IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 30, 2020, accepted July 13, 2020, date of publication July 23, 2020, date of current version August 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011517

Bitrate Adaptation for Video Streaming Services in Edge Caching Systems

ANH-TIEN TRAN[1], NHU-NGOC DAO[2], (Member, IEEE), AND SUNGRAE CHO[1], (Member, IEEE)

[1]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea
[2]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea

Corresponding author: Sungrae Cho (srcho@cau.ac.kr)

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019R1A2C1090447).

ABSTRACT With the evolution of 5G networks, mobile edge computing (MEC) is being considered as a key enabler for realizing significant improvements in heterogeneous video streaming services. This is because MEC provides storage and computation resources for adaptive bitrate (ABR)-video streaming services to transcode the original video into lower bitrate variants at the edge caching server proactively, thereby facilitating the heterogeneous demands of users. In this paper, we propose a caching and processing framework that jointly considers the popularity and retention rate of video streams to maximize their video bitrate. The problem is formulated as an integer linear program (ILP), which that is challenging because of its NP-hardness. Our algorithm is called online iterative greedy-base adaptation (OIGA); it is built based on the greedy approach with strict constraints for storage size and computing capacity of the cache server. The simulation results show that our proposed solution adapts well to the change in video popularity and retention rate for a maximal video bitrate.

INDEX TERMS Video streaming services, cache placement, bitrate adaptation, edge caching systems.

I. INTRODUCTION

Over the past few years, the proliferation of over-the-top (OTT) video content providers (YouTube, Amazon Prime, Netflix, etc), coupled with ever-increasing multimedia processing capabilities and affordability of mobile devices has become the main driving force behind the increase in on-demand mobile video streaming services. The prevalence of mobile devices causes mobile internet traffic to increase considerably. Mobile video streaming is predicted to be accountable for 82% of the total mobile data traffic by 2022 [1]; and Ericsson estimates that there will be 7.3 million subscriptions connected to different networks [2]. While these demands create a considerable pressure on mobile network operators, edge caching has been regarded as a promising solution to store popular video content in close proximity to end users, which helps minimize data traffic through backhaul links and the time required to deliver content, thereby helping alleviate traffic during peak times. In wireless edge caching, the cache server, which is either the cellular base station or wireless access point, proactively cache trendy videos in expectation that requests for the same content from users can be accommodated easily without the need to duplicate transmission from the original video content sources.

In fact, streamers generate their video streaming content via designated applications supported by popular live streaming platforms. The live streaming platforms are solutions for video hosting that allow streamers to upload and broadcast video content to their audiences. Recently, several video sharing platforms have emerged; for example Twitch, YouNow, Facebook Live, and YouTube Live, which target different classes of users [3]–[5]. Twitch is a streaming platform that enables streamers to broadcast their screen while playing games, thus sharing their gaming experience with their followers and interacting with others in real time; this platform has become the fourth largest source of peak internet traffic in the US [6]. YouNow is primarily used by children and teenagers to broadcast self-portrayal videos. In case of Facebook Live, some streams reach tens or even hundreds of thousands of visitors. Facebook updated its ranking algorithm to show more live videos on people's news feeds. YouTube Live is a live streaming service, and offers an easy way to reach audience in real time. Whether you're streaming a video game, hosting a live Q&A, or teaching a class, YouTube live

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

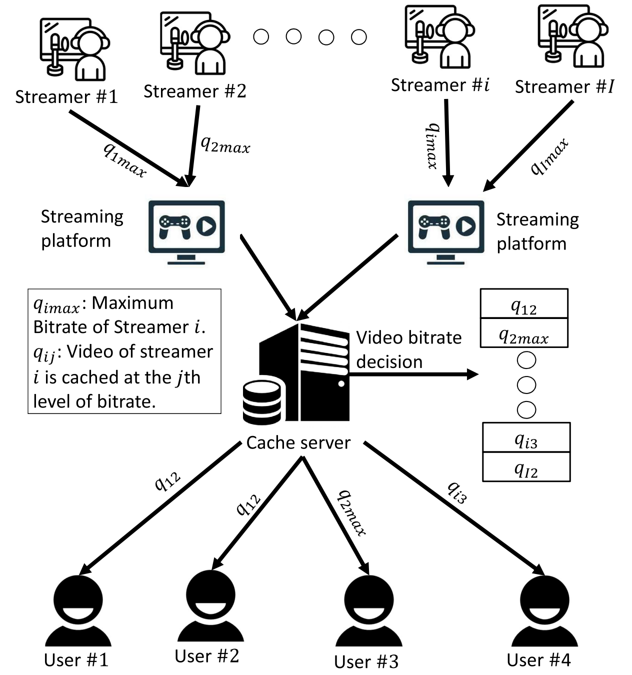

FIGURE 1. Typical network model of video streaming services in edge caching systems.

tools will help you manage your stream and interact with viewers in real time.

Along with the heterogeneous computing capacities of the streamers and the variations in the network condition, the maximum video bitrate that streamers can provide may oscillate over different times. The abnormal network conditions translate into inappropriate frame freeze, or sudden video quality changes, which can lead to deterioration in the quality of experience for the viewers. To resolve these problems, adaptive bitrate (ABR) streaming techniques have been widely used to assess the scenarios of network conditions and then adapt the quality of the transmitted video accordingly. The deterioration of quality may result from video compression or network conditions (e.g., loss of packets, insufficient bandwidth, delay, and jitter) [7]. Figure 1 illustrates a typical model of these video streaming services powered by edge caching systems. Dynamic Adaptive Streaming over HTTP (DASH), which is a recent development in HTTP streaming promotes adaptive bitrate streaming (ABR) that enables a video player to switch between available representations of bitrate/quality during a video viewing session based on the current network condition and their preferences. A common implementation is to encode the entire video in chunks, where each chunk comprises several seconds of the video using different bitrates.

However, caching multiple video variants of a same video incurs high overhead in terms of storage and, and thus, it is generally not optimal. The redundancy on storage usage also leads to an increase in monetary cost for purchasing additional hard disks. A preferable solution for this redundancy is to use video transcoding techniques so that only a version of the video is kept at the cache server. Among these techniques, the most favorable are compressed domain-based approaches such as bitrate reduction and spatial resolution reduction [8]. Upon request, they compress a higher video bitrate to a lower video bitrate variant [9]. Owing to its real-time computing capability, a cache server can perform the transcode operation for an original video to different variants to dynamically control the size of its video. The potential benefits of transcoding the video at the cache server is three-fold: (i) multimedia processing capabilities are more appropriately achieved at the cache server than at the battery-limited user devices; (ii) unique video variants free the space for other video contents from other streamers, and therefore, the cache hit ratio will hopefully increase; and (iii) more end users may be reached because they have more video bitrate options because of their preferences and network conditions. However, transcoding a large number of videos simultaneously can rapdily exhaust the available computing resource on cache servers [10]. Therefore, it is very essential to plan a resource allocation scheme for caching that efficiently utilizes both the given storage and processing resources.

Despite several studies on optimizing video streaming services using transcoding techniques, there remain several challenges in adopting the streaming services into video caching systems. First, the problem of determining the set of video bitrates for all streamers to be cached is nontrivial because of the variations in the optimization values of the variables, which are updated based on the time period. The global optimization approaches may have to consider a large number of variables, and it can lead to unreasonable running time to output global optimal solutions. Second, many research works focus only on video popularity, which is quantified by the fame of the content provider in the community instead of the attractiveness of the content. This bias can lead to impractical optimization solutions because the intention of users may unexpectedly change the subject to the content they are currently watching. Therefore, even if the streamer is popular, users will abandon the streaming session if they find that the content is not enjoyable. After the first 15 seconds of a video, most audiences are likely to finish their video viewing session [11]. These behaviors define the user retention rate. The retention rate of each video reflects the attractiveness of the video content according to video playback time, and the popularity of the streamers implies the fame of streamers in the community. If the users abandon the video session abruptly, video content that are downloaded but are never watched will be considered wasted resources in terms of energy, storage, and computing capability. These challenges motivate us to adopt users retention rate and develop a lightweight heuristic algorithm for determining the set of optimal video bitrate for all streamers.

This paper propose an effective approach called online iterative greedy-based adaptation (OIGA), which runs within each time period to optimize the cached video bitrate at the cache server. At each iteration, the OIGA algorithm jointly considers current video popularity and retention rate to greedily selects available streamers for video bitrate adjustment thus, the average video bitrate is maximally increased for the entire system. To adapt to environment changes, the OIGA algorithm considers dynamic transmission throughput, cache storage, and computing capabilities. The main contributions of this article are,

- Adapting the notions of popularity of streamers and retention rate of users based on the video content of streamers into the optimization problem to increase its viability in practical scenarios.
- Dealing with the unexpected dynamicity of practical transmission schemes by calculating the available video bitrate at each time period because of the observed transmission throughput between streamers and cache server such that the smoothness in video streaming services is reserved.
- Formulating the video caching optimization problem under the constraints of computing and storage capacity of the cache server. The target is to maximize the average video bitrate for all streamers. To this end, we propose the OIGA algorithm to solve the optimization problem; the algorithm decides the optimal level of the video bitrate for each streamer.
- Conducting performance analysis to prove that the OIGA algorithm achieves an approximate optimality compared to heavy global optimization approaches (e.g., branch-and-bound approach). The results suggest that our proposed algorithm can be applied to optimize on-the-fly even for a high density of streamers.

The remainder of this paper is organized as follows. The Section II surveys cutting-edge approaches proposed for streaming services. Section III-A demonstrates the system model. Section III-B formulates the optimization problem. Section IV proposes the OIGA algorithm. Section V shows the advantages of the OIGA algorithm by simulation results. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Emerging ABR streaming techniques over heterogeneous wireless networks are highly promising for the research community [12]–[22]. These techniques aim to improve the quality of experience for end users in terms of average video bitrate, mean-opinion-score (MOS), etc.

Yang *et al.* [12] derives an online adaptive rate control algorithm to adapt the bit stream to the variable bit rate downlink channels to maximize attainable visual quality. Zhang *et al.* [13] maximizes MOS by adapting cache management for ABR streaming over HTTP. In [14], Mehrabi *et al.* proposes a heuristic cache replacement strategy with a self-tuned bitrate selection algorithm. They aim to improve the cache hit ratio and average video bitrate by considering statistical information about retention pattern of clients. However, they use linear curves to represent client retention toward different videos, which may be impractical because the attractiveness of the content could be disregarded. Pedersen and Dey [15] improve video quality by enabling transcoding ability at the network edge of the radio access network. A rate adaptation algorithm is proposed to simultaneously and correspondingly change the transmission throughput and video encoding because of the observed video characteristics. Zahran *et al.* [16] design an ABR streaming algorithm to maintain a high video bitrate level while avoiding stalling events. They strike a balance between video quality and the ratio of stalling events to deal with the variability in network conditions. Dong *et al.* [17] proposes an ABR method with proxy caching for video streaming, targetted to improve the average quality level of users. However, their studies partly rely on the accuracy of the prediction of the channel conditions. This dependence could lead to the unexpected behavior of video streaming networks in reality under fluctuating network conditions.

Yin *et al.* [18] formulate ABR streaming as a stochastic optimal control problem, predict environment variables, and then solve the exact optimization problem accordingly. The proposed approach follows a table enumeration approach. Offline optimal solutions are stored for further use. Unfortunately, the efficacy relies solely on the correctness of the prediction step. Zhou *et al.* [19] proposes a Markov decision-based rate adaptation scheme and a sub-optimal greedy algorithm designed for ABR video streaming. The perceived video quality at the clients' side is better than the default video streaming protocol of Netflix in many test cases. The fundamental assumption that network bandwidth is Markovian confines the state space to consider only one past bandwidth measurement. However, it is not reliable to predict the distribution of the future bandwidth. Mao *et al.* [20] develop ABR algorithms using reinforcement learning. The analyses of the resulting efficiency of past decisions are embedded to train the neural network model. The average QoE is improved compared to conventional optimization techniques; however the running time of reinforcement learning is questionable in practice. In [21], Tran *et al.* propose an ABR streaming method to minimize the average download time for users in which media content are divided into equally sized chunks. In this paper, the cache server determines the number of chunks that should be cached, while the remaining video chunks are served from the remote server upon requests. Spiteri *et al.* [22] devise an online control algorithm, named BOLA, that utilizes the Lyapunov optimization framework to minimize the probability of video freezes; further, it maximizes the video quality. The BOLA algorithm sticks well to the abrupt bandwidth variations and requires no prediction on network bandwidth.

Although these aforementioned articles make significant contributions to improve the quality of experience for users in streaming services, a thorough investigation of video retention rate for improving the average video bitrate has not been considered.

III. PROBLEM STATEMENT

A. SYSTEM MODEL

In this paper, we consider the network model, as shown in Fig.1 which comprises a cache server, some streaming

TABLE 1. Key notation description.

Notation	Description
q_{ij}	Video content of streamer i is cached at video bitrate jth.
$q_{i\max}$	Highest available video bitrate of video content of streamer i.
$q_{\max}$	Highest video bitrate, defined by universal video codec.
$f(q_{i\max}, q_{ij})$	Number of cycles per seconds to transcode a one-second video from bitrate $q_{i\max}$ to q_{ij}.
c_{ij}	Caching decision variable to ecide to cache the video for streamer i at video bitrate j.
r_i	Retention rate of users for video content from streamer i at current time period.
$\mathcal{R}_i$	Observed transmission throughput between cache server and streamer i.
τ	Playback length of cached video in the cache server.
α	Zipf parameter.
$\bar{\mathcal{Q}}_i$	List of all available video bitrates for streamer i.
$\mathcal{I}$	List of all streamers.
I	Total number of streamers.
$\mathcal{M}$	Total size of cache storage of cache server.
$\mathcal{F}$	Total computing capacity of the CPU of cache server.

platforms, and many streamers. Streamers use live streaming platforms to deliver content to users via the cache server. Based on the current retention rate and the popularity of streamers, the cache server selects the optimal video bitrate for each streamer for each time period. The optimization problem is built for each time period. The list of all notations used in this paper are summarized in Table.1. Let $\mathcal{I} = \{1, 2, \ldots, i, \ldots, I\}$ denote the list of streamers, where $I = |\mathcal{I}|$ is the total number of streamers in the network. Suppose we have I streamers online in a current time period. These streamers transmit their video content to the cache server via the streaming platform. At each time period, the maximum video bitrate that the streamers can provide is denoted by $q_{i\max}$ and it is restricted by the current transmission throughput $\mathcal{R}_i$. The unexpected changes in the network conditions vary $\mathcal{R}_i$, and hence, $q_{i\max}$ over time period. At the cache server's side, the optimal set of video bitrate for streamers is decided such that the average video bitrate for the entire system is maximized. In Fig. 1, the video content of streamer #1 is degraded to level 2 of the video bitrate, while that for streamer #2 is cached as its maximum video bitrate $q_{2\max}$. For users who request video #1, they can only receive the video stream service with video bitrate 2 in the current time period. We denote the cached video bitrate j for streamer i as q_{ij}. In the scope of this paper, we do not discuss the transmission conditions between the users and the cache server.

For simplicity, the index of each streamer in $\mathcal{I}$ also represents their popularity in the descending order. The popularity of the ith streamer is denoted by p_i and it follows the Zipf distribution

$$p_i = \frac{\frac{1}{i^\alpha}}{\sum_{u=1}^{I} \frac{1}{u^\alpha}}, \quad 1 \le i \le I, \tag{1}$$

where α characterizes the distribution by controlling the entire relative popularity of files. A higher α implies higher content reuse, which means the first few popular streamers

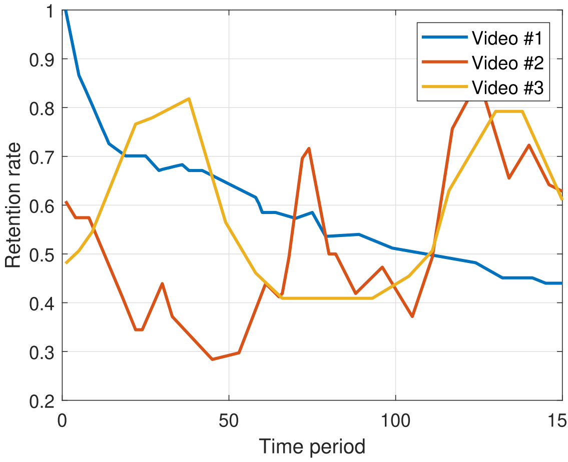

FIGURE 2. Examples of video retention rates. The retention rates of videos vary solely on the attractiveness of the video content. Different categories of videos may results in different patterns of retention rate graphs [23].

account for the majority of requests from users. Although we use the Zipf-distribution to stimulate the popularity of streamers, the efficiency of our algorithm is independent of the statistical distributions because it uses the observed values at the beginning of each time period as inputs. At each time the users open the streaming threads of the streamers, their retention may be different depending on the video content the streamer is providing. We define the probability if the new users continue watching the streaming service of streamer i right after opening it as the video retention rate and denote it by r_i. We assume that viewers' retention only depends on the appeal of the current video content that streamer is providing. Intuitively, p_i reflects the popularity of the streamer i, while r_i implies the attractiveness of its current video content. In this paper, we assume that p_i and r_i are independent. It is reasonable because even if a streamer is considerably famous, in time, their video content can be less appealing and users may be less likely to continue. As illustrated in Fig. 2, the retention rates of videos may change because of their categories and the current watching time. For video #1, the users start playing it with considerable interest. The introduction of the video may be very attractive; however, during playback, the retention rate decreases because the users lose their focus as the content may be boring. Video #3 represents the heterogeneity of users' retention. For the first quarter of the video length, the users increasingly enjoy the content; however, in the next quarter, they become less likely to continue watching.

Each streamer delivers their contents to the cache server via the streaming platform with a transmission throughput of $\mathcal{R}_i$ (kbps). To maintain a stable live-stream session, the video bitrate going out of the computer of each streamer cannot exceed its transmission throughput, i.e., $q_{i\max} \le \mathcal{R}_i$. For each streamer i, we define the set of available video bitrate as $\bar{\mathcal{Q}}_i = \{q_{ij} | 1 \le q_{ij} \le q_{i\max}\}$. We assume that video contents from all streamers follow an universal video codec, and the highest level of video bitrate in that video codec is $q_{\max}$. Obviously, $q_{i\max} \le q_{\max}, \forall i \in \mathcal{I}$.

We assume that the cache server always caches videos for the playback duration of τ seconds. However, owing to the limited capacity of cache storage, it may not be optimal to store all live-streaming videos of all streamers with its highest video bitrate $q_{i\max}$. Let $q_{ij} \leq q_{i\max}$ be the optimal cached video bitrate for streamer i with a level of j and $\boldsymbol{q} = \{q_{1j}, \ldots, q_{ij}, \ldots, q_{Ij}\}$ be the corresponding vector. The size of each cached video is proportional to its video bitrate. The total size for cached videos is restricted by the capacity of the cache storage $\mathcal{M}$

$$\sum_{i=1}^{I} q_{ij} \leq \frac{\mathcal{M}}{\tau}, \tag{2}$$

However, if we decide to cache the video bitrate $q_{ij} < q_{i\max}$, the cache server's CPU has to transcode the video bitrate from the original level ($q_{i\max}$) to the desired one (q_{ij}) for each second of the video length. It costs $f(q_{i\max}, q_{ij})$ GHz to do so; the total transcode cost is restricted by the computing capacity of the cache server, $\mathcal{F}$,

$$\sum_{i=1}^{I} f(q_{i\max}, q_{ij}) \leq \mathcal{F}, \tag{3}$$

where $f(q_{i\max}, q_{i\max}) = 0$ and $f(q_{i\max}, q_{ij})$ is an increasing function with respect to q_{ij}. To avoid trivial optimization solutions when the cache storage or computing capacity of the cache server are either too low to cache, even the lowest level of video bitrate or too high to store the highest level of video bitrate for all streamers, we assume $\mathcal{M}$ and $\mathcal{F}$ must be in the range

$$\sum_{i=1}^{I} q_1 < \frac{\mathcal{M}}{\tau} < \sum_{i=1}^{I} q_{i\max}, \tag{4}$$

$$\sum_{i=1}^{I} f(q_{i\max}, q_1) < \mathcal{F} < \sum_{i=1}^{I} f(q_{i\max}, q_{(i\max-1)}), \tag{5}$$

B. PROBLEM FORMULATION

We define the average video bitrate if the video content of streamer i is cached at video bitrate j as $\rho_{ij} = r_i p_i c_i q_{ij}$. Thus, the maximal average video bitrate of streamer i is $\rho_{i\max} = r_i p_i c_i q_{i\max}$. The objective of this paper is to select the optimal set of the level of video bitrate for all streamers with respect to the constraints of the cache storage size and the computing capacity of cache server, such that the average video bitrate is maximized.

$$(\mathcal{P}_1) : \rho = \max_{\boldsymbol{q}} \sum_{i=1}^{I} r_i p_i q_{ij}, \tag{6}$$

$$\text{s.t. (2), (3),}$$

$$q_{ij} \in \bar{\mathcal{Q}}_i, \quad \forall i \in \mathcal{I}, \tag{7}$$

To prove the NP-hardness of $(\mathcal{P}_1)$, we transform it into an equivalent multidimensional multiple-choice knapsack problem (MMKP). We use a set of binary selection variables $\boldsymbol{c} = (c_{ij})_{\left\{i\in\mathcal{I}, j\in\bar{\mathcal{Q}}_i\right\}}$. The variable c_{ij} is set to 1 if the cache server decides to cache the video for streamer i at the level of video bitrate j, and 0 otherwise. The objective function and the constraints of $(\mathcal{P}_1)$ are equivalently converted as

$$(\mathcal{P}_2) : \max_{\boldsymbol{c}} \sum_{i\in\mathcal{I}} \sum_{j\in\bar{\mathcal{Q}}_i} r_i p_i c_{ij} q_{ij}, \tag{8}$$

$$\text{s.t. } c_{ij} = \{0, 1\}, \quad \forall i, j, \tag{9}$$

$$\sum_{j\in\bar{\mathcal{Q}}_i} c_{ij} = 1, \quad \forall i, \tag{10}$$

$$\sum_{i\in\mathcal{I}} \sum_{j\in\bar{\mathcal{Q}}_i} c_{ij} q_{ij} \leq \frac{\mathcal{M}}{\tau}, \tag{11}$$

$$\sum_{i\in\mathcal{I}} \sum_{j\in\bar{\mathcal{Q}}_i} c_{ij} f(q_{i\max}, q_{ij}) \leq \mathcal{F}, \tag{12}$$

Constraint (9) reflects the binary nature of c_{ij}. Constraint (10) requires that only 1 video bitrate be selected for each streamer. Constraints (11) and (12) are the equivalent forms of (2) and (3). The structure of $(\mathcal{P}_2)$ strictly follows the typical structure of MMKP [24] and hence, it is NP-hard.

We propose the OIGA algorithm to solve $(\mathcal{P}_1)$ with a minimal optimality gap compared to branch-and-bound (BB) method. The details of this algorithm are explained in the next section.

IV. ONLINE ITERATIVE GREEDY-BASED ADAPTATION

This section describes our proposed OIGA algorithm for the bitrate adaptation of video streaming services in edge caching systems.

A. PROPOSED OIGA ALGORITHM

Let q_{ij} be the selected video bitrate for streamer i at the current iteration. In each iteration, all streamers are divided into two groups based on q_{ij},

- $\mathcal{I}_1 = \{i | q_{ij} < q_{i\max}\}$. This group contains streamers whose current selected video bitrate is not at the highest level.
- $\mathcal{I}_2 = \{i | q_{ij} = q_{i\max}\}$. This group contains the streamers who have the highest video bitrate. When the video content is cached at this level, the cache server does not need any additional computing cycle to transcode the video.
- Obviously, $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathcal{I}$ and, $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$.

In the initial phase, we cache all videos with the lowest video bitrate, $q_{ij} = q_1, \forall i \in \mathcal{I}$. For the streamers who can only support the lowest video bitrate, i.e., $q_{i\max} = 1$, we move them to $\mathcal{I}_2$ and do not reconsider them in the next iterations. The remaining streamers are sorted into $\mathcal{I}_1$. Let $\overline{\mathcal{M}}$ and $\overline{\mathcal{F}}$ be the remaining resource usage on the storage size and computing capacity of the cache server, respectively. Without the loss of generality, the streamers are considered *upgradable* if they are in $\mathcal{I}_1$ and the increase in video bitrate do not run out of the remaining storage size $\overline{\mathcal{M}}$, i.e., $\overline{\mathcal{M}} + q_{i(j+1)} - q_{ij} \geq 0$. At each iteration, we choose an upgradable streamer in $\mathcal{I}_1$, denoted as β, to upgrade such that the increase in the term of

Algorithm 1 Online Iterative Greedy-Based Adaptation

```
Set $q_{ij} = q_1, \forall i \in \mathcal{I}$ and $\overline{\mathcal{M}} = \frac{\mathcal{M}}{\tau} - \sum_{i=1}^{I} q_1$.
Pick streamers whose $q_{i\max} = 1$ and store in $\mathcal{I}_2$. Compute $\overline{\mathcal{F}} = \mathcal{F} - \sum_{i \in \mathcal{I}_1} f(q_{i\max}, q_1)$.
while $(\overline{\mathcal{M}} \geq 0)$ do
    Obtain streamer $\beta$ to upgrade based on (13).
    Update $\rho \leftarrow \rho + r_i p_i \left(q_{\beta(j+1)} - q_{\beta j}\right)$.
    Update $\overline{\mathcal{M}} \leftarrow \overline{\mathcal{M}} + q_{\beta(j+1)} - q_{\beta j}$.
    if $q_{\beta(j+1)} = q_{\beta\max}$ then
        Update $\overline{\mathcal{F}} \leftarrow \overline{\mathcal{F}} + f(q_{\beta\max}, q_{\beta j})$.
        Move $\beta$ from $\mathcal{I}_1$ to $\mathcal{I}_2$.
    else
        Update $\overline{\mathcal{F}} \leftarrow \overline{\mathcal{F}} - f(q_{\beta\max}, q_{\beta(j+1)}) + f(q_{\beta\max}, q_{\beta(j)})$.
    end if
    Update its video bitrate, $q_{\beta j} \leftarrow q_{\beta(j+1)}$.
end while
while $\overline{\mathcal{F}} < 0$ do
    Obtain streamer $\gamma$ to degrade based on (15).
    Update $\rho \leftarrow \rho + r_i p_i \left(q_{\gamma(j-1)} - q_{\gamma j}\right)$.
    Update $\overline{\mathcal{M}} \leftarrow \overline{\mathcal{M}} + q_{\gamma(j-1)} - q_{\gamma j}$.
    Update $\overline{\mathcal{F}} \leftarrow \overline{\mathcal{F}} - f(q_{\gamma\max}, q_{\gamma(j-1)}) + f(q_{\gamma\max}, q_{\gamma(j)})$.
    Update its video bitrate, $q_{\beta j} \leftarrow q_{\beta(j-1)}$.
end while
```

the average video bitrate is maximized,

$$\beta = \arg\max_{i \in \mathcal{I}_1} \left\{ r_i p_i (q_{i(j+1)} - q_{ij}) \right\}, \tag{13}$$

$$\text{s.t. } \overline{\mathcal{M}} + q_{i(j+1)} - q_{ij} \geq 0, \tag{14}$$

until the cache storage is no longer available. Constraint (14) ensures that the OIGA algorithm considers only upgradable streamers. If the upgraded video bitrate of streamer β matches its maximum video bitrate, i.e., $q_{\beta(j+1)} = q_{\beta\max}$, we move β from $\mathcal{I}_1$ to $\mathcal{I}_2$. Otherwise, we keep it in $\mathcal{I}_1$ for further consideration in the next iterations. After the storage size is utilized, if constraint (3) is also satisfied, we terminate the algorithm. Otherwise, we move to the next phase.

In this phase, we choose streamers from $\mathcal{I}_1$ to degrade. The degradation operation reduces the cost on transcoding tasks until $\overline{\mathcal{F}} \geq 0$. At each iteration, we choose streamer γ to degrade such that the decrease in terms of average video bitrate is minimized,

$$\gamma = \arg\min_{i \in \mathcal{I}_1, j>1} \left\{ r_i p_i \left(q_{ij} - q_{i(j-1)} \right) \right\}, \tag{15}$$

The algorithm terminates as soon as $\overline{\mathcal{F}} \geq 0$. The details of the OIGA algorithm are described in Algorithm.1.

B. COMPLEXITY ANALYSIS

At each iteration, the OIGA algorithm only upgrades/degrades a streamer by one level of the video bitrate. Although there are two while loops inside the OIGA algorithm, the second loop is optional because of the feasibility of the solution from the first loop. Hence, the time complexity of the OIGA algorithm is determined by the first while loop. The number of iterations of the first while loop increases with an initial value of $\overline{\mathcal{M}}$ and number of available video bitrate candidates from streamers. Intuitively, more video bitrate candidates or higher cache storage size requires more iterations in the first while loop. Therefore, the worst case occurs when all streamers support the highest video bitrate in the video codec, i.e., $q_{i\max} = q_{\max}, \forall i \in \mathcal{I}$, and the storage size is in the range $\sum_{i=1}^{I-1} q_{\max} + q_{\max-1} \leq \frac{\mathcal{M}}{\tau} < \sum_{i=1}^{I} q_{\max}$. The range of $\frac{\mathcal{M}}{\tau}$ implies that the storage size can support upto $I-1$ streamers with its highest available bitrate $q_{\max}$ and video bitrate $q_{\max-1}$ for the last streamer without violating constraint (4). In this scenario, for each streamer except the last streamer, it costs $(q_{\max} - 1)$ iterations to upgrade video bitrate from level 1 to $q_{\max}$. For the last streamer only, it costs $(q_{\max} - 2)$ iterations. Therefore, the OIGA algorithm requires $(I(q_{\max} - 1) - 1)$ iterations in total. This complexity reveals that the OIGA algorithm is *simple and easy to implement* in practice even with a large number of streamers.

TABLE 2. Simulation Parameters.

Parameter	Value
Number of streamers, I	[10, 500]
Video bitrates, q_{ij}	$\{3400, 2930, 1789, 1144, 374, 283\}$ kbps
Transcode task, f_{q_i}	$\{0, 1.83, 1.22, 0.82, 0.42, 0.37\}$ GHz
Computation capacity, $\mathcal{F}$	$\{32, 48, 64, 80, 96\}$ GHz
Length of cached chunk, τ	6 seconds
Zipf parameter, α	[0,1.2]
Transmission rate, $\mathcal{R}_i^s$	[1.5,5] Mbps
Cache storage size, $\mathcal{M}$	$\{50,75,100\}$ MB

V. PERFORMANCE EVALUATION

This section presents the simulation settings and results to verify the performance of the OIGA algorithm compared to the BB algorithm.

A. SIMULATION SETTINGS

We consider the network system illustrated in Fig. 1 and assume that there is an arbitrary number of streamers in the range of [10,500] associated with the cache server during each time period. The transmission rates of the streamers toward the streaming platforms vary in the range of [1.5,5] Mbps. We use the same video bitrate levels and transcode task specifications as in [25]. The capacity of the cache server in terms of storage and computing are in the range of $\{50, 75, 100\}$ MB and $\{32, 48, 64, 80, 96\}$ GHz, respectively. The value of α is in range of [0,1.2]. The values of the simulation parameters are provided as listed in Table. 2. We define the maximum number of cached streamers as the number of streamers that the cache server can cache with the lowest video bitrate. If the number of streamer exceeds this number, the cache server is unable to cache the lowest video bitrate of video content for all streamers, which is out of the scope of this paper. In the simulation, we use MATLAB R2018a as the optimization tool. We stimulate the

TABLE 3. Comparison of BB and OIGA.

Number of streamers	Average video bitrate (kbps)			Running time		
	BB	OIGA	Gap (%)	BB (s)	OIGA (ms)	Diff (s)
50	3008.15	2999.6	0.3	4.913	1.979	4.911
75	2394.5	2390.5	0.27	36.17	0.708	36.163
100	2027.9	2021.57	0.31	46.05	0.564	46.045

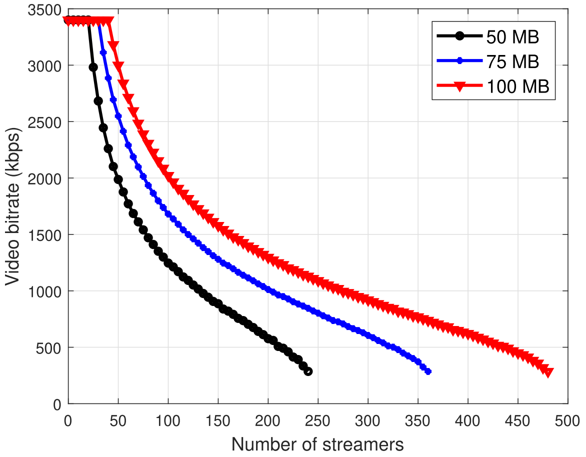

FIGURE 3. Storage size vs. Average video bitrate of streamers.

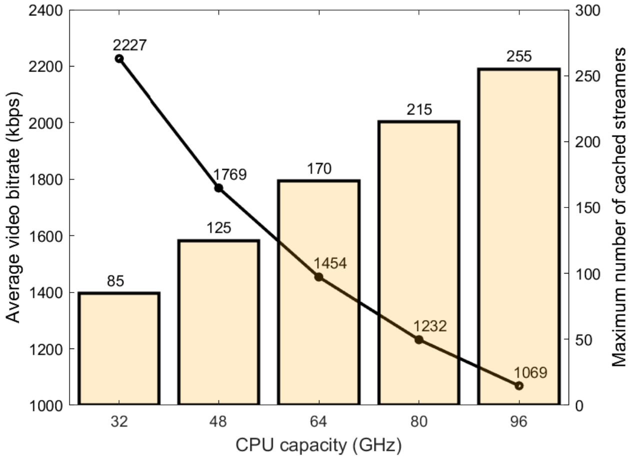

FIGURE 4. CPU capacity vs. Average video bitrate of streamers.

BB method using the *intlinprog*$(\cdot)$ default function [26] to solve MMKP ($\mathcal{P}_2$).

B. PERFORMANCE ANALYSIS

Table 3 indicates the superiority of our algorithm compared to the BB method. The optimality gap is negligible in all cases of the simulation process. The running time for the OIGA algorithm is infinitesimal compared to that for the BB method. For OIGA, we Fig. 3 represents the average video bitrate of all streamers with respect to the number of streamers with three levels of storage size. The average video bitrate steadily decreases with the number of streamers. A higher storage size supports streamers to be cached, and it offers a higher chance to cache higher video bitrates for each streamer, thereby improving the average video bitrate.

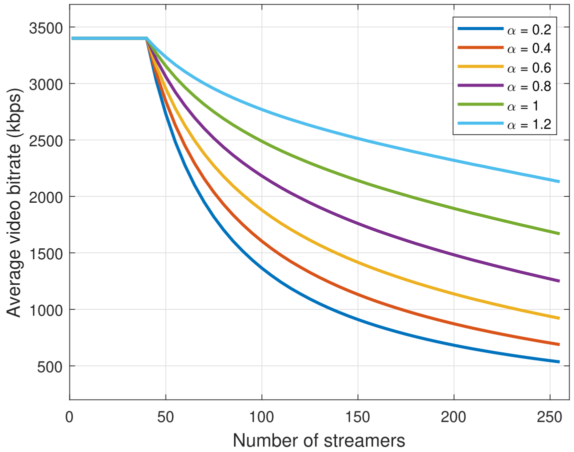

FIGURE 5. Average video bitrate of streamers vs. Number of Streamers under various Zipf parameters value.

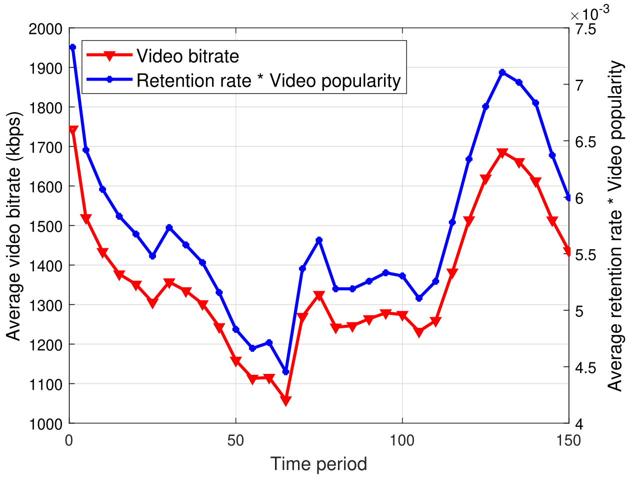

FIGURE 6. Average product of retention rate and video popularity vs. Average video bitrate of streamers.

In case we set $\mathcal{M}$ to 100 MB, the cache server can serve up to 480 streamers concurrently at each time period. The lowest average video bitrate is approximately 283 kbps, which is the lowest level of the video bitrate.

Fig. 4 represents the behavior of the system toward levels of CPU capacity while keeping the same storage size $\mathcal{M}$ of 100 MB. A higher computing capacity helps the cache server to increase the maximum number cached streamers. At a computing capacity of 96 GHz, the cache server can support up to 255 streamers simultaneously. However, if the computing capacity is restricted to 32 GHz, only 85 streamers can be cached.

Fig. 5 shows the effects of various values of the Zipf parameter α on an average video bitrate under various number of streamers ranging from 0 to 255. From the figure, we can see that the average video bitrate decreases with a number of streamers for all α values. Moreover, we can see that the average video bitrate is the lowest when the α parameter is the smallest and vice versa.

Fig. 6 reflects the behavior of the OIGA algorithm towards the change in the average retention rate of all streamers. It proves that the OIGA algorithm sticks well to abrupt changes on network conditions. The average video bitrate follows the same trend with the average retention rate. For instance, the average video bitrate decreases during the first 25 time periods and slightly increases in the next 5 time periods and again starts to decrease for the next 40 time periods, which is similar to the average retention rate. This is caused by the OIGA algorithm, which prioritizes video content interesting to a majority of users at each time period and decreases video bitrate for the video content unlikely to be continued by users, implying a low retention rate. This indicates that the proposed algorithm adaptively adjusts the video bitrate according to the retention rate.

VI. CONCLUSION

In this paper, we the adopted audience retention rate into the cache-support streaming network model, whereby the cache server determines the optimal bitrate for video contents for each streamer. The optimization problem aimed to maximize the average video bitrate with respect to the limit capacity of the storage size and computing capacity. The problem was formulated as an ILP. Because of the NP-hardness of the problem, we design a low-complexity algorithm called OIGA, which is easily implementable and provides solutions with near-optimal performance. The running time of the OIGA algorithm is negligible compared to that of the BB method. Extensive simulation results depicted that the OIGA algorithm achieves an infinitesimal optimality gap compared to BB, and it adapts well to abrupt changes in the retention of users towards streaming content over time periods.

REFERENCES

[1] *Cisco VNI Global IP Traffic Forecast, 2017–2022*, Cisco Vis. Netw. Index, San Jose, CA, USA, 2018.

[2] Ericsson. (2018). *Erricsson Mobility Report*. [Online]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf

[3] K. Pires and G. Simon, "YouTube live and twitch: A tour of user-generated live streaming systems," in *Proc. 6th ACM Multimedia Syst. Conf. (MMSys)*, 2015, pp. 225–230.

[4] D. Stohr, T. Li, S. Wilk, S. Santini, and W. Effelsberg, "An analysis of the YouNow live streaming platform," in *Proc. IEEE 40th Local Comput. Netw. Conf. Workshops (LCN Workshops)*, Oct. 2015, pp. 673–679.

[5] L. R. Jimenez, M. Solera, and M. Toril, "A network-layer QoE model for YouTube live in wireless networks," *IEEE Access*, vol. 7, pp. 70237–70252, 2019.

[6] J. Deng, F. Cuadrado, G. Tyson, and S. Uhlig, "Behind the game: Exploring the twitch streaming platform," in *Proc. Int. Workshop Netw. Syst. Support Games (NetGames)*, Dec. 2015, pp. 1–6.

[7] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2014.

[8] 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.

[9] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 18–29, Mar. 2003.

[10] 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.

[11] YouTube. (2016). *Audience Retention Report*. [Online]. Available: https://support.google.com/youtube/answer/1715160?hl=en-GB

[12] J. Yang, Y. Ran, S. Chen, W. Li, and L. Hanzo, "Online source rate control for adaptive video streaming over HSPA and LTE-style variable bit rate downlink channels," *IEEE Trans. Veh. Technol.*, vol. 65, no. 2, pp. 643–657, Feb. 2016.

[13] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, Oct. 2013.

[14] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "QoE-traffic optimization through collaborative edge caching in adaptive mobile video streaming," *IEEE Access*, vol. 6, pp. 52261–52276, 2018.

[15] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 996–1010, Apr. 2016.

[16] A. H. Zahran, J. Quinlan, D. Raca, C. J. Sreenan, E. Halepovic, R. K. Sinha, R. Jana, and V. Gopalakrishnan, "OSCAR: An optimized stall-cautious adaptive bitrate streaming algorithm for mobile networks," in *Proc. 8th Int. Workshop Mobile Video (MoVid)*, 2016, pp. 1–6.

[17] K. Dong, J. He, and W. Song, "QoE-aware adaptive bitrate video streaming over mobile networks with caching proxy," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 737–741.

[18] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, 2015, pp. 325–338.

[19] C. Zhou, C.-W. Lin, and Z. Guo, "mDash: A Markov decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 738–751, Apr. 2016.

[20] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 197–210.

[21] A.-T. Tran, T.-V. Nguyen, V.-D. Tuong, N.-N. Dao, and S. Cho, "On stalling minimization of adaptive bitrate video services in edge caching systems," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 115–116.

[22] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[23] E. Altman and T. Jiménez, "Measuring audience retention in YouTube," in *Proc. 12th EAI Int. Conf. Perform. Eval. Methodol. Tools (VALUETOOLS)*, 2019, pp. 79–85.

[24] S. Laabadi, M. Naimi, H. El Amri, and B. Achchab, "The 0/1 multidimensional knapsack problem and its variants: A survey of practical models and heuristic approaches," *Amer. J. Oper. Res.*, vol. 8, no. 5, p. 395, 2018.

[25] M. Song, Y. Lee, and J. Park, "Scheduling a video transcoding server to save energy," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 11, no. 2s, pp. 1–23, Feb. 2015.

[26] MATLAB. (2020). *Mixed-Integer Linear Programming (MILP)*. [Online]. Available: https://www.mathworks.com/help/optim/ug/intlinprog.html

ANH-TIEN TRAN received the B.S. degree in electronics and telecommunications from the Danang University of Science and Technology, Da Nang, Vietnam, in 2018. He is currently pursuing the Ph.D. degree in computer science with Chung-Ang University, South Korea. His research interests include wireless network communications, video streaming, fog computing, and machine learning.

NHU-NGOC DAO (Member, IEEE) received the B.S. degree in electronics and telecommunications from the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam, in 2009, and the M.S. and Ph.D. degrees in computer science from the School of Computer Science and Engineering, Chung-Ang University, Seoul, South Korea, in 2016 and 2019, respectively. From 2019 to 2020, he was a Postdoctoral Researcher with the Institute of Computer Science, University of Bern, Switzerland. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Sejong University, Seoul. His research interests include network softwarization, mobile cloudization, and the Internet of Things.

SUNGRAE CHO (Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Korea University, Seoul, South Korea, in 1992 and 1994, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002.

From 1994 to 1996, he was a Research Staff Member with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. From 2003 to 2006, he was an Assistant Professor with the Department of Computer Sciences, Georgia Southern University, Statesboro, GA, USA. He was a Senior Member of Technical Staff with the Samsung Advanced Institute of Technology (SAIT), Kiheung, South Korea, in 2003. From 2012 to 2013, he held a Visiting Professorship with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. He is currently a Professor with the School of Computer Science and Engineering, Chung-Ang University (CAU), Seoul. His current research interests include wireless networking, ubiquitous computing, and ICT convergence. He has served as a Program Committee Member, such as the IEEE ICC, GLOBECOM, VTC, MobiApps, SENSORNETS, and WINSYS. He has served numerous international conferences as an Organizing Committee Chair, such as the IEEE SECON, ICOIN, ICTC, ICUFN, TridentCom, and the IEEE MASS. From 2012 to 2017, he was an Area Editor of *Ad Hoc Networks Journal* (Elsevier). Since 2018, he has been a Subject Editor of *IET Electronics Letter*.