_{f,k} \tag{5}$$

where $s_{f,k}$ denotes the size of $v_{f,k}$, w_{cache} denotes the energy of caching one bit data in MEC servers (in J/bit).

3) TRANSMISSION MODEL

We denote the UA strategy decision at time slot t as $\mathcal{Y}^{(t)} = \left\{y_{m,n}^{(t)} | m \in \mathcal{M}, n \in \mathcal{N}\right\}$, where $y_{m,n}^{(t)} \in \{0, 1\}$. If user n is associated with BS m at time slot t, $y_{m,n}^{(t)} = 1$; otherwise, $y_{m,n}^{(t)} = 0$. We assume that one user can only be associated with one BS at each time slot, therefore, $y_{m,n}^{(t)}$ satisfies

$$\sum_{m=1}^{M} y_{m,n}^{(t)} = 1, \quad \forall n \in \mathcal{N} \tag{6}$$

If user n is associated with BS m at time slot t, the downlink transmission rate from BS m to user n can be represented as

$$R_{m,n}^{(t)} = \frac{B}{\sum_{n=1}^{N} y_{m,n}^{(t)}} \log\left(1 + \gamma_{m,n}^{(t)}\right) \tag{7}$$

where $\gamma_{m,n}^{(t)}$ denotes the received signal-to-interference-plus-noise-ratio (SINR) of user n from BS m at time slot t, $\sum_{n=1}^{N} y_{m,n}^{(t)}$ is the number of users associated with BS m. Here, B is equally allocated to all associated users [17]. The time-varying WCSI, i.e., the SINR $\gamma_{m,n}^{(t)}$, is modeled as a FSMC [36], [37]. In this model, the value range of $\gamma_{m,n}^{(t)}$ is quantized into D discrete levels: if $\gamma_0^* \leq \gamma_{m,n}^{(t)} < \gamma_1^*$, γ_1; if $\gamma_1^* \leq \gamma_{m,n}^{(t)} < \gamma_2^*$, γ_2; ...; if $\gamma_{m,n}^{(t)} \geq \gamma_{D-1}^*$, γ_D. Each level is a state of the FSMC and the corresponding state set is represented as $\Re = \{\gamma_1, \gamma_2, \ldots, \gamma_D\}$. The SINR transfers over time slots and the transition probability from one state to another state is denoted as $\mathbb{P}\left(\gamma_{m,n}^{(t+1)} \middle| \gamma_{m,n}^{(t)}\right)$, where $\gamma_{m,n}^{(t)} \in \Re$, $\gamma_{m,n}^{(t+1)} \in \Re$.

4) CONTENT DELIVERY ENERGY CONSUMPTION MODEL

Let $E_{m,n}^{(t)}$ denote the content delivery energy consumption from BS m to user n at time slot t. Because of the computing and caching resources available from MEC servers, BSs have the ability to cache video chunks as well as transcode a video chunk from a higher quality version to a lower quality version. Depending on whether BSs cache the requested quality version or a higher quality version, there are three cases to handle requests from users. In the following, the content delivery energy consumption of these three cases will be discussed.

Case 1: BS m has cached the requested $v_{f,k}$ at time slot t (i.e., $x^{(t)}_{m,f,k}=1$), and $v_{f,k}$ can be delivered to user n directly. In this case, $E^{(t)}_{m,n}$ only contains the energy consumption for downlink radio transmission of the requested $v_{f,k}$ from BS m to user n denoted as $E^{(t),trans}_{m,n,f,k}$. Thus, $E^{(t),1}_{m,n}$ can be calculated by

$$E^{(t),1}_{m,n}=\sum_{f=1}^{F}\sum_{k=1}^{K}p^{(t)}_{n,f,k}x^{(t)}_{m,f,k}E^{(t),trans}_{m,n,f,k} \tag{8}$$

where $E^{(t),trans}_{m,n,f,k}$ is given as $E^{(t),trans}_{m,n,f,k}=P_m\frac{s_{f,k}}{R^{(t)}_{m,n}}$. P_m is the transmission power of BS m.

Case 2: BS m does not cache the requested $v_{f,k}$ at time slot t (i.e., $x^{(t)}_{m,f,k}=0$), but caches a higher quality version (i.e., $h^{(t)}_{m,f,k}=\min\left\{\sum_{k'=k+1}^{K}x^{(t)}_{m,f,k'},1\right\}=1$). In this case, handling users' requests is divided into two steps: transcoding and downlink radio transmission. Thus, $E^{(t)}_{m,n}$ contains $E^{(t),trans}_{m,n,f,k}$ and the energy consumption for transcoding the cached higher quality version to the requested quality version denoted as $E^{(t),comp}_{m,n,f,k}$. $E^{(t),2}_{m,n}$ can be calculated by

$$E^{(t),2}_{m,n}=\sum_{f=1}^{F}\sum_{k=1}^{K}p^{(t)}_{n,f,k}\left(1-x^{(t)}_{m,f,k}\right)h^{(t)}_{m,f,k}\left(E^{(t),trans}_{m,n,f,k}+E^{(t),comp}_{m,n,f,k}\right) \tag{9}$$

where $h^{(t)}_{m,f,k}\in\{0,1\}$. If BS m caches at least one quality version of video chunk f higher than k at time slot t, $h^{(t)}_{m,f,k}=1$; otherwise, $h^{(t)}_{m,f,k}=0$. $E^{(t),comp}_{m,n,f,k}$ is given as $E^{(t),comp}_{m,n,f,k}=P^{(t),comp}_{m,n}\frac{w_0\left(s_{f,k^*}-s_{f,k}\right)}{c^{(t)}_{m,n}}$, where $P^{(t),comp}_{m,n}$ represents the transcoding power consumption of BS m to handle user n's request at time slot t, $\frac{w_0\left(s_{f,k^*}-s_{f,k}\right)}{c^{(t)}_{m,n}}$ represents the transcoding time [38], w_0 represents the CPU cycles required to transcode one bit data (in cycles/bit), $k^*=\left\{k'\middle|\min\left(k'-k\right),k<k',x^{(t)}_{m,f,k'}=1\right\}$ represents the minimum cached quality version of video chunk f higher than k, $c^{(t)}_{m,n}$ represents the computing resources allocated to user n by BS m at time slot t. Similar to [39] and [40], we model $P^{(t),comp}_{m,n}$ as $P^{(t),comp}_{m,n}=w_{comp}\left(c^{(t)}_{m,n}\right)^3$, where w_{comp} is a constant coefficient related to the CPU chip architecture. Here, we assume that the computing resources of BSs are equally allocated to all associated users. Thus, $c^{(t)}_{m,n}$ can be expressed as $c^{(t)}_{m,n}=\frac{C^{comp}_m}{\sum_{n=1}^{N}y^{(t)}_{m,n}}$.

Case 3: BS m caches neither the requested $v_{f,k}$ at time slot t (i.e., $x^{(t)}_{m,f,k}=0$) nor a higher quality version (i.e., $h^{(t)}_{m,f,k}=0$). In this case, handling users' requests is also divided into two steps: backhaul link transmission and downlink radio transmission. Here, we assume that different quality versions of all video chunks are available in the core network. Thus, $E^{(t)}_{m,n}$ contains $E^{(t),trans}_{m,n,f,k}$ and the energy consumption for backhaul link transmission of the requested $v_{f,k}$ from the core network to BS m denoted as $E^{(t),BH}_{m,n,f,k}$. $E^{(t)}_{m,n}$ can be calculated by

$$E^{(t),3}_{m,n}=\sum_{f=1}^{F}\sum_{k=1}^{K}p^{(t)}_{n,f,k}\left(1-\sum_{k'=k}^{K}x^{(t)}_{m,f,k'}\right)\left(E^{(t),trans}_{m,n,f,k}+E^{(t),BH}_{m,n,f,k}\right) \tag{10}$$

where $E^{(t),BH}_{m,n,f,k}$ is given as $E^{(t),BH}_{m,n,f,k}=w_{BH}s_{f,k}\frac{s_{f,k}}{R^{(t),BH}_{m,n}}$. w_{BH} represents the power of backhaul links to transmit one bit data (in Watt/bit). $R^{(t),BH}_{m,n}$ represents the backhaul link transmission rate allocated to user n by BS m at time slot t. Here, the backhaul link transmission rate of BSs is equally allocated to all associated users. Thus, $R^{(t),BH}_{m,n}$ can be expressed as $R^{(t),BH}_{m,n}=\frac{R^{BH}_m}{\sum_{n=1}^{N}y^{(t)}_{m,n}}$.

Based on the above analysis of the content delivery energy consumption of these three cases, $E^{(t)}_{m,n}$ is given by

$$E^{(t)}_{m,n}=E^{(t),1}_{m,n}+E^{(t),2}_{m,n}+E^{(t),3}_{m,n} \tag{11}$$

Therefore, the total content delivery energy consumption at time slot t can be represented as

$$E^{(t)}_{delivery}=\sum_{m=1}^{M}\sum_{n=1}^{N}y^{(t)}_{m,n}E^{(t)}_{m,n} \tag{12}$$

B. PROBLEM FORMULATION

Given the above models, the total energy consumption at each time slot consists of the caching energy consumption and the content delivery energy consumption, which can be expressed as

$$E^{(t)}_{total}=E^{(t)}_{cache}+E^{(t)}_{delivery} \tag{13}$$

In this article, our goal is to minimize the long-term system energy consumption by jointly optimizing video caching and UA with the given WCSI $\gamma^{(t)}_{m,n},\forall m\in\mathcal{M},\forall n\in\mathcal{N}$ and VPD $p^{(t)}_n,\forall n\in\mathcal{N}$ in dynamic networks. To achieve this goal, according to (13), we formulate the long-term optimization problem as follows:

$$\min_{\mathcal{X}^{(t)},\mathcal{Y}^{(t)}}\sum_{t\in\mathcal{T}}E^{(t)}_{total} \tag{14}$$

$$\text{s.t.}\ \sum_{f=1}^{F}\sum_{k=1}^{K}x^{(t)}_{m,f,k}\le C^{cache}_m,\ \forall m\in\mathcal{M} \tag{14a}$$

$$\sum_{m=1}^{M}y^{(t)}_{m,n}=1,\ \forall n\in\mathcal{N} \tag{14b}$$

$$x^{(t)}_{m,f,k}\in\{0,1\},\ \forall m\in\mathcal{M},\forall f\in\mathcal{F},\forall k\in\mathcal{K} \tag{14c}$$

$$y^{(t)}_{m,n}\in\{0,1\},\ \forall m\in\mathcal{M},\forall n\in\mathcal{N} \tag{14d}$$

where (14a)-(14d) show the constraints. Constraint (14a) indicates the caching capacity limitation of each BS. Constraint (14b) indicates that one user can only be associated with one BS at each time slot. Constraints (14c) and (14d)

indicate that $x_{m,f,k}^{(t)}$ and $y_{m,n}^{(t)}$ are both binary variables, i.e., the values of them are either 0 or 1.

So far, the joint video caching and UA optimization problem has been formulated. In Section III, we will show how to solve the problem (14) and present a solution to it.

III. SOLUTION TO JOINT VIDEO CACHING AND UA OPTIMIZATION PROBLEM

In this section, aiming at minimizing the long-term total energy consumption in dynamic networks, we first transform the optimization problem (14) into a MDP. Then, for each time slot, given VPD $p_n^{(t)}, \forall n \in \mathcal{N}$ and WCSI $\gamma_{m,n}^{(t)}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}$, a DQN-based algorithm is designed, using which the optimal video caching and UA solutions can be obtained.

A. MARKOV DECISION PROCESS MODEL

In general, a MDP problem is defined by a tuple $\{\mathbb{S}, \mathbb{A}, \mathbb{P}, r\}$, where $\mathbb{S}$ represents state space, $\mathbb{A}$ represents action space, $\mathbb{P}$ represents state transition probability, r represents the immediate reward. Specifically, according to the optimization problem (14), the key elements of MDP are defined as follows.

1) STATE SPACE

$\mathbb{S}$ contains all possible states in dynamic networks, so $S^{(t)} \in \mathbb{S}$, where $S^{(t)}$ represents the network state at time slot t. $S^{(t)}$ is composed of VPD and WCSI at time slot t. Therefore, $S^{(t)}$ is defined as

$$S^{(t)} = \left\{p^{(t)}, \gamma^{(t)}\right\} \tag{15}$$

where $p^{(t)} = \left\{p_n^{(t)} \,|n \in \mathcal{N}\right\}, \gamma^{(t)} = \left\{\gamma_{m,n}^{(t)} \,|m \in \mathcal{M}, n \in \mathcal{N}\right\}$.

2) ACTION SPACE

$\mathbb{A}$ is the set of feasible actions in dynamic networks, so $A^{(t)} \in \mathbb{A}$, where $A^{(t)}$ represents the action at time slot t. $A^{(t)}$ is composed of video caching policy decision $\mathcal{X}^{(t)} = \left\{x_{m,f,k}^{(t)} \,|m \in \mathcal{M}, f \in \mathcal{F}, k \in \mathcal{K}\right\}$ and UA strategy decision at time slot t $\mathcal{Y}^{(t)} = \left\{y_{m,n}^{(t)} \,|m \in \mathcal{M}, n \in \mathcal{N}\right\}$. Therefore, $A^{(t)}$ is defined as

$$A^{(t)} = \left\{\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\right\} \tag{16}$$

3) TRANSITION PROBABILITY

The state transition probability is defined as $\mathbb{P}(S^{(t+1)}|S^{(t)}, A^{(t)})$.

4) REWARD FUNCTION

At time slot t, an agent first observes and senses the state of the dynamic network environment $S^{(t)}$. Then, according to a certain policy function π, the agent performs an action $A^{(t)}$. After the action is taken, the agent can obtain an immediate reward. Since the goal of the optimization problem (14) is to minimize the energy consumption, to achieve this goal, the energy consumption is set as the main reward. Therefore, the immediate reward is defined as

$$r\left(S^{(t)}, A^{(t)}\right) = -E_{total}^{(t)} \tag{17}$$

The MDP problem can be solved by determining the optimal policy π^* that maximizes the long-term system reward. Here, $\pi : \mathbb{S} \to \mathbb{A}$ is a policy function that maps a state $S \in \mathbb{S}$ to an action $A \in \mathbb{A}$. There are two popular methods to assess the long-term system reward, namely state value function and state-action value function. Given a policy π, the state value function in state S is defined as

$$V^{\pi}(S) = \mathbb{E}^{\pi}\left[\psi^{(t)} \middle| S^{(t)} = S\right] \tag{18}$$

where $\mathbb{E}^{\pi}[\cdot]$ represents the mathematical expectation, $\psi^{(t)} = \sum_{\tau=t}^{\infty} \beta^{\tau-t} r\left(S^{(\tau)}, A^{(\tau)}\right)$. $\beta \in (0, 1]$ is the discount factor to determine the importance of immediate reward and future rewards. Similarly, the state-action value function in state S and action A is defined as

$$Q^{\pi}(S, A) = \mathbb{E}^{\pi}\left[\psi^{(t)} \middle| S^{(t)} = S, A^{(t)} = A\right] \tag{19}$$

RL algorithms as a branch of machine learning algorithms are generally used to solve the MDP problem. Q-learning algorithm [41] is a classical RL algorithm, which aims to train an agent to learn π^*. Since the state-action value function is used to assess the long-term system reward, learning π^* is equivalent to learning the optimal state-action value function $Q^*(S, A)$. π^* can be determined by $Q^*(S, A)$, i.e., $\pi^*(S) = \arg\max_{A \in \mathbb{A}} Q^*(S, A), \forall S \in \mathbb{S}$. In order to learn $Q^*(S, A)$, the agent needs to interact with the dynamic network environment repeatedly. Specifically, during each interaction step, the agent observes the environment's state, chooses an action, executes it, and receives an immediate reward. As a result of executing the action, the state S is transferred to the next state S'. Then, the state-action value function is updated by

$$Q(S, A) \leftarrow (1-\zeta) Q(S, A) + \zeta\left[r(S, A) + \beta \max_{A' \in \mathbb{A}} Q\left(S', A'\right)\right] \tag{20}$$

where $\zeta \in (0, 1]$ represents the learning rate. After several interaction steps, the agent can eventually learn π^* and $Q^*(S, A)$.

During the learning process, there are two methods for the agent to select an action, namely "exploitation" and "exploration". "Exploitation" means that the agent selects the action with the highest state-action value. "Exploration" means that the agent randomly selects an action except for the action with the highest state-action value. The "exploitation" process can maximize the long-term system reward, while the "exploration" process can avoid the Q-learning algorithm converging into a local optimum. Therefore, to learn π^*, the "exploitation" and "exploration" need to be balanced when selecting an action under a given state.

B. THE DQN-BASED CACHING AND UA ALGORITHM

In the Q-learning algorithm, the state-action values of all state-action pairs are stored in a Q-table. However, with the increase of state space and action space, the Q-learning algorithm faces the challenge of "curse of dimensionality", which means that when the number of state-action pairs is huge, storing and searching the Q-table will take lots of time and space, leading to a slow learning speed and influencing the convergence efficiency. Motivated by deep learning, DQN [42] has been proposed to overcome the above challenges.

Based on deep neural networks (DNNs)' nonlinear nature, deep learning can utilize DNNs to approximate almost any function by finding the low-dimensional features of high-dimensional data. The core idea of DQN is to combine Q-learning algorithm with deep learning. DNNs are utilized to approximate the state-action value function, i.e., $Q(S, A; \theta) \approx Q(S, A)$, where θ represents the set of weights and biases in DNNs. Two outstanding innovations are applied to make DQN more efficient and robust:

1) EXPERIENCE REPLAY

Experience replay utilizes a finite-sized replay memory to store the agent's past learning experience, i.e., $(S^{(t)}, A^{(t)}, r^{(t)}, S^{(t+1)})$. By this way, the DQN can break the temporal correlations among past learning experiences and make the DNNs updating more efficient.

2) FIXED TARGET DNN

There are two DNNs in DQN, i.e., the evaluated DNN and the target DNN, which have the same architecture. At each training step, the weights and biases in the evaluated DNN are updated. However, the weights and biases in the target DNN are kept fixed for a period of time and are only updated with the evaluated DNN periodically. Here, we assume that the weights and biases in the target DNN are updated every G training steps, i.e., $\theta^{-(t)} = \theta^{(t-G)}$, where θ and θ^{-} are the weights and biases in the evaluated DNN and the target DNN respectively. This innovation can stabilize and smooth the learning process.

Both the evaluated DNN and the target DNN take the state $S \in \mathbb{S}$ as input and all actions' state-action values under the state S as output. At each training step, the agent randomly selects a mini-batch of samples from the replay memory and updates the weights and biases in the evaluated DNN to minimize loss function $Loss(\theta)$, which is defined as the mean-squared deviation between the target state-action value $Q_{target} = r(S, A) + \beta \max_{A' \in \mathbb{A}} Q(S', A'; \theta^{-})$ and the estimated state-action value $Q(S, A; \theta)$, i.e., $Loss(\theta) = \mathbb{E}\left[\left(Q_{target} - Q(S, A; \theta)\right)^2\right]$. The workflows of DQN are presented in Figure 2.

In the standard DQN algorithm, the output of the evaluated DNN and the target DNN is all actions' state-action values. So the output layer's dimension in the evaluated DNN and the target DNN is related to the size of the action space, which is

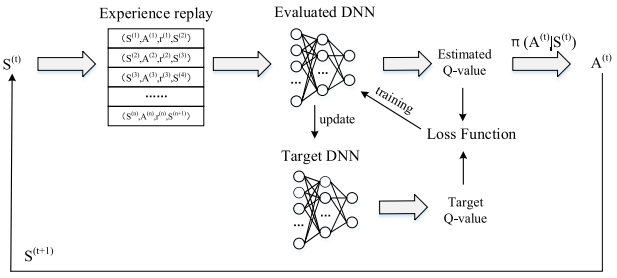

FIGURE 2. The workflows of DQN.

$|\mathbb{A}| = \binom{FK}{C_m^{cache}}^M (2^{MN})$ in the formulated MDP problem. $\binom{FK}{C_m^{cache}}^M$ is the number of all possible caching policies and 2^{MN} is the number of all possible UA strategies. It is obvious that as the number of video chunks F, the number of ground BSs M and the number of users N increasing, the size of the action space increases exponentially, resulting in the challenge of "curse of dimensionality". To overcome the challenge, we integrate linear approximation into the DQN algorithm, which reduces the action space from exponential size $\binom{FK}{C_m^{cache}}^M (2^{MN})$ to linear size $MFK + MN$. In linear approximation-integrated DQN algorithm, the first MFK outputs of the evaluated DNN are used to select the caching policy $\mathcal{X}^{(t)}$ and the last MN outputs of the evaluated DNN are used to select the UA strategy $\mathcal{Y}^{(t)}$, both of which are combined as the action $A^{(t)} = \{\mathcal{X}^{(t)}, \mathcal{Y}^{(t)}\}$.

Specifically, we denote the first MFK outputs of the evaluated DNN under a state as $\{Q_1, Q_2, \ldots, Q_{MFK}\}$. Then, the elements of BS 1's caching policy that correspond with the largest C_m^{cache} values in $\{Q_1, Q_2, \ldots, Q_{FK}\}$ are set to 1 and the other elements are set to 0. Similarly, the elements of BS 2's caching policy that correspond with the largest C_m^{cache} values in $\{Q_{FK+1}, Q_{FK+2}, \ldots, Q_{2FK}\}$ are set to 1 and the other elements are set to 0. By this way, all M BSs' caching policies $\mathcal{X}^{(t)}$ can be obtained. Moreover, we denote the last MN outputs of the evaluated DNN under a state as $\{Q_{MFK+1}, Q_{MFK+2}, \ldots, Q_{MFK+MN}\}$. Then, the element of user 1's UA strategy that corresponds with the largest value in $\{Q_{MFK+1}, Q_{MFK+2}, \ldots, Q_{MFK+M}\}$ is set to 1 and the other elements are set to 0. Similarly, the element of user 2's UA strategy that corresponds with the largest value in $\{Q_{MFK+M+1}, Q_{MFK+M+2}, \ldots, Q_{MFK+2M}\}$ is set to 1 and the other elements are set to 0. By this way, all N users' UA strategies $\mathcal{Y}^{(t)}$ can be obtained. Thus, we can ensure that the obtained $\mathcal{X}^{(t)}$ satisfies the constraint (14a) and the obtained $\mathcal{Y}^{(t)}$ satisfies the constraint (14b).

According to the basic principle of DQN and linear approximation, a DQN-based caching and UA algorithm, i.e., Algorithm 1 is presented to solve the optimization problem (14). ε-greedy policy (lines 10-15) is used to select

an action under the observed state aiming at balancing the "exploitation" and "exploration".

Algorithm 1 The DQN-Based Caching and UA Algorithm

Initialization:
Initialize the maximum number of training episodes $\Xi_{\max}$ and the maximum number of steps in each episode $g_{\max}$.
Initialize the experience replay memory and the mini-batch size.
Initialize the discount factor β, the learning rate ζ and the exploration probability ε.
Initialize the weights and biases in the evaluated DNN with θ.
Initialize the weights and biases in the target DNN with $\theta^- = \theta$.
for *episode* $= 1, 2, \ldots, \Xi_{\max}$ **do**
  Reset the environment with the initial state S_{ini}, i.e., $S^{(t)} = S_{ini}$.
  for $t = 1, 2, \ldots, g_{\max}$ **do**
    Choose a random probability p.
    if $p > \varepsilon$ **then**
      Select an action $A^{(t)}$ with linear approximation.
    else
      Randomly select an action $A^{(t)}$.
    end if
    Execute the selected action, obtain the immediate reward $r^{(t)}$ and observe the next state $S^{(t+1)}$.
    Store $\left(S^{(t)}, A^{(t)}, r^{(t)}, S^{(t+1)}\right)$ into the experience replay memory.
    Randomly select a mini-batch of samples from the experience replay memory.
    Calculate the estimated state-action value $Q(S, A; \theta)$.
    Calculate the target state-action value Q_{target} by $Q_{target} = r(S, A) + \beta \max_{A' \in \mathbb{A}} Q\left(S', A'; \theta^-\right)$.
    Train the evaluated DNN to minimize the loss function $Loss(\theta) = \mathbb{E}\left[\left(Q_{target} - Q(S, A; \theta)\right)^2\right]$.
    Update θ^- every G training steps.
    Set $S^{(t)} \leftarrow S^{(t+1)}$.
  end for
end for

IV. SIMULATION RESULTS AND DISCUSSIONS

In this section, we employ computational simulation method to evaluate the effectiveness of the proposed DQN-based caching and UA algorithm. For the sake of simplicity in our simulations, the algorithm is referred to as "DQN-based CA and UA". All simulations are conducted on a X64-based laptop, which is equipped with 2.8GHz Intel Core i7, 32GB LPDDR3 and 512GB memory. The proposed algorithm is implemented in PyTorch 1.12.1 with Python 3.9. We consider a cellular network with $M = 3$ BSs and $N = 10$ users. We set $F = 10$, $K = 3$, $C_m^{cache} = 4$, $C_m^{comp} = 3.4GHz$, $P_m = 46dBm$ [20], $B = 20MHz$, $\vartheta = 2$, $w_0 = 400cycles/bit$ [38], $R_m^{BH} = 1.5Mbps$ [21], $w_{cache} = 8 \times 10^{-8} J/bit$ [43], $w_{comp} = 10^{-27}$ [40], $w_{BH} = 8 \times 10^{-6} Watt/bit$ [44]. Besides, the sizes of video chunks with three quality versions are set as 1, 2 and 4Mbit. Furthermore, the VPD is set as a three-state FSMC with three different skewness factors $\{\eta_1, \eta_2, \eta_3\} = \{0.2, 0.5, 0.8\}$. Their transition probability matrix is assumed as

$$P^{\eta} = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.6 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix} \tag{21}$$

Similarly, the WCSI is set as a three-state FSMC with three different spectrum efficiency parameters, i.e., 10, 2 and 0.2, which means that the state of wireless channels between BSs and users are good, medium and bad respectively. Their transition probability matrix is assumed as

$$P^{SE} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.7 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \tag{22}$$

For comparison, the following three benchmark algorithms are considered:

A. "DQN-BASED CA ONLY"

In this algorithm, the learning agent only tries to learn the optimal caching policy, but makes UA strategy decision randomly. That is to say, users are associated with BSs randomly. Compared with "DQN-based CA and UA", the potential benefits of optimizing UA strategy can be indicated.

B. "DQN-BASED UA ONLY"

In this algorithm, the learning agent only tries to learn the optimal UA strategy, but makes caching policy decision randomly. That is to say, each BS caches video chunks randomly until its caching capacity is filled up. Compared with "DQN-based CA and UA", the potential benefits of optimizing caching policy can be indicated.

C. "RANDOM CA AND UA"

In this algorithm, the learning agent makes both caching policy and UA strategy decisions randomly. Compared with "DQN-based CA and UA", the potential benefits of jointly optimizing caching policy and UA strategy can be indicated.

We first investigate the convergence performance of all the algorithms. Here, we set $\zeta = 0.01$, $\beta = 0.9$, $\varepsilon = 0.1$. In Figure 3, the abscissa denotes the number of episodes (each episode contains 40 time slots) and the ordinate represents the values of reward per episode, i.e., the system energy consumption. It can be seen that all DQN-based algorithms can gradually converge to a stable value with the number of episodes increasing. Specifically, "DQN-based CA and UA", "DQN-based CA only" and "DQN-based UA only" reach stability after about 1150 episodes, 650 episodes and 2600 episodes respectively. Besides, it shows that the converged stable value of "DQN-based CA and UA" is

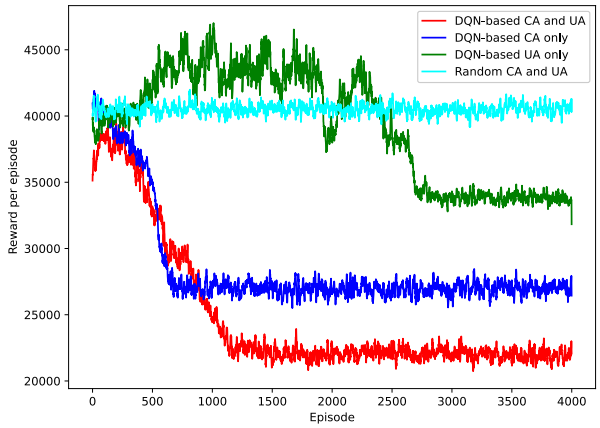

FIGURE 3. Convergence of the proposed algorithm.

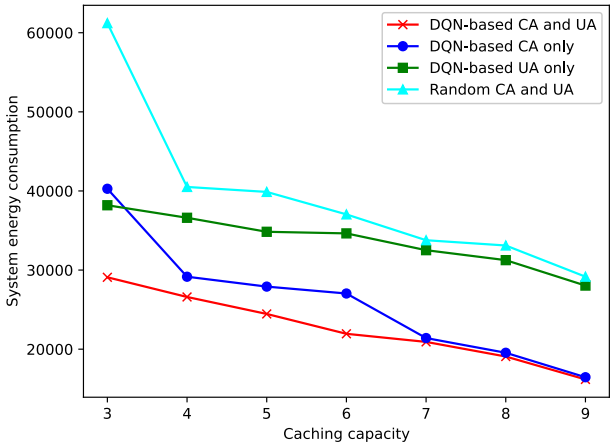

FIGURE 4. Total energy consumption under different values of the caching capacity of each BS.

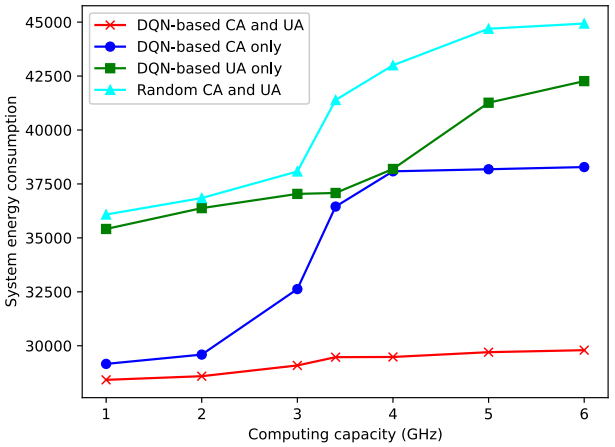

FIGURE 5. Total energy consumption under different values of the computing capacity of each BS.

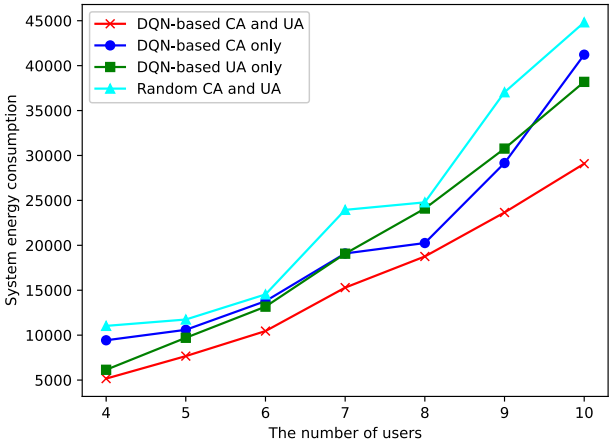

FIGURE 6. Total energy consumption under different values of the number of users.

the smallest, the corresponding value of "Random CA and UA" is the largest, while the corresponding values of other algorithms are medium. This demonstrates the potential benefits of jointly optimizing caching policy and UA strategy.

Figure 4 illustrates the system energy consumption of all the algorithms under different values of the caching capacity of each BS C_m^{cache}. We observe that the system energy consumption of all the algorithms decreases with the increase of C_m^{cache}. This is intuitive, when C_m^{cache} is larger, each BS can cache more video chunks, and more users' requests can be satisfied locally without incurring video transcoding and backhaul link transmission. As a result, the video transcoding energy consumption and backhaul link transmission energy consumption are decreased, which leads to the decrease of the total energy consumption. This figure also shows that the total energy consumption achieved by the proposed algorithm is smaller than the other benchmark algorithms.

The impact of computing capacity of each BS C_m^{comp} on the performance of all the algorithms is revealed in Figure 5. As expected, when C_m^{comp} becomes larger, the system energy consumption of all the algorithms increases. The reason is that larger C_m^{comp} means a larger $P_{m,n}^{(t),comp}$ and a smaller time of transcoding the cached higher quality version to a lower quality version. As a result, the video transcoding energy consumption $E_{m,n,f,k}^{(t),comp}$ is increased, which leads to the increase of the total energy consumption.

Figure 6 reveals the relationship between the performance of all the algorithms and the number of users N. As we can see, the larger the number of users, the larger the system energy consumption of all the algorithms, which is in line with the intuition. Larger N leads to the increase of the number of video requests. In this case, given the VPD and caching capacity, more video requests consume more radio transmission energy consumption, video transcoding energy consumption and backhaul link transmission energy consumption, resulting in the increase of the total energy consumption. Besides, as N increases, the performance gap between the proposed algorithm and the other benchmark algorithms becomes more pronounced. The reason is that when N is small, the impact of optimizing caching policy and UA strategy on the performance is small, while when N becomes larger, optimizing caching policy and UA strategy plays a more important role for the performance.

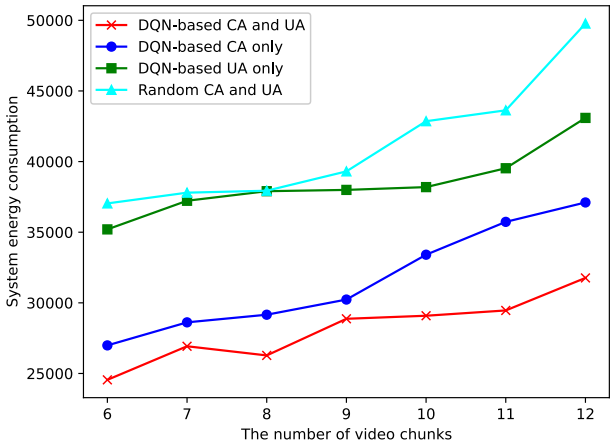

FIGURE 7. Total energy consumption under different values of the number of video chunks.

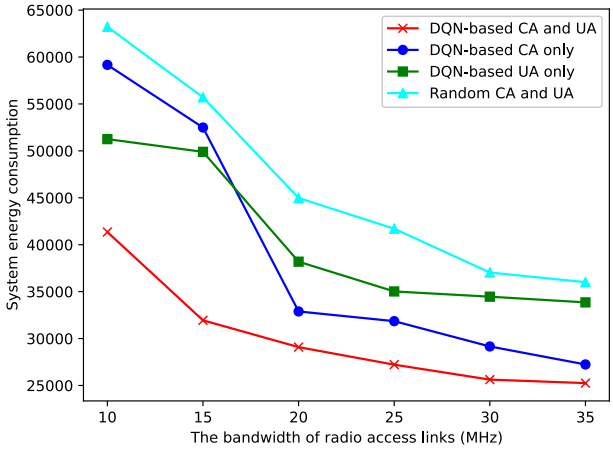

FIGURE 8. Total energy consumption under different values of the bandwidth of radio access links.

How the number of video chunks F affects the performance of all the algorithms is illustrated in Figure 7. As we can see, the system energy consumption of all the algorithms gradually increases with the increase of F. The reason is that when F becomes larger, users' requests scatter more widely, which leads to higher cache miss rate and definitely consumes larger content delivery energy consumption, which leads to the increase of the total energy consumption.

Figure 8 reveals the impact of the bandwidth of radio access links B on the performance of all the algorithms. As expected, with the increase of B, the system energy consumption of all the algorithms gradually decreases. The reason is that when B becomes larger, $R_{m,n}^{(t)}$ increases which leads to the decrease of radio transmission delay and energy consumption. As a result, the total energy consumption is reduced.

Next, we demonstrate the performance of all the algorithms under different values of the backhaul link transmission rate R_m^{BH}. As shown in Figure 9, when R_m^{BH} is larger, the system energy consumption of all the algorithms decreases. This is intuitive, larger R_m^{BH} means smaller backhaul link

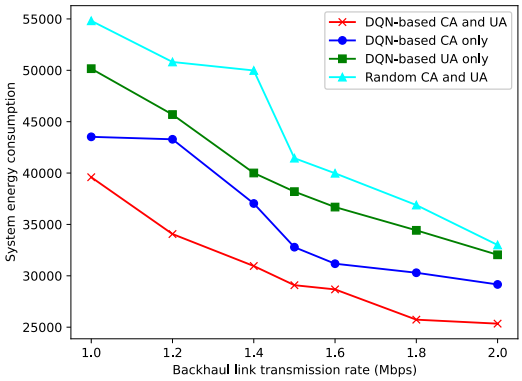

FIGURE 9. Total energy consumption under different values of the backhaul link transmission rate.

transmission delay and energy consumption, which leads to the decrease of the total energy consumption.

V. CONCLUSION

This article has investigated the content caching, transcoding and transmission for ABRVS in cache-enabled cellular networks. Taking into account the dynamic characteristics of video popularity distribution and wireless channels, we focused on reducing the long-term system energy consumption. To achieve this, we formulated the problem of optimizing video caching and UA as a MDP. We utilized both DQN and linear approximation techniques to tackle the MDP problem and determine the optimal video caching and UA decisions. Simulation results have demonstrated that both video caching and UA decisions have effect on the system energy consumption, and the proposed algorithm yields significant performance gains in enhancing the energy efficiency compared with benchmark algorithms. In our future works, UAV-assisted cellular networks will be considered, and UAV deployment, UA, content caching and resource allocation will be jointly optimized to improve users' QoE for ABRVS.

REFERENCES

[1] Ericsson. (Nov. 2022). *Ericsson Mobility Report*. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report
[2] Z. Piao, M. Peng, Y. Liu, and M. Daneshmand, "Recent advances of edge cache in radio access networks for Internet of Things: Techniques, performances, and challenges," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1010–1028, Feb. 2019.
[3] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.
[4] B. Jedari, G. Premsankar, G. Illahi, M. D. Francesco, A. Mehrabi, and A. Ylä-Jääski, "Video caching, analytics, and delivery at the wireless edge: A survey and future directions," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 431–471, 1st Quart., 2021.
[5] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive bitrate selection: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2985–3014, 4th Quart., 2017.
[6] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 1965–1978, Sep. 2019.
[7] J. Zhang, H. Wu, X. Tao, and X. Zhang, "Adaptive bitrate video streaming in non-orthogonal multiple access networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 3980–3993, Apr. 2020.
[8] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2522–2545, 4th Quart., 2016.

[9] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2134–2168, 3rd Quart., 2019.
[10] Y. Jin, Y. Wen, and C. Westphal, "Optimal transcoding and caching for adaptive streaming in media cloud: An analytical approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1914–1925, Dec. 2015.
[11] W. Liu, H. Zhang, H. Ding, and D. Yuan, "Delay and energy minimization for adaptive video streaming: A joint edge caching, computing and power allocation approach," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9602–9612, Sep. 2022.
[12] Y. Guo, Q. Yang, F. R. Yu, and V. C. M. Leung, "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5445–5459, Jun. 2018.
[13] L. Li, D. Shi, R. Hou, R. Chen, B. Lin, and M. Pan, "Energy-efficient proactive caching for adaptive video streaming via data-driven optimization," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5549–5561, Jun. 2020.
[14] N.-N. Dao, D. T. Ngo, N.-T. Dinh, T. V. Phan, N. D. Vo, S. Cho, and T. Braun, "Hit ratio and content quality tradeoff for adaptive bitrate streaming in edge caching systems," *IEEE Syst. J.*, vol. 15, no. 4, pp. 5094–5097, Dec. 2021.
[15] C. Liu, H. Zhang, H. Ji, and X. Li, "MEC-assisted flexible transcoding strategy for adaptive bitrate video streaming in small cell networks," *China Commun.*, vol. 18, no. 2, pp. 200–214, Feb. 2021.
[16] F. Jiang, Z. Yuan, C. Sun, and J. Wang, "Deep Q-learning-based content caching with update strategy for fog radio access networks," *IEEE Access*, vol. 7, pp. 97505–97514, 2019.
[17] Y. Jiang, M. Ma, M. Bennis, F. Zheng, and X. You, "User preference learning-based edge caching for fog radio access network," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1268–1283, Feb. 2019.
[18] T. Zhang, Y. Wang, W. Yi, Y. Liu, C. Feng, and A. Nallanathan, "Two time-scale caching placement and user association in dynamic cellular networks," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2561–2574, Apr. 2022.
[19] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Caching placement and resource allocation for cache-enabling UAV NOMA networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12897–12911, Nov. 2020.
[20] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Joint resource, deployment, and caching optimization for AR applications in dynamic UAV NOMA networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 5, pp. 3409–3422, May 2022.
[21] T. Zhang, X. Fang, Z. Wang, Y. Liu, and A. Nallanathan, "Stochastic game based cooperative alternating Q-learning caching in dynamic D2D networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13255–13269, Dec. 2021.
[22] P. Lin, Q. Song, J. Song, A. Jamalipour, and F. R. Yu, "Cooperative caching and transmission in comp-integrated cellular networks using reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5508–5520, May 2020.
[23] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.
[24] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9517–9529, Oct. 2020.
[25] J. Feng, F. R. Yu, Q. Q. Pei, X. L. Chu, J. B. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.
[26] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. C. M. Leung, "Resource management for pervasive-edge-computing-assisted wireless VR streaming in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7607–7617, Nov. 2021.
[27] Y. Ren, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Quantum collective learning and many-to-many matching game in the metaverse for connected and autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12128–12139, Nov. 2022.
[28] Q. Tang, R. Xie, F. R. Yu, T. Chen, R. Zhang, T. Huang, and Y. Liu, "Collective deep reinforcement learning for intelligence sharing in the internet of intelligence-empowered edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6327–6342, Nov. 2023.
[29] C. Qiu, H. Yao, F. R. Yu, C. Jiang, and S. Guo, "A service-oriented permissioned blockchain for the Internet of Things," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 203–215, Mar. 2020.
[30] A. H. Sodhro, S. Pirbhulal, Z. Luo, K. Muhammad, and N. Z. Zahid, "Toward 6G architecture for energy-efficient communication in IoT-enabled smart automation systems," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5141–5148, Apr. 2021.
[31] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "AI models for green communications towards 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 210–247, 1st Quart., 2022.
[32] U. M. Malik, M. A. Javed, S. Zeadally, and S. U. Islam, "Energy-efficient fog computing for 6G-enabled massive IoT: Recent trends and future opportunities," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14572–14594, Aug. 2022.
[33] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM Conf. Comput. Commun. 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, New York, NY, USA, vol. 1, 1999, pp. 126–134.
[34] L. Cherkasova and M. Gupta, "Analysis of enterprise media server workloads: Access patterns, locality, content evolution, and rates of change," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 781–794, Oct. 2004.
[35] B. Shen, S.-J. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 375–386, Apr. 2004.
[36] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Nov. 2017.
[37] F. X. Guo, F. R. Yu, H. L. Zhang, H. Ji, M. T. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.
[38] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.
[39] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.
[40] J. Du, W. Liu, G. Lu, J. Jiang, D. Zhai, F. R. Yu, and Z. Ding, "When mobile-edge computing (MEC) meets nonorthogonal multiple access (NOMA) for the Internet of Things (IoT): System design and optimization," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7849–7862, May 2021.
[41] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[42] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[43] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 756–764.
[44] R. Xie, Z. Li, J. Wu, Q. Jia, and T. Huang, "Energy-efficient joint caching and transcoding for HTTP adaptive streaming in 5G networks with mobile edge computing," *China Commun.*, vol. 16, no. 7, pp. 229–244, Jul. 2019.

JUNFENG XIE received the B.S. degree in communication engineering from the University of Science and Technology, Beijing, in 2013, and the Ph.D. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, in 2019. From September 2017 to September 2018, he visited Carleton University, Ottawa, ON, Canada, as a Visiting Ph.D. Student. He is currently an Assistant Professor with the North University of China. His research interests include machine learning, content delivery networks, resource management, and wireless networks.