IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 June 2022, accepted 14 July 2022, date of publication 28 July 2022, date of current version 5 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3194512

RESEARCH ARTICLE

Bandit Learning-Based Edge Caching for 360-Degree Video Streaming With Switching Cost

ZHENDONG YU[1], JIAYI LIU[1], (Member, IEEE), CHEN WANG[2], AND QINGHAI YANG[1], (Member, IEEE)
[1]School of Telecommunication Engineering, Guangzhou Institute of Technology, Xidian University, Xian 710071, China
[2]2012 Laboratories, Huawei Technologies, Shenzhen 518129, China

Corresponding author: Jiayi Liu (jyliu@xidian.edu.cn)

This work was supported in part by the National Nature Science Foundation of China under Grant 61971327 and in part by Shaanxi Key Project under Grant 2020ZDLGY05-03.

ABSTRACT Virtual Reality (VR) and Augmented Reality (AR) applications are expected to be a key driver for the 5G network. The need of the MEC support for caching and transcoding of omnidirectional content has been identified as a major topic of research. In this paper, we study the optimal caching problem in an MEC-enabled VR system for tile-based viewport-adaptive 360-degree video streaming. For content caching, the replacement of the cached content inevitably incurs content switching cost, which is largely ignored in caching policy design. In this work, we aim to find the optimal caching policy to optimize the long-term transmission quality by considering both transmission latency and content switching cost. We apply the combinatorial multi-armed bandit (CMAB) theory to solve the problem with no a-priori knowledge on content popularity. Moreover, the CUCB with switching cost (CUCBSC) algorithm is adopted in this scenario. A transformation mechanism is designed to transform the generated single period optimization (SPO) problem into a multiple choice knapsack problem (MCKP). Rigorous theoretical analysis on the performance of the proposed algorithm is also provided. Finally, the effectiveness of the proposed learning based caching policy is confirmed by simulation results in terms of learning, hit-ratio, transmission and content switching delay.

INDEX TERMS 360-degree video streaming, tiling, mobile edge computing, switching cost, multi-armed bandit.

I. INTRODUCTION

Digital immersion offers the merging of the physical world with the digital data world, to create an altered and enhanced environment. It is realized by technologies like virtual reality (VR) and augmented reality (AR). As an essential component of VR/AR systems, 360-degree video provides the immersive experience for end users through streaming VR contents towards VR devices, such as Head Mounted Displays (HMDs). 360-degree video requires high bit-rate, low transmission latency and massive computation resources. Constrained by limited power and computation resources, HMDs cannot fulfill these requirements. Therefore, it is challenging for the current network to support such latency-sensitive, computing-intensive and bandwidth consuming services.

The associate editor coordinating the review of this manuscript and approving it for publication was Nurul I. Sarkar.

Inspired by the fact that the viewport of a VR user, a.k.a. Field of View (FoV), is only about 30% of the panoramic video [1], viewport adaptive streaming is proposed, which effectively reduces the required transmission bit-rates of VR videos [2]. In viewport adaptive streaming, 360-degree videos are delivered with non-homogeneous quality: high-quality for the region of FoV, and low-quality for the remaining part of the video. Furthermore, tile-based 360-degree video [3] is one implementation of viewport-adaptive streaming, which has been adopted by several video vendors. On the basis of the High Efficiency Video Coding (HEVC) encoding standard, the idea is to split the video frame into non-overlapping tiles in space, and to encode them independently. Each tile is offered at multiple quality levels, and they are delivered with proper quality levels with respect to the user's requests.

Furthermore, mobile edge computing (MEC) [4] is recognized as a promising network paradigm to support the streaming of 360-degree videos, which provides caching and computing capabilities on the edge of the network. Specifically, in an MEC-enabled VR system, MEC server can cache popular 360-degree video content to relief the traffic burden of backhaul links and performs computing tasks to transcode the video into different quality versions to meet diverse user requests.

There are some existing works in the literature in edge caching for tile-based 360-degree video streaming. On one hand, some works [5]–[9] study the caching policy in the VR systems with a-priori knowledge on content popularity, which, however, is impractical in the real world. For example, the caching system needs to learn a lot of data in order to obtain the content popularity in advance, which also requires much time cost, or if the content popularity changes over time, it is meaningless to obtain the content popularity in advance. On the other hand, the authors in [10]–[15] adopt the dynamic and learning-based caching replacement to overcome the above shortcoming. However, the replacement of the cached content is neglected in the above works, which limits their practicality in actual usage. For content caching, the cache replacement is also an essential process which inevitably incurs content switching cost. Therefore, taking the content switching cost into account, we adopt an online algorithm based on the multi-armed bandit (MAB) theory to solve the 360-degree video edge caching problem. Moreover, to solve the generated single period optimization (SPO) problem for each time period, a mechanism is designed to transform the SPO into a multiple-choice knapsack problem (MCKP), which could be solved efficiently. Finally, the main contributions of this paper are concluded as follows:

- We study the optimal 360-degree video caching policy in a typical MEC-enabled VR system with no a-priori knowledge on content popularity, where the request latency and content switching cost are both taken into consideration.
- We consider the joint caching and transcoding problem for tile-based 360-degree video for balancing the trade-offs between MEC storage and computation resource consumption. The problem is formulated as a sequential decision making problem, which is a typical explore-exploit (EE) problem.
- The MAB theory is utilized to analyze the joint caching and transcoding problem. Considering the content switching cost, we adopt the CUCB with switching cost (CUCBSC) algorithm to solve the problem and theoretically analyze its non-trivial regret bound.
- Based on the CUCBSC algorithm, by considering the long-tail distribution of content popularity in real world, an improved algorithm (ICUCBSC) is also proposed which speeds up the learning process for practical usage. The effectiveness of the proposed algorithms are confirmed by simulation results.

Section II introduces the related work. In Section III, a typical 360-degree VR caching system is presented. Section IV formulates the problem of minimizing the overall delay containing request delivery latency and content switching cost. In section V, CUCBSC is utilized to handle the problem and its performance is theoretically analyzed. In Section VI, the improved CUCBSC is proposed and simulation results demonstrate the effectiveness of the proposed algorithms. Finally Section VII summarizes the paper.

II. RELATED WORK

A. TILE-BASED 360-DEGREE VIDEO STREAMING

To improve the transmission efficiency of 360-degree video streaming, many schemes [16]–[21] based on the tile-based 360-degree video delivery have been proposed. In [16], the authors compare the performance of two viewport-adaptive streaming methods: tile-based method and truncated square pyramid (TSP) projection. The simulation results show that the tile-based approach has slightly lower streaming performance, but saves much more transcoding time and storage space. In [17], the authors present an FoV rendering solution on the edge of networks, which is designed to optimize the bandwidth and latency during the delivery of 360-degree videos. In [18], a QoE-driven viewport adaptation system is proposed based on a probabilistic approach to prefetch tiles, which can achieve a high viewport PSNR. The authors in [19] study an optimal multicast of tile-based 360° video, aiming to optimize the transmission delay and power allocation to minimize the average transmission energy. In [20], the authors propose an optimal bandwidth allocation scheme in wireless VR delivery system to minimize the maximum user transmission and computation latency. In [21], a deep reinforcement learning-based rate adaptation algorithm is presented to handle the delivery of 360-degree video streaming, in order to maximize the QoE of users by adapting the transmitted video quality to the time-varying network conditions.

However, the above works focus on the delivery of 360-degree video streaming. By caching 360-degree videos in MEC, the efficiency of the VR delivery systems can be further improved.

B. EDGE CACHING AND ONLINE CACHING FOR 360-DEGREE VIDEO STREAMING

MEC caching can improve the efficiency of 360-degree video delivery. Consequently, a number of works have focused on 360-degree video edge caching. In [5], the authors propose a scheme of joint collaborative caching and delivery of 360-degree videos, which shows the benefits of MEC caching. Comparing to [5], transcoding is also considered in the 360-degree video edge caching problem in [6]. In [7], the authors formulate the optimization problem based on the tradeoff of computing, caching and communication (3C) to minimize the average transmission rate in an MEC-enabled VR delivery framework. Dang [8] proposes a mobile

VR delivery framework in F-RANs and formulates a similar 3C tradeoff problem to minimize the average latency. In [9], a novel VR delivery system is proposed, which integrates scalable multi-layer 360-degree video tiling, edge computing and caching, and optimal resource allocation of viewport-adaptive rate-distortion.

The above studies consider that the distribution of video content popularity is known in advance, which is impractical in most cases. To handle this limitation, many works use dynamic caching replacement or online algorithms. Specifically, in [10], the authors think it is impractical to obtain all of the priori knowledge about user requests in real scenario and thus propose a view synthesis based online caching algorithm called MaxMinDistance. And [11] is devoted to solve the 3D video caching problem using *Markov Decision Process* (MDP), where *Depth Image Based Rendering* (DIBR) is supported to allow the view synthesis. Machine learning [12]–[15] is also applied to seek the optimal caching policy with no a-priori knowledge on video content popularity. Based on the tradeoff between the transmission delay and energy consumption, the authors in [12] propose a *long short-term memory auto-encoder deep deterministic policy gradient* (LSTMAE-DDPG) algorithm to solve the VR video caching problem. Taking the transcoding of VR into the account, the authors in [13] formulate the collaborative caching of edge MBS and SBS as a networked multi-agent MDP, and propose a *multi-agent Actor-Critic* algorithm to minimize the latency of delivery preparation. In [14], the authors study the proactive 3C resource allocation problem for supporting VR videos at network edge, and formulate the problem as an MDP in terms of each kind of resource. To handle the formulated problem, the DDPG algorithm is introduced and learn the effective policy. In [15], the authors propose a reactive caching scheme that assumes unknown popularity of videos and viewports. Specifically, they first formulate the 360° videos edge caching as an MDP, and then determine the optimal caching placement using the *Deep Q-Network* (DQN) algorithm.

C. MAB-BASED CACHING STRATEGY

As one typical reinforcement learning method, MAB [22] is a common online learning method for caching replacement. The authors in [23] formulate the optimal content placement in a small cell base station as a combinatorial MAB (CMAB) problem. And they further take the switching cost into account and propose the CUCB with switching cost (CUCBSC) algorithm [24]. In [25], the authors propose an online service placement scheme in an MEC-enabled system with no a-priori knowledge of server demand and network states, which is analysed by CMAB. In [26], the authors formulate the wireless caching problem as a contextual MAB problem, which considers that similar content preference is often shared by the users with similar features. In [27], the authors utilize CMAB to formulate the optimal caching policy for 360-degree videos to co-optimize users' quality of experience (QoE) and MEC energy consumption.

For content caching, content replacement should be considered, which inevitably incurs content switching cost. However, such switching cost is often neglected for 360-degree video caching policy design. By joint considering the transmission and switching cost, we utilize the MAB theory to handle the 360-degree video caching policy problem through the CUCBSC algorithm. Typically, we try to balance the tradeoff between caching and computation resource consumption to minimize both content transmission and switching cost. Moreover, for the generated SPO, a transformation mechanism is proposed to solve the SPO by MCKP algorithm, which improves the efficiency of the proposed algorithm. Finally, we rigorously analyze the regret bound of the proposed algorithm.

III. SYSTEM MODEL

A. SYSTEM OVERVIEW

The 360-degree VR caching system is illustrated in Figure 1. The user plane function (UPF) receives the requests from end-users and forwards the requests to the MEC or the remote server. The MEC server is deployed at the edge of network, which directly serves the users by providing certain caching and computing capabilities. The maximum caching capacity of the MEC server is denoted by C and the CPU-cycle frequency is f (in $cycles/s$). Considering that the transmission latency of the backhaul link is generally greater than that from the MEC server to end-users [28], we mainly consider the transmission latency of the backhaul link.

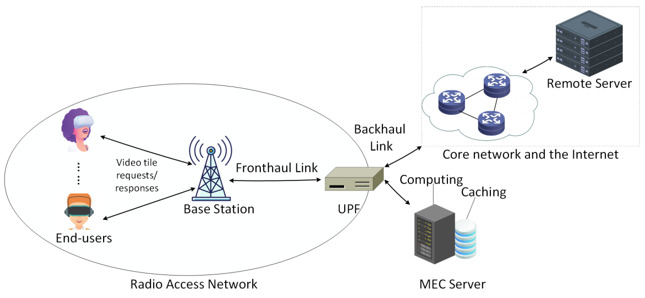

FIGURE 1. MEC-based 360° video caching system.

Note that, 360-degree video tiles stored in the remote server are 2-dimensional (2D). To ensure an immersive VR experience, the projection of 2D into 3D shown is essential [7]. In order to make full use of both the caching and computing resources of MEC servers, we consider that MEC can perform transcoding and projection. Therefore, one 2D tile at highest quality level, called raw tile, can be transformed into one 3D tile at any quality level by MEC.

A set of V 360-degree videos indexed by $v \in \{1, 2, \ldots, V\}$ are stored in the remote server in the form of 2D raw video with highest quality level. Each video is split into M chunks indexed by $m \in \{1, 2, \ldots, M\}$ and each chunk is divided spatially into N tiles indexed by $n \in \{1, 2, \ldots, N\}$. Furthermore, we allow the MEC server to cache both raw and 3D tiles. Each 3D tile is offered at Q quality levels indexed

TABLE 1. Notations.

Symbol	Meaning
V, v	the total number of 360-degree videos, the v-th 360-degree video
M, m	the total number of chunks in a 360-degree video, the m-th chunk
N, n	the total number of tiles in a chunk, the n-th tile
Q, q	the total number of quality levels, the q-th quality level
U^t, u	the total number of users in time period t, the u-th user
l_{vmnq}	the n-th tile of m-th chunk in video v with q-th quality level
s_{vmnq}	the size of tile l_{vmnq}
x^t_{vmnq}	0-1 caching variables
C	the cache capacity of the MEC server
f	the CPU-cycle frequency of the MEC server
w	the CPU cycles consumed by the MEC processing one bit
r	the minimum allowable transmission rate of backhaul link

by $q \in \{1, 2, \ldots, Q\}$, while we use $q = 0$ to indicate that this is a raw tile for convenient expression. Denote tile n encoded into the q-th quality level in the m-th chunk of video v with l_{vmnq}. Besides, we define the size of l_{vmnq} as s_{vmnq}. Due to the fact that our system is a dynamic system, we define the total number of users watching the 360-degree videos in time period t as U^t.

B. CACHING MODEL

For 360-degree video caching in the MEC server, we introduce 0-1 caching decision variables x^t_{vmnq}, where $x^t_{vmnq} = 1$ represents that l_{vmnq} is stored in the MEC server in time period t and $x^t_{vmnq} = 0$ otherwise.

For the limited caching capacity, the size of data stored in the MEC server cannot exceed C. Thus, we can obtain the capacity constrain as follows:

$$\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=0}^{Q} s_{vmnq}x^t_{vmnq} \le C, \quad \forall t \tag{1}$$

Caching the raw tiles in MEC consumes less storage resources, but brings higher transcoding delay. On the contrary, caching the 3D tiles reduces delay by sacrificing more storage. Hence, there is a tradeoff between caching and computing resource consumption. To balance the two, we can either cache a raw tile, or cache multiple 3D tiles at different quality levels which are transcoded from the former. Such trade-off relationship between storage and computing resources can be expressed as:

$$x^t_{vmnq} + x^t_{vmn0} \le 1, \ \forall t, \ \forall v, \ \forall m, \ \forall n, \ \forall q(q \neq 0) \tag{2}$$

C. REQUEST LATENCY

The VR device sends the user's requests to the MEC server, where we can denote the request of user u for tile l_{vmnq} $(q \neq 0)$ in time period t by d^t_{uvmnq}. Furthermore, we can obtain the request number of l_{vmnq}, i.e., $\tau^t_{vmnq} = \sum_{u=1}^{U^t} d^t_{uvmnq}$.

To meet the request d^t_{uvmnq}, there exist three situations as follows:

- When l_{vmnq} can be obtained from the MEC server, i.e., $x^t_{vmnq} = 1$, user u can meet the request directly via the MEC.
- When l_{vmn0} is stored in the MEC server, i.e., $x^t_{vmn0} = 1$, the MEC server transcodes the raw tile into the corresponding form, and then transmits l_{vmnq} to the user. The delay of transcoding can be expressed as $x^t_{vmn0}\frac{\omega|s_{vmnq}-s_{vmn0}|}{f}$, where ω (in $cycle/bit$) is the CPU cycles when the MEC server processes one bit.
- Otherwise when $x^t_{vmnq} = 0$ and $x^t_{vmn0} = 0$, the remote server transmits the requested raw tile l_{vmn0} to the MEC server. Then l_{vmn0} is transcoded into l_{vmnq}, and delivered to the user. The total delay containing transmission and transcoding can be expressed as $(1 - x^t_{vmnq} - x^t_{vmn0})(\frac{s_{vmn0}}{r} + \frac{\omega|s_{vmnq}-s_{vmn0}|}{f})$, where r is the minimum allowable transmission rate of backhaul link.

Therefore, we can obtain the downloading delay D_R^t in time period t, which is expressed as follows:

$$\begin{aligned} D\_R^t = &\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=1}^{Q} \tau^t_{vmnq}(\frac{\omega|s_{vmnq}-s_{vmn0}|}{f}x^t_{vmn0} \\ &+(\frac{s_{vmn0}}{r}+\frac{\omega|s_{vmnq}-s_{vmn0}|}{f}) \\ &\times(1-x^t_{vmnq}-x^t_{vmn0})) \end{aligned} \tag{3}$$

D. SWITCHING COST

Since the MEC server may cache different content at different times, the cost of switching content also needs to be considered. We define the switching cost as the required delay when the content cached in the MEC is switched.

When one tile needs to be switched into the MEC cache, i.e., the tile needs to be stored at the current time but not in the cache at the previous time, there also exist three situations:

- If the tile is a raw tile, i.e., l_{vmn0}, the MEC only needs to request the corresponding tile from the remote server. Its transmission delay is expressed as s_{vmn0}/r.
- If the tile is one 3D tile l_{vmnq} and its corresponding raw tile is stored at the previous time, the raw tile needs to be transcoded into the corresponding quality level by the MEC server and deleted from the cache, where the delay consumed by transcoding is $x^t_{vmnq}x^{t-1}_{vmn0}(1-x^{t-1}_{vmnq})\frac{\omega|s_{vmnq}-s_{vmn0}|}{f}$.
- If the tile is one 3D tile l_{vmnq} and its corresponding raw tile is not stored at the previous time, the raw tile needs to be transmitted to the MEC and transcoded into l_{vmnq}. So the total delay is $x^t_{vmnq}(1-x^{t-1}_{vmn0})(1-x^{t-1}_{vmnq})(\frac{s_{vmn0}}{r}+\frac{\omega|s_{vmnq}-s_{vmn0}|}{f})$.

Therefore, the switching delay D_S^t in time period t can be expressed as:

$$\begin{aligned} D\_S^t = &\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}(x^t_{vmn0}(1-x^{t-1}_{vmn0})\frac{s_{vmn0}}{r} \\ &+\sum_{q=1}^{Q}(x^t_{vmnq}(1-x^{t-1}_{vmnq})((1-x^{t-1}_{vmn0})\frac{s_{vmn0}}{r} \\ &+\frac{\omega|s_{vmnq}-s_{vmn0}|}{f})) \end{aligned} \tag{4}$$

IV. PROBLEM FORMULATION

Our goal is to minimize the overall delay containing request latency D_R^t and switching cost D_S^t. So we can formulate the problem as follows:

$$P1: \min_{x} \ D = \sum_{t=1}^{T}(D\_R^t + D\_S^t)$$
$$s.t: \ (1), (2) \tag{5}$$

where T represents the limited time range, and the overall delay D can be converted into:

$$D = \sum_{t=1}^{T}\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}(x_{vmn0}^{t}(1 - x_{vmn0}^{t-1} - \sum_{q=1}^{Q}\tau_{vmnq}^{t}) \cdot \frac{s_{vmn0}}{r} + \sum_{q=1}^{Q}x_{vmnq}^{t}(((1 - x_{vmnq}^{t-1})(1 - x_{vmn0}^{t-1}) - \tau_{vmnq}^{t})\frac{s_{vmn0}}{r} + (1 - x_{vmnq}^{t-1} - \tau_{vmnq}^{t}) \cdot \frac{\omega|s_{vmnq} - s_{vmn0}|}{f}) + \sum_{q=1}^{Q}\tau_{vmnq}^{t}(\frac{s_{vmn0}}{r} + \frac{\omega|s_{vmnq} - s_{vmn0}|}{f})) \tag{6}$$

where it is obvious that some terms are independent of caching variables, which are as follows:

$$Cons = \sum_{t=1}^{T}\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=1}^{Q}\tau_{vmnq}^{t}(\frac{s_{vmn0}}{r} + \frac{\omega|s_{vmnq} - s_{vmn0}|}{f}) \tag{7}$$

Therefore, the objective function of $P1$ can be further simplified by deleting these constant terms. And then the minimization problem can be turned into a maximization problem. We firstly define a system reward r^t in time period t, which is expressed as:

$$r^t = \sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}(x_{vmn0}^{t}(\sum_{q=1}^{Q}\tau_{vmnq}^{t} + x_{vmn0}^{t-1} - 1)\frac{s_{vmn0}}{r} + \sum_{q=1}^{Q}x_{vmnq}^{t}((\tau_{vmnq}^{t} - (1 - x_{vmnq}^{t-1})(1 - x_{vmn0}^{t-1})) \cdot \frac{s_{vmn0}}{r} + (\tau_{vmnq}^{t} + x_{vmnq}^{t-1} - 1)\frac{\omega|s_{vmnq} - s_{vmn0}|}{f})) \tag{8}$$

Then our problem can be converted into:

$$P2: \max_{x} \ \sum_{t=1}^{T} r^t$$
$$s.t: \ (1), (2) \tag{9}$$

where $P2$ is equivalent to $P1$, which is a sequential decision making problem, and its goal is to maximize the long-term accumulated system gain over T time periods.

We observe that problem $P2$ can evolve into a single period optimization (SPO) problem if tile popularity is known.

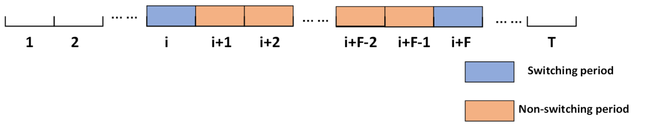

FIGURE 2. Swiching and non-switching periods.

In this case, the problem can be tackled in the initial time period and then the caching content does not need to be switched in the following periods. However, the assumption of known content popularity is impractical. To tackle the problem $P2$, the estimation of tile popularity is necessary in each time period, which can be used as the parameter basis for MEC cache. According to the estimation of tile popularity, there are two trends in the MEC caching. On one hand, the MEC server prefers to store what are estimated to be popular with higher priority in order to maximize the system gain, which reflects the VR system's ability of exploitation. On the other hand, the MEC intends to store those less stored, aiming at making the estimation of tile popularity increasingly accurate and the popular tiles not to be missed as far as possible, which is the embodiment of the VR system's exploration. Obviously, it is a typical exploit-explore (EE) problem.

The MAB theory is applied to solve the EE problems [22]. Because MEC needs to cache multiple tiles, i.e., to play multiple base arms in each time period, traditional MAB method is not suitable where one base arm is played in one iteration. So the combinatorial MAB (CMAB) framework [29] is utilized to analyze our model, where the set of base arms played in time period t is called a super arm represented by S^t.

V. ONLINE LEARNING-BASED CACHING ALGORITHM

A. CUCBSC ALGORITHM

In our model, each tile l_{vmnq} is a base arm, that is, when a tile is stored in the MEC server, the corresponding base arm is played and otherwise the base arm is not played. So the total number of base arms is $L = V \times M \times N \times (Q + 1)$. Because $P2$ is a sequential decision making problem, a caching decision is made on the MEC server, i.e., a super arm is played in each time period, and then a comprehensive reward defined by the objective function which reflects the estimated tile popularity can be obtained. Considering the existence of switching cost in the objective function, we introduce the CUCB with switching cost (CUCBSC) [24] algorithm to handle $P2$. The detail of CUCBSC is shown in **Algorithm 1**.

There are two kinds of time periods: switching periods and non-switching periods. In a switching period arms can be switched, that is, the content replacement is allowed in such a time period. In contrast, in a non-switching period, the cached contents of MEC remain unchanged, which is designed to effectively reduce the switching cost. As shown in Figure 2, there are $F - 1$ non-switching periods between two adjacent switching periods.

Specially in switching periods, *Oracle* is used to choose a super arm S in each time period. In view that problem $P2$ can evolve into a SPO problem in each time period,

Algorithm 1 CUCBSC

```
Input: For each arm $l_{vmnq}$, maintain: (1) $T_{vmnq}$ as the number of times arm $l_{vmnq}$ has been played; (2) $\bar{p}_{vmnq}$ as the average reward of arm $l_{vmnq}$ so far.
Output: Super arm $S^t$ for each period.
1:  Initialize: Play each base arm at least once, and update $T_{vmnq}$ and $\bar{p}_{vmnq}$.
2:  Set $t = L$.
3:  Switching period:
4:  $\forall l_{vmnq}$, set $\widetilde{p}_{vmnq} = \frac{\bar{p}_{vmnq}}{p_{max}} + \sqrt{\frac{3\ln t}{2T_{vmnq}}}$, where $p_{max} = max\{\bar{p}_{vmnq}\}$.
5:  Set $S^t = Oracle(\widetilde{P})$, where $\widetilde{P} = \{\widetilde{p}_{vmnq}|\forall l_{vmnq}\}$.
6:  Play $S^t$ and update $T_{vmnq}$ and $\bar{p}_{vmnq}$, $\bar{p}_{vmnq} = \frac{\bar{p}_{vmnq}T_{vmnq}+p^t_{vmnq}}{T_{vmnq}+1}$, $T_{vmnq} = T_{vmnq} + 1, \forall l_{vmnq} \in S^t$.
7:  set $t = t + 1$.
8:  Non-switching period:
9:  for $i = 1 \to F - 1$ do
10:     $\forall l_{vmnq} \in S^t$, $\bar{p}_{vmnq} = \frac{\bar{p}_{vmnq}T_{vmnq}+p^t_{vmnq}}{T_{vmnq}+1}$, $T_{vmnq} = T_{vmnq} + 1$.
11:     set $S^{t+1} = S^t$, $t = t + 1$ and $i = i + 1$.
12: end for
13: go to Line 4.
```

TABLE 2. Illustration of the encoding scheme.

$x^t_{vmn0}x^t_{vmn1}x^t_{vmn2}\cdots x^t_{vmnQ}$	j	$R_{vmn}(j)$
000...0	0	$\varnothing$
000...1	1	$\{Q\}$
...	...	...
011...1	$2^Q - 1$	$\{1, 2, ..., Q\}$
100...0	2^Q	$\{0\}$

the actual purpose of *Oracle* is to find an optimal caching solution to the SPO problem according to parameters $\widetilde{P}$ in line 5 of **Algorithm 1** which reflects the currently estimated tile popularity.

Now, we explain that the SPO problem can be transformed into a multiple-choice knapsack problem (MCKP) [30]. Specifically, when $x^t_{vmn0} = 1$, $x^t_{vmnq}(\forall q(q \neq 0))$ has to be equal to 0, and when $x^t_{vmn0} = 0$, $x^t_{vmnq}(\forall q(q \neq 0))$ can take any value between 0 and 1 within the feasible region. We utilize a binary coding scheme to enumerate all possible situations of $x^t_{vmn0}x^t_{vmn1}x^t_{vmn2}\ldots x^t_{vmnQ}$. And after the binary coding, the corresponding decimal number is denoted with j, which is shown in Table 2. Furthermore, we utilize the set $R_{vmn}(j)$ to record the quality levels of which the value is equal to 1 among $x^t_{vmn0}, x^t_{vmn1}, x^t_{vmn2}, \ldots, x^t_{vmnQ}$, i.e., $R_{vmn}(j) = \{q|x_{vmnq} = 1\}$.

Therefore, we can regard the SPO problem as an MCKP, where there are $V * M * N$ classes and each class has $2^Q + 1$ options. And the SPO problem can be converted into:

$$P3: \max_{y} \sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{j=0}^{2^Q} y_{vmnj}p_{vmnj}$$

$$\begin{aligned} s.t.\quad & C1: \sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{j=0}^{2^Q} y_{vmnj}w_{vmnj} \leq C \\ & C2: \sum_{j=0}^{2^Q} y_{vmnj} = 1, \quad \forall v, \forall m, \forall n \\ & C3: y_{vmnj} \in \{0, 1\}, \quad \forall v, \forall m, \forall n, \forall j \end{aligned} \tag{10}$$

where the binary decision variable y_{vmnj} can be interpreted into the corresponding caching decision, of which the binary coding $x^t_{vmn0}x^t_{vmn1}x^t_{vmn2}\ldots x^t_{vmnQ}$ is numerically equivalent to j. The profit p_{vmnj} and weight w_{vmnj} are defined respectively as follows:

$$\begin{aligned} p_{vmnj} &= \sum_{q \in R_{vmn}(j)} p'_{vmnq} \\ w_{vmnj} &= \sum_{q \in R_{vmn}(j)} s_{vmnq} \end{aligned} \tag{11}$$

where p'_{vmnq} can be obtained from Eq(8), i.e., when $q = 0$, $p'_{vmn0} = (\sum_{q=1}^{Q} \tau^t_{vmnq} + x^{t-1}_{vmn0} - 1)\frac{s_{vmn0}}{r}$, and otherwise, $p'_{vmnq} = (\tau^t_{vmnq} - (1 - x^{t-1}_{vmnq})(1 - x^{t-1}_{vmn0}))\frac{s_{vmn0}}{r} + (\tau^t_{vmnq} + x^{t-1}_{vmnq} - 1)\frac{\omega(s_{vmnq}-s_{vmn0})}{f}$.

Due to the fact that MCKP is NP-hard, we utilize a fully polynomial time approximation scheme (FPTAS) [31] to handle $P3$, which is shown in **Algorithm 2**. Please note that **Algorithm 2** is utilized as the *Oracle* in **Algorithm 1**, which returns the caching decision and the corresponding super arm. Define P^* as the optimal value of $P3$, and P as the value induced by FPTAS. It has been proved that $P \geq (1 - \epsilon)P^*$, where $0 < \epsilon \leq 1$, i.e., the solution is $(1-\epsilon)$-optimal. Besides, the total time complexity of FPTAS is $O(V^2 \times M^2 \times N^2 \times (2^Q + 1)/\epsilon)$. More detailed proof and algorithm steps can be found in [31]. Therefore, we can use FPTAS to obtain the approximate solution of the SPO problem, i.e., to learn which tiles to be stored in switching periods.

Algorithm 2 FPTAS for Solving Problem 3

```
Input: Profit $\widetilde{P} = \{\widetilde{p}_{vmnq}|\forall l_{vmnq}\}$
Output: Caching decision $Y$
1: Transform the MCKP ($P3$) into the Continuous MCKP (CMCKP), where the constraint C3 is relaxed as: $0 \leq y_{vmnj} \leq 1, \forall v, \forall m, \forall n, \forall j$.
2: Solve the CMCKP with Dyer-Zemel Algorithm. Then the solution is $Y_0$ and its corresponding profit is $P_0$.
3: Set the scale factor $K = \epsilon P_0/L$.
4: Replace $\widetilde{p}_{vmnj}$ with $a_{vmnj} = \lfloor\frac{\widetilde{p}_{vmnj}}{K}\rfloor$ in MCKP.
5: Use the exact pseudo-polynomial time dynamic programming algorithm to obtain a solution $Y_1$ to MCKP and its corresponding profit $P_1$.
6: Return the more profitable of $Y_0$ and $Y_1$.
```

B. REGRET BOUND

In this section, we theoretically analyze the performance of the CUCBSC algorithm. Firstly, the regret is defined as the difference between the accumulated reward of the optimal caching policy and that of our policy which caches the content according to the approximation algorithm based on the knowledge of the currently estimated tile population. Assume that we can obtain at least α fraction of the optimal value of $P3$ with a success rate of β by the approximation algorithm. Due to the division of time periods in CUCBSC, we divide the regret into the sampling regret and the switching regret, which are caused by not knowing the tile popularity and switching the caching content, respectively. The sampling regret until time period t is defined as follows:

$$R\_R(t) = t\alpha\beta r_{opt} - \mathbb{E}[Cons - \sum_{k=1}^{t} D\_R^k] \tag{12}$$

where $t\alpha\beta r_{opt}$ is the accumulated reward of the optimal caching which is the solution to the SPO problem solved by **Algorithm 2** until time period t. The switching regret is defined as the expected total switching cost until time period t, which is expressed as:

$$R\_S(t) = \mathbb{E}[\sum_{k=1}^{t} D\_S^k] \tag{13}$$

Then, the regret of the CUCBSC algorithm is the sum of the sampling regret and switching regret:

$$R(t) = R\_R(t) + R\_S(t) \tag{14}$$

According to the above definition, we present a regret bound for the CUCBSC algorithm, of which the complete proof is shown in Appendix.

Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_L)$ be the vector of expectations of all base arms, where L is the number of all base arms. According to [29], there exists a strictly increasing function $f(\cdot)$ which is called *bounded smoothness function*, and for any two expectation vectors $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$, we have $|r_{\boldsymbol{\mu}} - r_{\boldsymbol{\mu}'}| \leq f(\Lambda)$ if $max_{i\in\mathcal{L}}|\mu_i - \mu_i'| \leq \Lambda$, where i represents the i-th base arm and $\mathcal{L}$ is the set of all base arms. A super arm is bad if the reward satisfies the following formula: $r_{\boldsymbol{\mu}}(S) < \alpha \cdot r_{opt}$, and its corresponding time period is called a bad period. We define $\mathcal{S}_B = \{S | r_{\boldsymbol{\mu}}(S) < \alpha \cdot r_{opt}\}$ as the set of bad super arms. Similarly, $\mathcal{S}_G = \{S | r_{\boldsymbol{\mu}}(S) >= \alpha \cdot r_{opt}\}$ is defined as the set of good super arms. Furthermore, we define $\Delta^i_{max} = \alpha \cdot r_{opt} - min\{r_{\boldsymbol{\mu}}(S) | S \in \mathcal{S}_B, i \in S\}$, $\Delta^i_{min} = \alpha \cdot r_{opt} - max\{r_{\boldsymbol{\mu}}(S) | S \in \mathcal{S}_B, i \in S\}$, and $\Delta_{max} = max_{i\in\mathcal{L}}\Delta^i_{max}$, $\Delta_{min} = min_{i\in\mathcal{L}}\Delta^i_{min}$. We denote a counter for each base arm which is updated in each bad period by $N_{i,t}$. In any bad period t, we choose the i-th base arm which has the smallest value of $N_{i,t-1}$ in S^t, and set $N_{i,t} = N_{i,t-1}+1$. Specially, when there exist multiple choices to update $N_{i,t}$, only one of the counters is chosen arbitrarily and updated. Based on the above definition, $\sum_{i\in\mathcal{L}} N_{i,t}$ is the number of bad periods until time period t.

Lemma 1: The expected value of the counter $N_{i,t}$ is denoted by $\overline{N}_t = \mathbb{E}[\sum_{i\in\mathcal{L}} N_{i,t}]$, which is given by:

$$\overline{N}_t \leq (1-\beta)bF + L(\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2}) \tag{15}$$

where b is the number of switching periods until time period t, and F is related to the number of continuous non-switching periods.

On the basis of **Lemma 1** and (12), we can obtain the bound of the sampling regret.

Theorem 1: The sampling regret of the CUCBSC algorithm is bounded by:

$$R\_R(t) \leq (\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2})L\Delta_{max} \tag{16}$$

In order to bound the switching regret, we introduce $Sw_B(t)$ to count the number of switches to super arms in $\mathcal{S}_B$.

Lemma 2: The number of switches to super arms in $\mathcal{S}_B$ is at most:

$$\begin{aligned} Sw_B(t) \leq (1-\beta)b + \frac{L}{F}(\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2} \\ + \frac{6\ln(1+\frac{F-1}{L+1})}{(f^{-1}(\Delta_{min}))^2} - 1) \end{aligned} \tag{17}$$

Theorem 2: The switching regret of the CUCBSC algorithm is bounded by:

$$R\_S(t) \leq Sw_{\mathcal{B}}(t) \cdot 2(C_u - C_l) + L \cdot C_u + b \cdot C_l \tag{18}$$

where C_u is the maximum cost of switching between super arms, and C_l is the maximum cost of switching between good super arms.

Assume that there exists a unique optimal solution to the SPO problem, and **Algorithm 2** can always find the optimal solution, where $\alpha = \beta = 1$ and $C_l = 0$. In this case, we have the following theorem which provides the regret bound of the CUCBSC algorithm.

Theorem 3: When there exists a unique optimal solution to the SPO problem, where $\alpha = \beta = 1$ and $C_l = 0$, the regret bound of the CUCBSC algorithm is given by:

$$\begin{aligned} R(t) \leq L(\frac{6\ln t}{(f^{-1}(\Delta_{min}))^2} + \frac{F\pi^2}{3} + F)(\Delta_{max} + \frac{2C_u}{F}) \\ + \frac{2LC_u}{F}(\frac{6\ln(1+\frac{F-1}{L+1})}{(f^{-1}(\Delta_{min}))^2} - 1) + LC_u \end{aligned} \tag{19}$$

Note that, the regret in **Theorem 3** grows logarithmically in t. Furthermore, if we don't take the switching regret into account and $F = 1$, i.e., each time is a switching period, (16) is equal to the regret bound of CUCB. **Theorem 3** is an extension of the CMAB theory [29].

VI. SIMULATION AND PERFORMANCE ANALYSIS

A. SIMULATION SETTING

In this section, numerical simulations are presented to evaluate the performance of CUCBSC. Regarding the content, a 360-degree video library is set to contain $V = 20$ files. The length of each 360-degree video is set to $10s$, and each video is split into $M = 5$ chunks (i.e. each chunk is set to $2s$ and contains 60 frames with 30fps [13]). Each chunk is divided into $N = 24$ tiles. Each tile can be encoded at two resolutions: 640×540 and 320×270 pixels, that is, $Q = 2$. So the resolution of complete 360-degree video is 3840×2160 at high quality, and 1920 × 1080 at low quality [32]. And the size of one tile is set to 12Mbits at high quality, and 4 Mbits at low quality. Due to the fact that the ratio of the tile size of 3D to that of 2D is usually set to $\lambda \geq 2$ [7], the size of one raw tile is set to 6Mbits. Both the video popularity and chunk popularity follow the Zipf distribution with parameter 0.8, while the popularity of tiles in the same chunk follows the uniform distribution [33]. The preference of quality level is set to be random [6]. The number of users ranges from 50 to 100. For the MEC server, the cache capacity of the MEC server is $C = 20$G, and the CPU-cycle frequency is $f = 5$GHz. When the MEC server processes one bit, the CPU cycles consumed is $\omega = 10$ [7]. The minimum allowable transmission rate of the backhaul link is set to $r = 640$Mbps [10]. Lastly, in the CUCBSC algorithm, F is set to 10 [24], i.e., there are 9 non-switching periods between two adjacent switching periods.

B. BASELINE ALGORITHMS

To better evaluate the performance of the proposed algorithms, we introduce the following algorithms for comparison:

- **LRU algorithm**. The LRU algorithm is a common replacement algorithm. Considering the particularity of the established model, we make the MEC server cache only 3D tiles in LRU.
- **LFU algorithm**. The LFU algorithm is another common replacement algorithm. Similarly to LRU, MEC only caches 3D tiles.
- **CUCB algorithm**. The CUCB algorithm is introduced in [29], where the switching cost is not taken into account.
- **Cons-UCBSC algorithm**. The Cons-UCB algorithm is another UCB algorithm which is introduced in [28] to solve the optimal video caching problem at mobile edges. There are two main differences between Cons-UCB and CUCB: (1) the perturbation term is set as $\widetilde{p}_{vmnq} = \frac{\overline{p}_{vmnq}}{p_{max}} + \sqrt{\frac{2 \ln Ct}{T_{vmnq}}}$, where C is equal to the MEC cache capacity. (2) the approach to choose a super arm in each iteration is to sort all the arms in descending order by the value of $\widetilde{p}_{vmnq}$, and to play arms in this order until the MEC caching capacity is full. In view that problem $P3$ is MCKP, some adjustments are made on the above approach: firstly we select one choice with larger p_{vmnq} for each class, and then repeat the operations of (2) for selected choices. Similarly to CUCBSC, Cons-UCBSC takes the switching cost into account based on Cons-UCB.
- **ICUCBSC algorithm**. In light of the references [10], [33], the popularity of VR content follows the Zipf-like distribution. Due to the fact that popular tiles are in the minority of total tiles in the Zipf-like distribution, the over exploration of the perturbation in line 5 of CUCBSC is unfriendly to the Zipf-like distribution, which may result in constant caching of unpopular tiles. Although the estimated tile popularity is increasingly accurate as time goes on, CUCB may take quantities of iterations to learn an optimal caching decision, of which the time cost is unacceptable. Therefore, similarly to the reference [23], we propose an improved CUCBSC (ICUCBSC), where the perturbation in line 5 of **Algorithm 1** is converted into:

$$\widetilde{p}_{vmnq} = \frac{\overline{p}_{vmnq}}{p_{max}} + \sqrt{\frac{3 \ln (U_{max} t)}{2 U_{max} T_{vmnq}}} \tag{20}$$

where U_{max} is the maximum number of users served by MEC. When parameter U_{max} is large, the exploitation of the algorithm is promoted, which can effectively reduce the probability of caching unpopular tiles when the Zipf-like distribution is skewed.
- **Optimal algorithm**. In the optimal algorithm, it is assumed that the tile popularity is known in advance, which can be substituted into the SPO problem, i.e. $P3$. Therefore, **Algorithm 2** can be utilized to optimally solve the problem and achieving the optimal caching strategy.
- **DRL algorithm**. [12] proposes a method based on deep reinforcement learning (DRL) which jointly considers the deterministic offloading and the dynamic caching replacement of 360° videos. And the cache part of this method is chosen as one of algorithms to compare. Due to the difference of the models, to make a comparable experiment, we adopt their MDP model and replace the prediction of the unknown tile popularity with historical requests in the model states.

C. PERFORMANCE ANALYSIS

Firstly, we evaluate the average learning regret of the four UCB algorithms through 1×10^6 iterations. Then we change the MEC cache size, and analyze the performance of all algorithms, including two replacement algorithms, in terms of average request delay, total switching delay and hit ratio. In order to better reflect the learning performance, we set the number of iterations as 1×10^5, which is much smaller than that in the former training. And data are collected in the last 1000 times.

1) AVERAGE LEARNING REGRET

Figure 3 shows the average learning regret of these UCB algorithms, where the average regret of each UCB algorithm is on the decrease and trends to stabilize. It is obvious that the CUCBSC performs better than CUCB and Cons-UCBSC. Furthermore, the learning regret of ICUCBSC is much smaller than CUCBSC, and it is practically steady after few time periods, which proves that the improvement of ICUCBSC is effective and speeds up the learning process.

Figure 4 and Figure 5 show the switching regret and the sampling regret, respectively. We observe that Cons-UCBSC have smaller switching regret than CUCB, which results from the fact that Cons-UCBSC takes the switching cost into account. On the contrary, due to the limitation of the approach

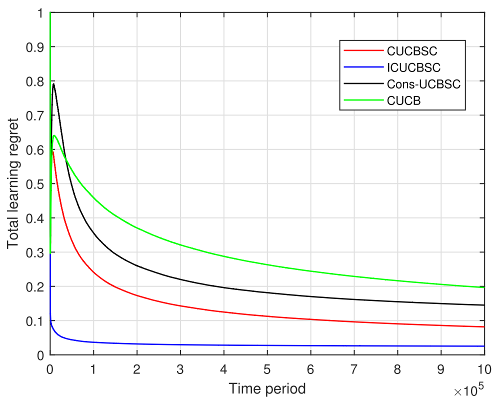

FIGURE 3. Average total learning regret of different UCB algorithms.

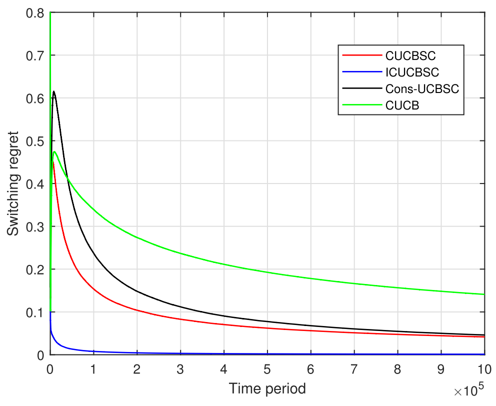

FIGURE 4. Average switching regret of different UCB algorithms.

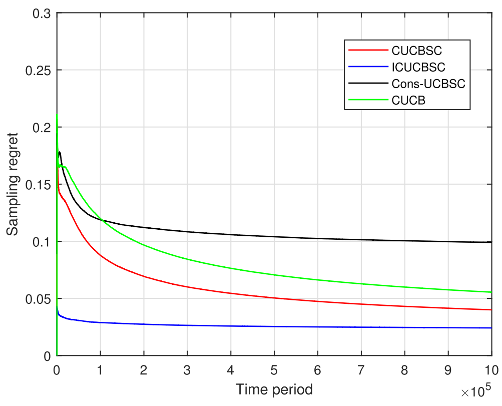

FIGURE 5. Average sampling regret of different UCB algorithms.

to solve the SPOs, the sampling regret of Cons-UCBSC is larger than CUCB.

2) REQUEST DELAY AND SWITCHING DELAY

Figure 6 shows the influence of MEC cache size on the average request delay under eight algorithms. It is observed that the average request delay per user is on the decrease with the increase of MEC cache capacity. Obviously, Optimal and ICUCBSC perform best, followed by CUCBSC. And the performance of optimal algorithm and ICUCBSC is almost

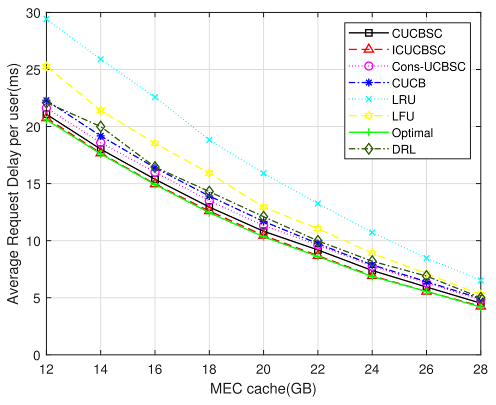

FIGURE 6. Average request delay under different algorithms.

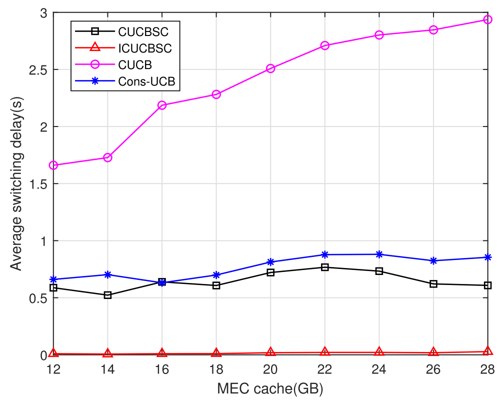

FIGURE 7. Average switching delay under different UCB algorithms.

the same in terms of average request delay, which confirms the performance of the ICUCBSC algorithm. The DRL algorithm obtains better request delay than the LRU and LRU strategies, however, comparing to the other algorithms, it has larger request delay.

Since that once new tiles are sent to the MEC server, caching replacement should be simultaneously considered in LRU, LFU and DRL, and MEC only stores the theoretical optimal solution in Optimal, Figure 7 shows only the average switching delay of each UCB algorithm. Obviously, the switching delay of CUCB is much larger than other UCB algorithms, which is consistent with the fact that it does not consider switching cost. Moreover, with the increase of the MEC cache, the content that needs to be switched increases, and the switching delay of CUCB is on the increase. And the curves of other UCB algorithms are more stable than that of CUCB due to the optimization of switching cost.

3) HIT RATIO

Firstly, due to the fact that MEC stores raw tiles and 3D tiles at the same time, but the requested tiles are 3D tiles, we define the hit as follows: for a requested tile, either it or its raw form which is stored in the MEC cache is a hit. So the hit ratio is the ratio of the number of hits to that of requests.

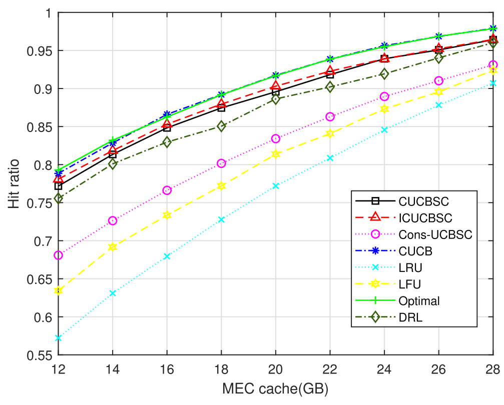

FIGURE 8. Hit ratio of all tiles for different algorithms.

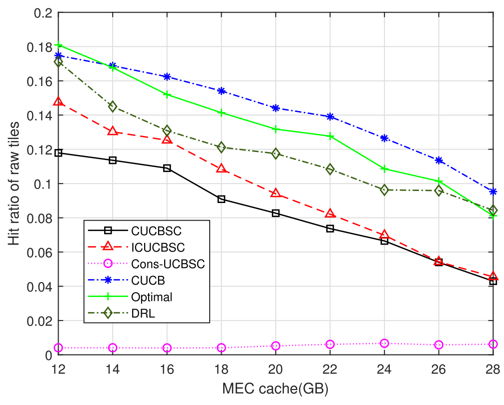

FIGURE 9. Hit ratio of raw tiles for different algorithms.

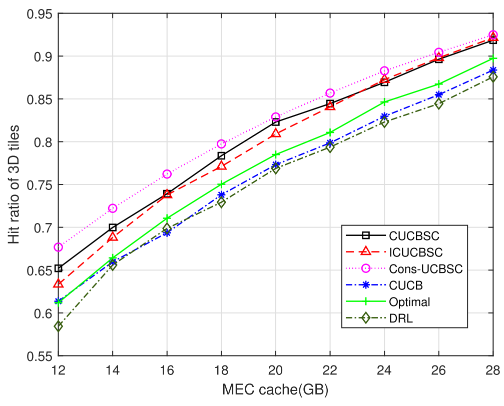

FIGURE 10. Hit ratio of 3D tiles for different algorithms.

Figure 8 shows the effect of MEC cache size under different caching algorithms in terms of hit ratio. With the increase of the MEC cache, the hit ratios of all algorithms are on the increase. It is interesting to see that CUCB and Optimal have almost the same performance and outperform other algorithms, followed by ICUCBSC. To further analyze the phenomenon, we plot the Figure 9 and Figure 10, which show the hit ratios of raw tiles and 3D tiles, respectively. Due to the fact that MEC stores only 3D tiles in LRU and LFU, these two algorithms are not included in Figure 9 and Figure 10.

From Figure 9 and Figure 10, we observe that with the increase of the MEC cache, the hit ratios of raw tiles decrease under all algorithms except Cons-UCBSC, while the hit ratios of 3D tiles are on the increase. Compared with other algorithms, CUCB stores more raw tiles which leads to higher total hit ratios. In contrast, Cons-UCB almost stores only 3D tiles, and thus its hit ratios of 3D tiles are the highest. Furthermore, Optimal algorithm does not perform best in Figure 9 and Figure 10, but the sums of its two hit ratios, i.e. its total hit ratios, are the highest. Whilst CUCBSC and ICUCBSC trade off between caching raw tiles and 3D tiles for achieving a lower transmission latency and switching latency, which is demonstrated in Figure 6 and Figure 7.

VII. CONCLUSION

In this paper, we study the optimal 360-degree video caching problem in an MEC-enabled VR system, where edge caching and transcoding are jointly considered. Besides, we note that the cost during the content replacement is an important issue which is often neglected in the exiting works. Based on the above considerations, we model the problem as a CMAB problem, and introduce the CUCBSC algorithm to solve the former problem. On the basis of CUCBSC, an improved CUCBSC (ICUCBSC) is proposed to speed up the learning process for practical usage in the video caching scenario. The simulation results show the superior performance of the proposed algorithm and prove its efficiency.

PROOF OF THEOREM 1

Firstly, we consider the expected number of bad periods. Define $l_t = \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2}$ [29]. To bound the counter $N_{i,t}$, which is updated in bad periods, we introduce the counter T_{vmnq}, which is used to count the times one base arm played in the CUCBSC algorithm. To be consistent with the form of $N_{i,t}$, we use $T_{i,t}$ as its new form where $i = (v-1)\cdot M\cdot N\cdot Q+(m-1)\cdot N\cdot Q+(n-1)\cdot Q+q$. From the two definitions, $T_{i,t} \geq N_{i,t}$. We define b as the total number of switching periods until time period t, and c_j as the time of the j-th switching period, where $j \leq b$ and $c_b \leq t$. We denote the event that **Algorithm 2** outputs a super arm which is not α times the optimal in a switching period t by E_t. So the bound of the number of bad periods until time t is given by:

$$\sum_{i=1}^{L} N_{i,t} \tag{21a}$$

$$= \sum_{k=L+1}^{t} \mathbb{I}\{S^t \in \mathcal{S}_B\} + L \tag{21b}$$

$$\leq \sum_{j=1}^{b} \mathbb{I}\{S^{c_j} \in \mathcal{S}_B\} \cdot F + L \tag{21c}$$

$$= \sum_{j=1}^{b}\sum_{i=1}^{L} \mathbb{I}\{S^{c_j} \in \mathcal{S}_B, N_{i,c_j} > N_{i,c_j-1}\} \cdot F + L \tag{21d}$$

$$\leq \sum_{j=1}^{b}\sum_{i=1}^{L}\mathbb{I}\{S^{c_j}\in\mathcal{S}_B, N_{i,c_j}>N_{i,c_j-1}, N_{i,c_j-1}\geq l_t\} \cdot F + L(1+l_t+F-1) \quad (21e)$$

$$= \sum_{j=1}^{b}\mathbb{I}\{S^{c_j}\in\mathcal{S}_B, N_{i,c_j-1}\geq l_t, \forall i\in S^{c_j}\} \cdot F + L(l_t+F) \quad (21f)$$

$$\leq \sum_{j=1}^{b}\mathbb{I}\{S^{c_j}\in\mathcal{S}_B, N_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\} \cdot F + L(l_t+F) \quad (21g)$$

$$\leq \sum_{j=1}^{b}(\mathbb{I}\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, N_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\} + \mathbb{I}\{E_{c_j}\})\cdot F + L(l_t+F) \quad (21h)$$

$$\leq \sum_{j=1}^{b}\mathbb{I}\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\}\cdot F + \sum_{j=1}^{b}\mathbb{I}\{E_{c_j}\}\cdot F + L(l_t+F) \quad (21i)$$

where (21b) is based on the assumption that super arms played in the initialization stage of of the CUCBSC algorithm are bad, and $\mathbb{I}(A)$ indicates whether one event A is true, that is, $\mathbb{I}(A) = 1$ if A is true and otherwise $\mathbb{I}(A) = 0$; in (21c) we use the fact that only in switching periods can arms be switched; (21d) follows from the fact that only one N_{i,c_j} is updated in each bad period; the first line of (21e) denotes the counters which are larger than l_t arm summed, and the second line bounds the counters which are smaller than l_t; (21f) is based on the updating rules of N_{i,c_j} that only the one with the smallest counter among the played arms can be updated in each bad period; (21g) follows from the definition of l_t, and that $t \geq c_j$; we divide the first line of (21g) based on whether there are errors in the solution of super arms, and then we obtain (21h); (21i) follows since $T_{i,t} \geq N_{i,t}$.

To obtain the bound on the expected number of bad periods until time period t, $\overline{N}_t$, it is necessary to compute the probability of $\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\}$ being true, which means that **Algorithm 2** has not failed to find the super arm, the j-th switching period is a bad period, and all base arms played in S^{c_j} have been chosen more than l_{c_j} times, where $c_j \leq t$. With the similar techniques based on the Chernoff-Hoeffding's inequality in [29], we can obtain that $P\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\} \leq 2\cdot L\cdot c_j^{-2}$. Then we can get

$$\mathbb{E}[\sum_{j=1}^{b}\mathbb{I}\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\}\cdot F] \quad (22a)$$

$$\leq F\sum_{j=1}^{b}2\cdot L\cdot c_j^{-2} \quad (22b)$$

$$\leq 2FL\sum_{j=1}^{\infty}c_j^{-2} \quad (22c)$$

$$\leq \frac{FL\pi^2}{3} \quad (22d)$$

where (22b) follows since we use $P\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\} \leq 2\cdot L\cdot c_j^{-2}$; in (22c), we broaden the upper bound to infinity; (22d) is obtained from the Riemann zeta function with parameter 2.

Then we can obtain the upper bound on the expected number of bad periods until time period t on the basis of (21), (22) and the fact that $\mathbb{E}[\mathbb{I}\{E_{c_j}\}] = 1-\beta$.

$$\overline{N}_t \leq \mathbb{E}[\sum_{j=1}^{b}\mathbb{I}\{\overline{E}_{c_j}, S^{c_j}\in\mathcal{S}_B, T_{i,c_j-1}\geq l_{c_j}, \forall i\in S^{c_j}\}F] + \mathbb{E}[\sum_{j=1}^{b}\mathbb{I}\{E_{c_j}\}F] + \mathbb{E}[L(l_t+F)] \quad (23a)$$

$$\leq (1-\beta)bF + L(\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2}) \quad (23b)$$

Note that, each time we choose a bad super arm in time period t, the generated regret is at most Δ_{max}. Then we plug (23) into (12), and obtain the sampling regret as follows.

$$R\_R(t) \leq t\alpha\beta r_{opt} - (t\alpha r_{opt} - \overline{N}_t\Delta_{max}) \quad (24a)$$

$$\leq (\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2})L\Delta_{max} \quad (24b)$$

PROOF OF THEOREM 2

Now, we consider the regret bound corresponding to the switching cost. Due to the fact that only in the switching periods can the arms be switched, the switching cost remains constant during the non-switching periods. Thus, the regret bound of the expected switching cost until period t can be expressed as:

$$R\_S(t) \quad (25a)$$

$$= \mathbb{E}[\sum_{k=1}^{t}D\_S^k] \quad (25b)$$

$$= \mathbb{E}[\sum_{j=1}^{b}\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbb{I}\{l_{vmn0}\in S^{c_j}, l_{vmn0}\notin S^{c_j-1}\}\frac{s_{vmn0}}{r} + \sum_{q=1}^{Q}(\mathbb{I}\{l_{vmnq}\in S^{c_j}, l_{vmnq}\notin S^{c_j-1}, l_{vmn0}\notin S^{c_j-1}\}\cdot\frac{s_{vmn0}}{r} + \mathbb{I}\{l_{vmnq}\in S^{c_j}, l_{vmnq}\notin S^{c_j-1}\}\cdot\frac{\omega|s_{vmnq}-s_{vmn0}|}{f})] + LC_u \quad (25c)$$

$$\leq \mathbb{E}[\sum_{j=1}^{b}\sum_{v=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=0}^{Q}\mathbb{I}\{l_{vmnq}\in S^{c_j}, l_{vmnq}\notin S^{c_j-1}\}\cdot(\frac{s_{vmn0}}{r} + \frac{\omega|s_{vmnq}-s_{vmn0}|}{f})] + LC_u \quad (25d)$$

where (25b) is the definition of the expected switching regret until period t; the first three items of (25c) are the expansion of (25b), which is equivalent to (4), and the last item follows since we assume that in the initialization stage of of CUCBSC algorithm, the switching cost is the maximum cost of switching between super arms; we relax the conditions in the second line of (25c) and obtain (25d).

For simplicity, we use i to denote the arm l_{vmnq}, and define H_i as $\frac{s_{vmn0}}{r}+\frac{\omega|s_{vmnq}-s_{vmn0}|}{f}$. Then, we can further bound $R\_S(t)$, which is given by:

$$
\begin{aligned}
&R\_S(t) && \text{(26a)}\\
&= \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}\mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}\}H_i] + LC_u && \text{(26b)}\\
&= \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}H_i(\mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\in \mathcal{S}_B\}\\
&\quad + \mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\notin \mathcal{S}_B\})] + LC_u && \text{(26c)}\\
&= \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}H_i(\mathbb{I}\{T_{i,c_j} > T_{i,c_{j-1}}, i\notin S^{c_{j-1}}, S^{c_j}\in \mathcal{S}_B\}\\
&\quad + \mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\notin \mathcal{S}_B\})] + LC_u && \text{(26d)}\\
&\le C_u\mathbb{E}[\sum_{j=1}^{b}\mathbb{I}\{\sum_{i=1}^{L}T_{i,c_j} > \sum_{i=1}^{L}T_{i,c_{j-1}}, i\notin S^{c_{j-1}}, S^{c_j}\in \mathcal{S}_B\}]\\
&\quad + \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}\mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\notin \mathcal{S}_B\}H_i] + LC_u && \text{(26e)}\\
&\le C_u\mathbb{E}[\sum_{j=1}^{b}\mathbb{I}\{\sum_{i=1}^{L}N_{i,c_j} > \sum_{i=1}^{L}N_{i,c_{j-1}}\}]\\
&\quad + \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}\mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\notin \mathcal{S}_B\}H_i] + LC_u && \text{(26f)}\\
&= C_u\underbrace{\sum_{j=1}^{b}\frac{\overline{N}_{c_{j+1}-1}-\overline{N}_{c_j-1}}{F}}_{Sw_B(t)}\\
&\quad + \mathbb{E}[\sum_{j=1}^{b}\sum_{i=1}^{L}\mathbb{I}\{i\in S^{c_j}, i\notin S^{c_{j-1}}, S^{c_j}\notin \mathcal{S}_B\}H_i] + LC_u && \text{(26g)}\\
&\le C_uSw_B(t) + Sw_B(t)C_u + (b-2Sw_B(t))C_l + LC_u && \text{(26h)}\\
&= 2Sw_B(t)(C_u-C_l) + LC_u + bC_l && \text{(26i)}
\end{aligned}
$$

where (26b) is equivalent to (25d); in (26c), we divide the first item of (26b) into two parts according to whether the super arm is bad; (26d) follows since that $T_{i,t}$ is updated when its corresponding arm is played in time period t; (26e) follows since we use C_u to denote the maximum cost of switching between super arms; (26f) follows based on the fact that in each bad period only one counter $N_{i,t}$ is updated; in (26g), we use the fact that $\frac{\sum N_{c_{j+1}-1}-\sum N_{c_j-1}}{F}$ is one if $\sum N_{c_{j+1}-1} > \sum N_{c_j-1}$, and zero otherwise, and denote $\sum_{j=1}^{b}\frac{\overline{N}_{c_{j+1}-1}-\overline{N}_{c_j-1}}{F}$ with $Sw_B(t)$, which represents a bound on the number of switches to bad super arms; in (26h), we assume that when a switching period is bad, its next switching period is good so as to minimize the number of consecutive plays of good super arms; and (26i) proves (18).

PROOF OF THEOREM 3

From the definition of $Sw_B(t)$:

$$Sw_B(t) = \frac{\overline{N}_{c_{b+1}-1}-L}{F} \tag{27}$$

We plug (23) into (27), and use the fact that $c_b \le t < c_{b+1}$ and $t \ge L+1$. The bound of $Sw_B(t)$ is given by:

$$
\begin{aligned}
Sw_B(t) &\le (1-\beta)b + \frac{L}{F}(\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2}\\
&\quad + \frac{6\ln(1+\frac{F-1}{L+1})}{(f^{-1}(\Delta_{min}))^2} - 1) && \text{(28)}
\end{aligned}
$$

Therefore, we can obtain the bound of switching regret, which is expressed as:

$$
\begin{aligned}
R\_S(t) &\le 2((1-\beta)b + \frac{L}{F}(\frac{F\pi^2}{3} + F + \frac{6\ln t}{(f^{-1}(\Delta_{min}))^2}\\
&\quad + \frac{6\ln(1+\frac{F-1}{L+1})}{(f^{-1}(\Delta_{min}))^2} - 1))(C_u - C_l)\\
&\quad + LC_u + bC_l && \text{(29)}
\end{aligned}
$$

Based on the assumption that there exists a unique optimal solution to the SPO problem, that is, $\alpha = \beta = 1$ and $C_l = 0$, we plug (24) and (29) into (14), which proves (19).

REFERENCES

[1] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," in *Proc. 7th Int. Conf. Multimedia Syst.*, May 2016, pp. 1–3.

[2] J. Liu, G. Simon, X. Corbillon, J. Chakareski, and Q. Yang, "Delivering viewport-adaptive 360-degree videos in cache-aided MEC networks," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Oct. 2020, pp. 1–6.

[3] J. Shi, L. Pu, and J. Xu, "Allies: Tile-based joint transcoding, delivery and caching of 360° videos in edge cloud networks," in *Proc. IEEE 13th Int. Conf. Cloud Comput. (CLOUD)*, Oct. 2020, pp. 337–344.

[4] J. Liu, Q. Yang, and G. Simon, "Congestion avoidance and load balancing in content placement and request redirection for mobile CDN," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 851–863, Apr. 2018.

[5] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360° videos in heterogeneous networks," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–6.

[6] Q. Lu, C. Li, J. Zou, K. Tang, Q. Wang, and H. Xiong, "Transcoding-enabled edge caching and delivery for tile-based adaptive 360-degree video streaming," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2019, pp. 1–4.

[7] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7573–7586, Nov. 2019.

[8] T. Dang, M. Peng, Y. Liu, and C. Liu, "Joint bandwidth, caching, and computing resource allocation for mobile VR delivery in F-RANs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[9] J. Chakareski, "Viewport-adaptive scalable multi-user virtual reality mobile-edge streaming," *IEEE Trans. Image Process.*, vol. 29, pp. 6330–6342, 2020.

[10] J. Dai, Z. Zhang, S. Mao, and D. Liu, "A view synthesis-based 360° VR caching system over MEC-enabled C-RAN," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3843–3855, Oct. 2020.

[11] M. Yeh, C.-H. Wang, D.-N. Yang, J.-T. Lee, and W. Liao, "Mobile proxy caching for multi-view 3D videos with adaptive view selection," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2909–2921, Aug. 2022.

[12] C. Zheng, S. Liu, Y. Huang, and L. Yang, "MEC-enabled wireless VR video service: A learning-based mixed strategy for energy-latency tradeoff," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[13] H. Xiao, C. Xu, Z. Feng, R. Ding, S. Yang, L. Zhong, J. Liang, and G.-M. Muntean, "A transcoding-enabled 360° VR video caching and delivery framework for edge-enhanced next-generation wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1615–1631, May 2022.

[14] W. Chen, Q. Song, P. Lin, L. Guo, and A. Jamalipour, "Proactive 3C resource allocation for wireless virtual reality using deep reinforcement learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[15] P. Maniotis and N. Thomos, "Viewport-aware deep reinforcement learning approach for 360° video caching," *IEEE Trans. Multimedia*, vol. 24, pp. 386–399, 2021.

[16] A. Zare, A. Aminlou, and M. M. Hannuksela, "Virtual reality content streaming: Viewport-dependent projection and tile-based techniques," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 1432–1436.

[17] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "VR is on the edge: How to deliver 360° videos in mobile networks," in *Proc. Workshop Virtual Reality Augmented Reality Netw.*, Aug. 2017, pp. 30–35.

[18] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 315–323.

[19] C. Guo, Y. Cui, and Z. Liu, "Optimal multicast of tiled 360 VR video," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 145–148, Feb. 2019.

[20] T. Xu, Y. Sun, S. Xia, H. Li, L. Luo, and Z. Chen, "Optimal bandwidth allocation with edge computing for wireless VR delivery," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2019, pp. 903–907.

[21] N. Kan, J. Zou, K. Tang, C. Li, N. Liu, and H. Xiong, "Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 4030–4034.

[22] P. Auer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, May 2002.

[23] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1897–1903.

[24] P. Blasco and D. Gunduz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 51–55.

[25] W. He, D. He, Y. Huang, Y. Zhang, Y. Xu, G. Yun-feng, and W. Zhang, "Bandit learning-based service placement and resource allocation for mobile edge computing," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[26] Y. Du, P. Gao, X. Wang, B. Dong, Z. Chen, and S. Li, "Monte-carlo tree search aided contextual online learning approach for wireless caching," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–7.

[27] Z. Yu, J. Liu, S. Liu, and Q. Yang, "Co-optimizing latency and energy with learning based 360° video edge caching policy," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022, pp. 2262–2267.

[28] Y. Hao, L. Hu, Y. Qian, and M. Chen, "Profit maximization for video caching and processing in edge cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1632–1641, Jul. 2019.

[29] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. 30th Int. Conf. Int. Conf. Mach. Learn.*, vol. 28, 2013, pp. I-151–I-159.

[30] H. Kellerer, U. Pferschy, and D. Pisinger, "The multiple-choice knapsack problem," in *Knapsack Problems*. Berlin, Germany: Springer, 2004.

[31] M. Bansal, V. Venkaiah, and C. Venkaiah, "Improved fully polynomial time approximation scheme for the 0-1 multiple-choice knapsack problem," Int. Inst. Inf. Technol., Hyderabad, India, Tech. Rep., 2004.

[32] A. Mahzari, A. T. Nasrabadi, A. Samiei, and R. Prakash, "FoV-aware edge caching for adaptive 360° video streaming," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 173–181.

[33] J.-T. Lee, D.-N. Yang, and W. Liao, "Efficient caching for multi-view 3D videos," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

ZHENDONG YU received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 2019, where he is currently pursuing the M.S. degree in information and communication. His research interests include mobile edge computing and wireless VR transmission.

JIAYI LIU (Member, IEEE) received the Bachelor of Science degree in electronic engineering from Xidian University, Xi'an, China, in 2007, the Master of Science degree in computer science from Télécom Bretagne, France, in 2009, and the Ph.D. degree from the University of Rennes 1, France, in 2013. Since 2014, she has been working as a Lecturer with the School of Telecommunication Engineering, Guangzhou Institute of Technology, Xidian University. Her research interests include 5/6G networks, content distribution, network resource scheduling, and AI enabled network management.

CHEN WANG received the Bachelor of Science degree in computer science from Tongji University, Shanghai, China, in 2007, the Master of Science degree in computer science from Télécom Bretagne, France, in 2009, and the Ph.D. degree from the National Institutes of Science and Technology, France, in 2013. He worked as a Research Scientist with Bosch China, from 2014 to 2017. Since 2017, he has been working as a Research Scientist with Huawei Technologies. His current research interests include machine learning algorithms and AI applications.

QINGHAI YANG (Member, IEEE) received the B.S. degree in communication engineering from the Shandong University of Technology, China, in 1998, the M.S. degree in information and communication systems from Xidian University, China, in 2001, and the Ph.D. degree in communication engineering from Inha University, South Korea, in 2007. From 2007 to 2008, he was a Research Fellow with UWB-ITRC, South Korea. Since 2008, he has been a Full Professor with Xidian University. His current research interests include autonomic communication, content delivery networks, and 6G. He has won the University-President Award for his Ph.D. degree from Inha University.