This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

Online Bitrate Selection for Viewport Adaptive 360-Degree Video Streaming

Ming Tang and Vincent W.S. Wong

Abstract—360-degree video streaming provides users with immersive experience by letting users determine their field-of-views (FoVs) in real time. To efficiently utilize the limited bandwidth resources, recent works have proposed a viewport adaptive 360-degree video streaming model by exploiting the bitrate adaptation in spatial and temporal domains. In this paper, under this video streaming model, we propose an online bitrate selection algorithm to enhance the user's quality of experience (QoE). This is achieved by characterizing the user's personalized FoV and real-time downloading capacity in an online fashion. We address the unknown user-specific FoV by introducing the reference FoV and design an online bitrate selection algorithm to learn the difference between the user's actual FoV and the reference FoV. We prove that as the number of video segments increases, the performance of the proposed online algorithm approaches the optimal performance asymptotically, with a bounded error. We perform trace-driven simulations with real-world datasets. Simulation results show that under the scenario where the available video bitrates are relatively high, our proposed algorithm can improve the user's viewing quality level between $4.2\% - 29.4\%$ and reduce the average intra-segment quality switch by at least 12.4% when compared with several existing methods.

Index Terms—Adaptive 360-degree video streaming, virtual reality, quality of experience (QoE), online convex optimization, online gradient descent.

1 Introduction

1.1 Background and Motivation

With the development of virtual reality (VR) technologies, 360-degree video streaming is becoming increasingly popular. With such 360-degree videos, users can determine their field of views (FoVs) in real time by controlling the direction of the streaming devices, with which users can have immersive video streaming experience. Currently, there are various 360-degree video content providers (e.g., YouTube) and VR devices supporting 360-degree videos (e.g., Sony PlayStation VR [1]). An example of a 360-degree video streaming in a wireless network is shown in Fig. 1. The users watching the same 360-degree video can have heterogeneous FoVs (represented by the solid rectangles).

The 360-degree real-time interaction, however, is at the expense of additional bandwidth consumption. This is because all the 360-degree scenes, including the scenes that are being viewed or not being viewed by the users, have to be downloaded in real time in response to the users' interactions. This additional bandwidth consumption imposes the requirement of efficient allocation of radio resources in wireless networks, so as to provide good quality of experience (QoE) video streaming services.

To improve the user's QoE, a promising approach is to use viewport adaptive 360-degree (VA360) video streaming [2]–[4], which exploits bitrate adaptation in both spatial and temporal domains. Specifically, in the encoding process, an entire video is divided into multiple segments, each of which corresponds to a video segment within a certain playback time period. Each segment is further spatially divided into multiple tiles. Each tile corresponds to the video of a viewing area during the segment playback time period. Each tile is encoded at multiple quality levels, each corresponding to a bitrate. During video streaming, video players can select the quality level of each tile, adapting to human behaviors and real-time network conditions. For example, in Fig. 2 (a), in the encoding process, each video is temporally divided into five segments. Each segment is spatially divided into $4 \times 6 = 24$ tiles, each of which is encoded at two quality levels {Low, High}. In Fig. 2 (b), when streaming the video, the video player can select the quality level of each tile to adapt to the variation of

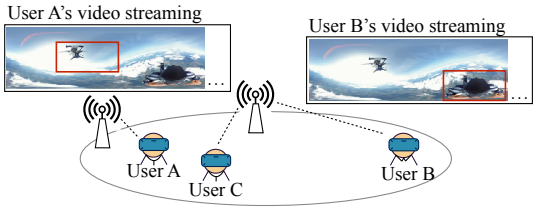


Fig. 1: 360-degree video streaming in a wireless network. Users A and B watch the same video, but they have heterogeneous FoVs.

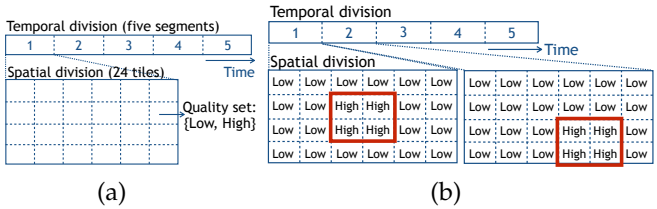


Fig. 2: Viewport adaptive 360-degree video streaming: (a) temporal and spatial divisions in video encoding; (b) bitrate adaptation in both spatial and temporal domains in video streaming.

Ming Tang and Vincent W.S. Wong are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada.
E-mail: {mingt, vincentw}@ece.ubc.ca

the user's FoV (represented by the solid rectangles) and network condition in order to improve the user's QoE. For example, if downloading all the tiles in high quality level will lead to rebuffering under the user's real-time network condition, then the video player can select high quality level for the tiles in the user's FoV and low quality level for the others to efficiently use the bandwidth.

In practice, however, the user's FoV for viewing a segment is unknown when the video player decides the bitrate or quality level of the segment, and the FoV may vary across segments. Meanwhile, the actual capacity for downloading a segment is unknown beforehand, and it may fluctuate significantly across time due to user mobility and handoff. To address the unknown and varying user's FoV and downloading capacity, existing works have proposed bitrate selection algorithms[1] for VA360 video streaming.

1.2 Related Work

The existing works on bitrate selection algorithms for VA360 video streaming can be classified into three types. The first type is head movement based algorithm, which selects the quality levels of tiles based on the user's short-term historical head movement. Qian *et al.* in [5] proposed an algorithm that predicts a user's FoV based on the user's head movement using linear and ridge regression. Zhang *et al.* in [6] proposed a deep reinforcement learning (DRL) algorithm to learn the optimal bitrate selection according to the predicted FoV and bandwidth. Pang *et al.* in [7] proposed a DRL algorithm according to the short-term historical head movement and bandwidth. He *et al.* in [8] proposed a bitrate selection algorithm for smartphones based on user's real-time head movement. Nguyen *et al.* in [9] proposed an algorithm that takes into account user's head movement and FoV estimation error. In [10], Sun *et al.* proposed a two-tier system taking into account both user's head movement and buffer occupancy. The second type is content based algorithm, which selects the quality levels based on saliency and motion maps. Borji *et al.* in [11] presented a survey of saliency detection methods. Shen *et al.* in [12] proposed a Lyapunov-based algorithm based on the detected saliency. Jiang *et al.* in [13] proposed a two-layer bitrate selection algorithm based on saliency and motion maps as well as buffer management. The third type is fixation clustering based algorithm, which optimizes the quality levels based on the historical fixation or FoVs of other users. In [14], Yuan *et al.* proposed to predict a user's FoV using a Gaussian model based on the user's real-time FoV. Xie *et al.* in [15] proposed an algorithm to maximize a user's video quality by identifying the user's class through comparing the historical FoVs of the user and other users in different classes. Xiao *et al.* in [16] proposed an algorithm based on the probability that each tile is being viewed by other users. Meanwhile, some works have considered combinations of the three types. Nguyen *et al.* in [17] and Fan *et al.* in [18] considered both head movement and saliency detection. Guan *et al.* in [19] considered both saliency detection and the historical FoV of other users.

1. Selecting the quality levels of the tiles is essentially selecting the bitrates of them, as the tiles encoded at different quality levels have different bitrates. In this work, we will use 'bitrate selection' and 'quality level selection' interchangeably if no confusion arises.

Despite the success of those algorithms, there is still potential for further improving the user's QoE by taking into account the personalized FoV. Specifically, there may be difference between a user's actual FoV and the predicted FoV. For those head movement based algorithms, such a difference may result from the user-specific head movement habits and the inaccuracy of the prediction algorithms. For those content based algorithms and fixation clustering based algorithms, such a difference may result from the user-specific viewing interests and preference.

In this work, we aim to propose an online bitrate selection algorithm that can optimize the user's QoE by learning the user's personalized FoV and real-time downloading capacity.

1.3 Solution Approach and Contributions

In this work, we propose an online bitrate selection algorithm, called OBS algorithm, for VA360 video streaming. This algorithm can learn the user's personalized FoV and real-time downloading capacity in an online fashion.

The idea of the OBS algorithm is inspired by the existing online gradient descent (OGD) algorithms [20]–[22], which can learn the users' features in an online fashion. The existing OGD algorithms, however, cannot be directly used in the VA360 video streaming model. The first reason is that those algorithms are only applicable to continuous decision variables, whereas the quality levels of the tiles are discrete in VA360 video streaming. The second reason is that in those algorithms, when the decision of an object (e.g., a segment in our work) is to be made, the realization of the parameters related to the previous object needs to be known. This requirement, however, may not always hold in VA360 video streaming. In comparison, our proposed algorithm can handle the discrete quality levels and the scenario where the video player may not have observed the realization of the parameters of the previous segment when it needs to make the bitrate decision of a segment.

The main contributions of this work are as follows.

- *Viewport-Adaptive 360-Degree Video Streaming:* We formulate a bitrate selection problem to maximize the user's QoE. It can characterize the user's personalized FoV and real-time downloading capacity.
- *Online Bitrate Selection Algorithm:* We propose an OBS algorithm for online bitrate selection. We prove that as the number of segments increases, the performance gap between the proposed algorithm and the offline optimal solution (where the user's FoV and downloading capacity are known beforehand) is bounded on the long-term average.
- *Performance Evaluation:* We perform trace-driven simulations with datasets from [23]–[25]. Simulation results show that our proposed algorithm can enhance the user's viewing quality level as well as reduce the inter-segment and intra-segment quality switch when compared with several existing methods.

The rest of this paper is organized as follows. We present the system model in Section 2. In Section 3, we propose the online bitrate selection algorithm. In Section 4, we show the performance evaluation. We conclude in Section 5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

2 System Model

In this section, we first introduce the model setting, and then formulate the user's QoE maximization problem.

2.1 Model Setting

We focus on the bitrate selection of a user's 360-degree video streaming. The video and user models are as follows.

2.1.1 Video Model

The video is temporally partitioned into segments (i.e., small video pieces), each corresponding to a playback time of β seconds. Let $\mathcal{I} = \{1, 2, \ldots, I\}$ denote the set of segments. Each segment is further spatially divided into K tiles with M rows and N columns, i.e., $K = MN$. Let $\mathcal{K} = \{1, 2, \ldots, K\}$ denote the set of tiles of each segment.

For each segment, we regard the predicted FoV of the user (based on the existing head movement, content, or fixation clustering based prediction algorithms) as a *reference FoV*. Hence, the user-specific FoV feature can be characterized based on the relative position and relative area of the user's actual FoV to the reference FoV. Fig. 3 shows an example of a 360-degree video streaming of car riding experience. During the playback time of a segment, the reference FoV may focus on looking ahead in the car (e.g., the four shaded tiles in the center), while the user's actual FoV may look towards the right-hand side of car (e.g., the solid rectangle). Moreover, the user may change his or her focus during the playback time of the segment, so the area covered by the user's actual FoV during the playback time may be different from that of the reference FoV.

The reference FoV can be different in different segments. Hence, in order to characterize the difference between the user's actual FoV and the reference FoV, we define the indices of the tiles based on their relative positions to the reference FoV.[2] The indices of the tiles are defined as follows. Suppose the top-left corner of the reference FoV is at the tile on row m_0 and column n_0. The tile on row m and column n is indexed with $N\text{mod}(m-m_0, M)+(\text{mod}(n-n_0, N)+1)$, where $\text{mod}(x, y)$ is equal to x modulo y, and M and N is the total number of rows and columns of the tiles, respectively. For example, in Fig. 4, the reference FoV is represented by the shaded area, which can be different for different segments. In segment 1, the top-left corner of the reference FoV is located at the tile on row $m_0 = 2$ and column $n_0 = 3$, so the tile on row $m = 2$ and column $n = 6$ is indexed by $6\text{mod}(2-2, 4) + (\text{mod}(6-3, 6)+1) = 4$. Note that the 360-degree video has no boundary, e.g., in segment 1, tile 5 is adjacent to the right-hand side of tile 4.

In the following, we use tile (i, k) to denote tile k of segment i. Each tile (i, k) is encoded at Q quality levels, i.e., $\mathcal{Q} = \{1, 2, \ldots, Q\}$, with quality level 1 corresponds to the lowest quality. In practice, different quality levels correspond to different constant rate factors [26] when a tile is being encoded [5, Section 9.1]. Suppose tile (i, k) is encoded at quality level $q \in \mathcal{Q}$. Let $r_{i,k}(q)$ (in Mbps) denote the bitrate of the tile, which is increasing in q. That is, a higher quality level q will lead to a higher bitrate $r_{i,k}(q)$. Note that under the same quality level q, different tiles may have different bitrates $r_{i,k}(q)$ due to the different features (e.g., motion) of their contents. The value of the bitrate represents the number of bits required to encode a tile corresponding to a playback time period of one second. Hence, if tile (i, k) corresponds to a playback time of β seconds and is encoded at quality $q \in \mathcal{Q}$, then the size of the tile is $r_{i,k}(q)\beta$ Mbits.

2. This is related to only tile indexing and does not affect the encoding or video streaming process. In practical systems, the indices of the tiles used in the video server may be different from the indices defined in this paper. In this case, when the video player requests a tile, it can first compute the index of the tile used in the server and then send the corresponding request.

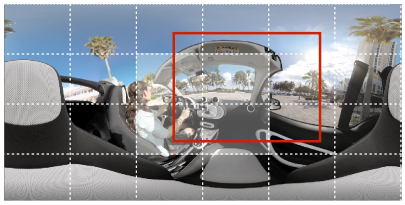


Fig. 3: An example with a 360-degree car riding video.

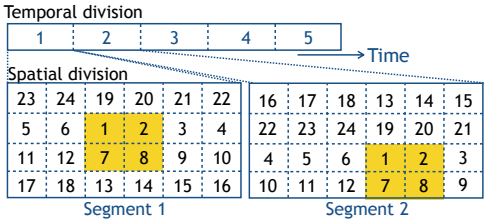


Fig. 4: An illustration of the reference tiles and tile indices.

2.1.2 User Model

During video streaming, the user's downloading capacity and his or her FoV are time-varying. The user's downloading capacity varies across time. We consider a continuous time interval $\mathcal{T} = [0, T)$. Let $d(t)$ denote the user's downloading capacity at time $t \in \mathcal{T}$, i.e., the user's maximum achievable downloading rate at time t. We assume that $d(t)$ is upper- and lower-bounded, i.e., $d(t) \in [d^{\min}, d^{\max}]$ for all $t \in \mathcal{T}$. On the other hand, the user's FoV varies across playback time (instead of the real time),[3] because the user changes the FoV to watch his or her interested contents in the video, which is playback time-associated. Let $\omega_{i,k}$ denote the fraction of tile (i, k) that is overlapped with the user's FoV, which is averaged across the playback time of segment i. Let $\mathbb{1}(\omega_{i,k})$ denote whether tile (i, k) is being viewed by the user or not, where $\mathbb{1}(\omega_{i,k}) = 1$ if $\omega_{i,k} > 0$ and is equal to zero otherwise. Due to the tile indexing considered in this work, the values of $\mathbb{1}(\omega_{i,k})$ for $k \in \mathcal{K}$ can imply how the user's actual FoV of segment i is different from the predicted one. For example, in Fig. 4, if $\mathbb{1}(\omega_{i,k}) = 1$ for tiles $k = \{1, 2, 3, 7, 8, 9\}$, then the actual user's FoV of segment i covers not only the reference FoV (i.e., the predicted FoV of the user) but also the right-hand side of it.

2.2 Problem Formulation

In the following, we first introduce the decision variables. We then describe the constraints and the QoE. Finally, we

3. For example, a user starts playing the video at time $t = 0$ sec, and the video rebuffers for one second during time interval $[1, 2]$ (sec). Then, at real time $t = 3$ sec, the playback time is at $3 - (2 - 1) = 2$ sec.

formulate the QoE maximization problem.

2.2.1 Decision Variables

The video player makes decisions on the quality levels of the tiles, which affect the corresponding bitrates of the tiles. Let $q_{i,k} \in \mathcal{Q}$ denote the quality level decision of tile (i,k). Note that as the indices of the tiles are defined based on their relative positions to the reference FoV, choosing the quality level of a tile with a particular index is essentially choosing the quality level of a tile covering a particular area with reference to the reference FoV. For example, in Fig. 4, if the video player chooses to download tiles $1-3$ and $7-9$ in high quality for both segments 1 and 2, then the tiles covering the reference FoV and the right-hand side of it are in high quality for both segments.

We assume that there is no segment replacement, and each tile is downloaded once. Let $\tau_{i,k} \in \mathcal{T}$ and $\hat{\tau}_{i,k} \in \mathcal{T}$ denote the time that the downloading of tile (i,k) is started and finished, respectively. The video player downloads the video tiles in a particular sequence: tile (i,k) is downloaded earlier than tile (i',k') if and only if either (a) $i < i'$ or (b) $i = i'$ and $k < k'$. Without loss of generality, we set $\tau_{1,1} = 0$ as the time when the downloading of the first tile of the first segment begins. When one tile has been downloaded, the downloading of the next tile will start immediately, i.e.,

$$\tau_{i,k} = \begin{cases} \hat{\tau}_{i,k-1}, & k > 1, \\ \hat{\tau}_{i-1,K}, & k = 1, i \neq 1, \\ 0, & k = 1, i = 1. \end{cases} \tag{1}$$

Let b_i denote the buffer occupancy when all the tiles of segment i have been downloaded. We define the buffer occupancy with respect to the segment index for the presentation simplicity of buffer update (i.e., the buffer is being updated when a segment has been received). The buffer occupancy is in the unit of playback time, which is commonly used in the existing works on dynamic adaptive streaming over HTTP (DASH), e.g., [27]. Receiving one segment will lead to a buffer occupancy increased by β seconds.

Without loss of generality, we set $b_0 = b^{\text{INI}}$ as the initial buffer occupancy. We define the following decision vectors: $\boldsymbol{q} = (q_{i,k}, i \in \mathcal{I}, k \in \mathcal{K})$, $\boldsymbol{q}_i = (q_{i,k}, k \in \mathcal{K})$, $\boldsymbol{\tau} = (\tau_{i,k}, i \in \mathcal{I}, k \in \mathcal{K})$, $\hat{\boldsymbol{\tau}} = (\hat{\tau}_{i,k}, i \in \mathcal{I}, k \in \mathcal{K})$, and $\boldsymbol{b} = (b_i, i \in \mathcal{I})$.

2.2.2 Downloading Capacity and Buffer Update Constraints

Within the downloading period of tile (i,k), the capacity constraint ensures that the total size of the downloaded tile should be no larger than the downloading capacity within the downloading period, i.e.,

$$r_{i,k}(q_{i,k})\beta \leq \int_{\tau_{i,k}}^{\hat{\tau}_{i,k}} d(t)dt, \ i \in \mathcal{I}, k \in \mathcal{K}. \tag{2}$$

When all the tiles of segment i have been received, the buffer is updated as follows:

$$b_i = [b_{i-1} - (\hat{\tau}_{i,K} - \tau_{i,1})]^+ + \beta, \ i \in \mathcal{I}, \tag{3}$$

where the operator $[x]^+ = \max\{x, 0\}$. In (3), if the buffer occupancy b_{i-1} is no smaller than the downloading period, then buffer occupancy b_i is the sum of the buffer occupancy immediately before receiving segment i and the received segment length β. If the buffer becomes empty before receiving segment i, then buffer occupancy b_i is equal to β.

2.2.3 User's QoE

Recall that the indicator function $\mathbb{1}(\omega_{i,k})$ denotes whether tile (i,k) is being viewed by the user or not. For the definition of the user's QoE, we define a function $\mu_i(\boldsymbol{q}_i)$ for any segment decision vector $\boldsymbol{q}_i$, indicating the average viewing quality when the user views segment i under the quality level decision, i.e.,

$$\mu_i(\boldsymbol{q}_i) = \frac{\sum_{k \in \mathcal{K}} \mathbb{1}(\omega_{i,k})\, q_{i,k}}{\sum_{k \in \mathcal{K}} \mathbb{1}(\omega_{i,k})}, \ i \in \mathcal{I}. \tag{4}$$

Note that this is the average quality level that the user actually views, considering the user's actual FoV.

The user's QoE consists of three terms, which are the user's gain due to its viewing quality $G(\boldsymbol{q})$, the rebuffering loss $L^{\text{RB}}(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b})$, and the quality switch loss $L^{\text{QS}}(\boldsymbol{q})$:[4]

$$U(\boldsymbol{q}, \boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b}) = G(\boldsymbol{q}) - L^{\text{RB}}(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b}) - L^{\text{QS}}(\boldsymbol{q}). \tag{5}$$

The user's gain of each segment is a function of the user's viewing quality of the segment, i.e., a higher viewing quality leads to a higher gain. The user's gain $G(\boldsymbol{q})$ is the summation of the user's gains of all segments:

$$G(\boldsymbol{q}) = \sum_{i \in \mathcal{I}} g_i(\mu_i(\boldsymbol{q}_i)), \tag{6}$$

where $g_i(\cdot)$ for segment i is a non-decreasing concave function. The concavity captures the fact that the user's marginal gain is non-increasing in the viewing quality.

Rebuffering happens when the video player has not receive a segment at the time that the segment is going to be played. The rebuffering loss is the user's loss resulting from the video freeze. This loss is proportional to the rebuffering time. The rebuffering loss is defined as follows:

$$L^{\text{RB}}(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b}) = l^{\text{RB}} \sum_{i \in \mathcal{I}} [\hat{\tau}_{i,K} - \tau_{i,1} - b_{i-1}]^+, \tag{7}$$

where l^{RB} is the unit loss if the user experiences a one-second rebuffering. In (7), if the duration between when the downloading of segment i starts and when the tiles of the segment have been downloaded is larger than the buffer occupancy b_{i-1}, then rebuffering will happen. This will lead to a rebuffering time of $[\hat{\tau}_{i,K} - \tau_{i,1} - b_{i-1}]^+$ seconds.

The quality switch (i.e., quality improvement and degradation) loss consists of two parts: an inter-segment quality switch loss $L^{\text{QS-E}}(\boldsymbol{q})$ and an intra-segment quality switch loss $L^{\text{QS-A}}(\boldsymbol{q})$. That is,

$$L^{\text{QS}}(\boldsymbol{q}) = L^{\text{QS-E}}(\boldsymbol{q}) + L^{\text{QS-A}}(\boldsymbol{q}). \tag{8}$$

The inter-segment quality switch loss is the loss resulting from the quality switch among segments, i.e.,

$$L^{\text{QS-E}}(\boldsymbol{q}) = l^{\text{QS-E}} \sum_{i \in \mathcal{I}/\{1\}} |\mu_{i-1}(\boldsymbol{q}_{i-1}) - \mu_i(\boldsymbol{q}_i)|, \tag{9}$$

where $l^{\text{QS-E}}$ is the loss if the viewing quality is switched by one unit. In (9), if $\mu_{i-1}(\boldsymbol{q}_{i-1})$ and $\mu_i(\boldsymbol{q}_i)$ are different, then

4. This work can be extended to the scenario with other formulations of user's QoE $U(\boldsymbol{q}, \boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b})$, as long as $U(\boldsymbol{q}, \boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b})$ is concave in $\boldsymbol{q}$ given any $\boldsymbol{\tau}$, $\hat{\boldsymbol{\tau}}$, $\boldsymbol{b}$. To ensure that the algorithm proposed in this work is applicable under the extended scenario, the per-segment QoE functions (i.e., $\widetilde{U}_i(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1}, b_{i-1})$ for $i \in \mathcal{N}$) in Section 3.1.1 have to be modified according to the specific formulation of $U(\boldsymbol{q}, \boldsymbol{\tau}, \hat{\boldsymbol{\tau}}, \boldsymbol{b})$.

the inter-segment quality switch occurs, leading to a loss which is proportional to the size of the quality switch. The intra-segment quality switch loss is the loss resulting from the variance of the quality levels of the tiles viewed by the user. That is,

$$L^{\text{QS-A}}(\boldsymbol{q}) = l^{\text{QS-A}} \sum_{i\in\mathcal{I}} \frac{\sum_{k\in\mathcal{K}} \mathbb{1}(\omega_{i,k}) \left(\mu_i(\boldsymbol{q}_i) - q_{i,k}\right)^2}{\sum_{k\in\mathcal{K}} \mathbb{1}(\omega_{i,k})}, \quad (10)$$

where $l^{\text{QS-A}}$ is the loss per unit variance. In (10), if the variance of the quality levels of the tiles that are being viewed by the user is nonzero, then the intra-segment quality switch happens, inducing a loss proportional to the variance.

2.2.4 Problem Formulation

We aim to determine the decision vectors $\boldsymbol{q}$, $\boldsymbol{\tau}$, $\hat{\boldsymbol{\tau}}$, $\boldsymbol{b}$ to maximize the user's QoE subject to the capacity and buffer update constraints. The problem is formulated as follows:

$$\begin{aligned}
\underset{\boldsymbol{q},\boldsymbol{\tau},\hat{\boldsymbol{\tau}},\boldsymbol{b}}{\text{maximize}} \quad & U(\boldsymbol{q},\hat{\boldsymbol{\tau}},\boldsymbol{\tau},\boldsymbol{b}) && \text{(11a)}\\
\text{subject to} \quad & q_{i,k} \in \mathcal{Q},\ i\in\mathcal{I}, k\in\mathcal{K}, && \text{(11b)}\\
& 0 \le \tau_{i,k} < T,\ i\in\mathcal{I}, k\in\mathcal{K}, && \text{(11c)}\\
& 0 \le \hat{\tau}_{i,k} < T,\ i\in\mathcal{I}, k\in\mathcal{K}, && \text{(11d)}\\
& b_i \ge 0,\ i\in\mathcal{I}, && \text{(11e)}\\
& \text{constraints (1), (2), (3).}
\end{aligned}$$

Problem (11) is a mixed-integer nonconvex programming problem with nonlinear constraints (2) and (3). This problem is challenging to solve, even in an offline case when all the parameters (i.e., the user's FoV and downloading capacity) are known beforehand. In this work, we focus on the online algorithm design. This addresses the realistic scenario where the user's FoV of a segment and the capacity for downloading the segment are unknown when the video player makes the quality level decision of the segment.

3 Online Algorithm Design

In this section, we focus on the online algorithm design. For the design of the online algorithm, we consider a set of per-segment problems, each corresponding to one of the segments of the video. The main idea of the algorithm is to make the quality level decision of a segment based on the quality level decisions of a set of previous segments as well as the per-segment problems of the previous segments, taking into account the corresponding realization of the real-time downloading capacity and user's actual FoV. The performance of the algorithm will be evaluated by dynamic regret [20], [21], reflecting the regret of the algorithm in the objective value. We show that as the number of segments increases, the dynamic regret of the proposed algorithm is bounded on the long-term average.

We first introduce the set of per-segment problems and the performance metric. Then, we introduce the online algorithm. After that, we show the performance guarantee of the proposed algorithm under particular conditions. Finally, we modify the online algorithm to address the scenario in practice systems.

3.1 Per-Segment Problem

For the design of the online algorithm, we consider a set of per-segment optimization problems. The per-segment problem corresponding to segment $i \in \mathcal{I}$ aims to optimize the quality level decision of segment i at the time that the decisions of all the previous segments have been made.

3.1.1 Per-Segment Objective Function

For segment $i \in \mathcal{I}$, we define a function $\widetilde{U}_i(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1}, b_{i-1})$ to indicate the user's QoE of segment i under decision vector $\boldsymbol{q}_i$, given the quality decision of segment $i-1$ (i.e., $\boldsymbol{q}_{i-1}$) and the buffer occupancy before downloading segment i (i.e., b_{i-1}). The function $\widetilde{U}_i(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1}, b_{i-1})$ is given as follows:

$$\begin{aligned}
&\widetilde{U}_i(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1}, b_{i-1})\\
&\quad = g_i(\mu_i(\boldsymbol{q}_i)) - l^{\text{RB}} \left(\frac{1}{\bar{d}_i} \sum_{k\in\mathcal{K}} r_{i,k}(q_{i,k})\beta - b_{i-1} \right)\\
&\qquad - l^{\text{QS-E}} \left|\mu_{i-1}(\boldsymbol{q}_{i-1}) - \mu_i(\boldsymbol{q}_i)\right|\\
&\qquad - l^{\text{QS-A}} \frac{\sum_{k\in\mathcal{K}} \mathbb{1}(\omega_{i,k}) \left(\mu_i(\boldsymbol{q}_i) - q_{i,k}\right)^2}{\sum_{k\in\mathcal{K}} \mathbb{1}(\omega_{i,k})}, \quad (12)
\end{aligned}$$

where $\bar{d}_i$ is the average downloading capacity when segment i is being downloaded. For simplification, we will use $\widetilde{U}_i(\boldsymbol{q}_i)$ to denote $\widetilde{U}_i(\boldsymbol{q}_i \mid \boldsymbol{q}_{i-1}, b_{i-1})$ in the rest of this paper.

In contrast to the user's QoE for segment i (as in Section 2.2.3), equation (12) has two major differences. First, we assume that the quality level decision of segment i does not have a significant impact on the value of $\bar{d}_i$.[5] As a result, we can use $\sum_{k\in\mathcal{K}} r_{i,k}(q_{i,k})\beta/\bar{d}_i$ to represent the downloading time of segment i under any quality level choice $q_{i,k}$ for all $k \in \mathcal{K}$ [8], [13]. Second, in (12), we relax the operator $[\cdot]^+$ involved in the rebuffering loss. Intuitively, as we focus on per-segment problems, this relaxation can help with characterizing the impact of the quality level decision of a segment on the user's QoE of the subsequent segments. For example, the rebuffering loss may fail to capture the difference between downloading a segment (with lower quality) for one second and downloading a segment (with higher quality) for five seconds, if neither downloading process induces rebuffering. The two downloading processes, however, may lead to different likelihood of having a rebuffering later due to the resulting different buffer occupancy after the downloading has been accomplished. This difference can be characterized if the operator $[\cdot]^+$ is relaxed.

3.1.2 Per-Segment Optimization Problem

Given $\boldsymbol{q}_{i-1}$ and b_{i-1}, the per-segment problem for segment i is to determine the decision $\boldsymbol{q}_i$ to maximize $\widetilde{U}_i(\boldsymbol{q}_i)$, i.e.,

$$\begin{aligned}
\underset{\boldsymbol{q}_i}{\text{maximize}} \quad & \widetilde{U}_i(\boldsymbol{q}_i)\\
\text{subject to} \quad & q_{i,k} \in \mathcal{Q},\ k\in\mathcal{K}.
\end{aligned} \quad \text{(OPT-SEGMENT-i)}$$

Problem (OPT-SEGMENT-i) does not include constraints (1), (2), and (3) (as in problem (11)) for the following reasons.

5. For example, consider a two-second 4K segment encoded at multiple quality levels, where the available bitrate range is $20-51$ Mbps [28]. Based on the downloading capacity traces from the dataset in [23], the average downloading time of this segment has a range of $2.1-4.4$ seconds. Downloading this segment at bitrates between 20 and 51 Mbps leads to an average downloading capacity difference of 10.3% on average.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

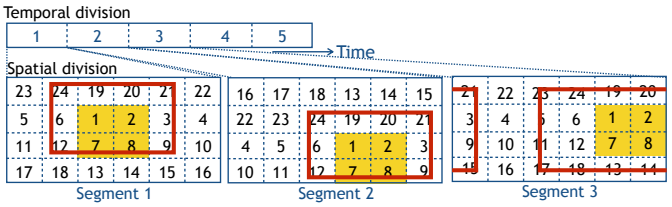


Fig. 5: An example of the user's actual FoV and the reference FoV. Recall that the 360-degree video has no boundary, e.g., for segment 3, tile 3 is on the right-hand side of tile 2.

Constraint (1) is eliminated, because under the per-segment problem, the relationship between the downloading time of different segments does not need to be considered. Constraint (2) is eliminated, because we assume that $\bar{d}_i$ is irrelevant to the quality level decision of segment i. Hence, the downloading time of the segment can be approximated using $\sum_{k\in\mathcal{K}} r_{i,k}\beta/\bar{d}_i$ and is accounted in the objective function $\widetilde{U}_i(\boldsymbol{q}_i)$. In addition, with the set of per-segment problems, at the time that the bitrate decision of segment i is optimized, the buffer occupancy b_{i-1} is directly observed, so the buffer update constraint (3) is not required.

3.2 Performance Metrics

We evaluate the performance of the online algorithm using dynamic regret [20], [21]. The dynamic regret characterizes the difference between the user's QoE under the optimal quality level decision (i.e., the decision after the realization of the user's FoV and downloading capacity) and the user's QoE under the actual quality level decision determined by the proposed algorithm (i.e., the decision before the realization of the user's FoV and downloading capacity). Specifically, let $\boldsymbol{q}_i^o$ denote the quality decision of segment i derived from the online algorithm. Let $\boldsymbol{q}_i^*$ denote the optimal quality decision of segment i obtained by solving problem (OPT-SEGMENT-i) under the realization of the user's FoV and downloading capacity. The dynamic regret is defined as follows:

$$\mathrm{Reg}_I \triangleq \sum_{i\in\mathcal{I}} \left(\widetilde{U}_i(\boldsymbol{q}_i^*) - \widetilde{U}_i(\boldsymbol{q}_i^o)\right). \tag{13}$$

Intuitively, a smaller dynamic regret implies that the performance of the online algorithm is closer to that of the offline optimal performance where the user's FoV and downloading capacity are known beforehand.

3.3 Online Bitrate Selection Algorithm

We design an online algorithm, called OBS algorithm. Intuitively, the algorithm aims at letting the video player choose higher quality for those tiles that are more likely to be viewed by the user, considering the difference between the user's actual FoV and the reference FoV. Fig. 5 shows an example. Due to the tile indexing, the difference between the user's actual FoV (i.e., the solid rectangle) and the reference FoV (i.e., the shaded area) can be represented using the tile indices regardless of the reference FoV. As shown in the figure, the user's actual FoV always covers tiles $1-3$, $6-9$, 12, $19-21$, 24 (i.e., the reference FoV as well as the tiles above and beside the reference FoV), and it sometimes covers tiles 5, 11, $13-15$, $17-18$, 23. Therefore, to enhance the user's QoE, the video player may choose high quality for the former group of tiles, medium quality (if available) for the latter ones, and low quality for the rest.

The main idea of the proposed algorithm is to update the quality decision of a segment (with respect to the decisions of the previous segments) that is likely to enhance the user's QoE based on the realization of the parameters (i.e., the real-time downloading capacity and user's actual FoV) related to the previous segments. In the following, we first introduce an auxiliary set for each segment, which indicates the segments whose parameter realizations (e.g., capacity for downloading the segment, FoV for viewing the segment) are observed after the quality decision of the previous segment is made. Then, we present our proposed OBS algorithm.

3.3.1 Auxiliary Set

Let $\tilde{I}_i$ denote the segment index such that segment i has been viewed by the user during the time period when the quality decisions of segments $\tilde{I}_i$ and $\tilde{I}_i+1$ are made. Since a segment should be downloaded before it is being viewed, the FoV and downloading capacity information of segment i has been observed by the video player when the decision of segment $\tilde{I}_i+1$ is to be made. The auxiliary set $\mathcal{I}_i \subset \mathcal{I}$ for segment $i \in \mathcal{I}\cup\{I+1\}$ is defined as follows:

$$\mathcal{I}_i \triangleq \{i' \in \mathcal{I} \mid \tilde{I}_{i'} = i-1\}, \tag{14}$$

which is the set of segments which have been viewed by the user during the time period when the quality level decisions of segments $i-1$ and i are made. The consideration of $i = I+1$ is used to include the segments whose parameter realizations are observed after the quality level decision of segment I is made.[6] Note that those existing OGD algorithms [20]–[22] can address only the scenario where set $\mathcal{I}_i = \{i-1\}$ for all $i \in \mathcal{I}$. That is, the information of segment $i-1$ can always be observed during the time period when the decisions of segments $i-1$ and i are made. In comparison, our proposed algorithm does not have such a requirement, i.e., set $\mathcal{I}_i$ for each $i \in \mathcal{I}$ can be empty or contain the indices of any segments.

3.3.2 OBS Algorithm

The OBS algorithm is given in Algorithm 1. The algorithm first initializes the following parameters: an initial quality vector $\boldsymbol{q}_0^o$, which can be any quality level in $\mathcal{Q}^K$; a parameter $\alpha > 0$, which corresponds to the positive stepsize in the existing OGD algorithms [20]–[22].

For any segment i, the video player first obtains the auxiliary set $\mathcal{I}_i$, which contains the indices of the segments whose FoV and downloading capacity information is newly observed as defined in (14). Based on set $\mathcal{I}_i$, the quality decision of segment i is made as follows.

If the set $\mathcal{I}_i$ is non-empty, then the quality level decision of segment i will be made based on the quality level decisions of the segments referred in set $\mathcal{I}_i$ and the corresponding downloading capacity and FoV information when each of those segments is downloaded and viewed, respectively. The determination of the quality level decision of segment i contains two steps. (i) Find a quality decision (for each tile)

6. Given the quality level decisions and the parameter realizations of all segments in set $\mathcal{I}$, set $\mathcal{I}_{I+1}$ and vector $\boldsymbol{q}_{I+1}^o$ can be determined.

Algorithm 1 OBS Algorithm

1: **Initialization:** $\boldsymbol{q}_0^o$ and α;
2: **for all** $i \in \mathcal{I}$ **do**
3: Obtain set $\mathcal{I}_i$ according to (14);
4: **if** $\mathcal{I}_i \neq \emptyset$ **then**
5: Compute J_i according to (16);
6: Compute $\boldsymbol{q}_i^c$ according to (15);
7: $\boldsymbol{q}_i^o \leftarrow \arg\min_{\boldsymbol{q}_i \in \mathcal{Q}^K} \|\boldsymbol{q}_i - \boldsymbol{q}_i^c\|$;
8: **else**
9: $\boldsymbol{q}_i^o \leftarrow \boldsymbol{q}_{i-1}^o$;
10: **end if**
11: **end for**

within continuous range $\text{conv}(\mathcal{Q}) = [1, Q]$, where $\text{conv}(\mathcal{Q})$ denotes the convex hull of $\mathcal{Q}$. These quality level decisions are the ones that are likely to enhance the user's QoE based on the decisions and parameter realizations of the previous segments, under the assumption that any value within $\text{conv}(\mathcal{Q})$ can be selected. (ii) Map the quality level decision (for each tile) derived based on the continuous set $\text{conv}(\mathcal{Q})$ to the discrete quality level set $\mathcal{Q}$.

Step (i) (i.e., lines 5 and 6 in Algorithm 1) corresponds to an online version of gradient descent. That is, it aims to update the quality level decision of a segment (with respect to the decisions of the previous segments) in the direction where the user's QoE increases fastest. Let $\boldsymbol{q}_i^c \in \text{conv}(\mathcal{Q})^K = [1, Q]^K$ denote the decision vector (containing the quality level decisions of all K tiles) derived in step (i), where the superscript 'c' refers to the 'continuous range'. The value of $\boldsymbol{q}_i^c$ is derived as follows:

$$\boldsymbol{q}_i^c = \arg \min_{\boldsymbol{q}_i \in \text{conv}(\mathcal{Q})^K} -\nabla \widetilde{U}_{J_i}(\boldsymbol{q}_{J_i}^c)^\top (\boldsymbol{q}_i - \boldsymbol{q}_{J_i}^c) + \frac{1}{2\alpha} \|\boldsymbol{q}_i - \boldsymbol{q}_{J_i}^c\|^2, \tag{15}$$

where $\nabla \widetilde{U}_{J_i}(\boldsymbol{q}_{J_i}^c)$ is the subgradient of $\widetilde{U}_{J_i}(\boldsymbol{q}_{J_i}^c)$, and J_i is defined as

$$J_i \triangleq \arg \min_{j \in \mathcal{I}_i} -\nabla \widetilde{U}_j(\boldsymbol{q}_j^c)^\top (\boldsymbol{q}_j^{c*} - \boldsymbol{q}_j^c), \tag{16}$$

where $\boldsymbol{q}_j^{c*} = \arg\max_{\boldsymbol{q}_j \in \text{conv}(\mathcal{Q})^K} \widetilde{U}_j(\boldsymbol{q}_j)$. Intuitively, J_i is the index of the segment in set $\mathcal{I}_i$ such that the continuous quality level decision of the segment is closest to the optimal quality level decision (within continuous range) of the segment and can lead to the fastest enhancement in terms of the user's QoE. In step (ii), the continuous quality decision $\boldsymbol{q}_i^c$ will be mapped to the discrete quality level set $\mathcal{Q}^K$ by finding the quality level in set $\mathcal{Q}^K$ whose difference with $\boldsymbol{q}_i^c$ is the smallest. This is achieved by line 7 in Algorithm 1.

When set $\mathcal{I}_i$ is empty, no new information can be used for characterizing the difference between the user's actual FoV and the reference FoV. Hence, the quality level decision of segment i is set to be the quality level decision of segment $i-1$, i.e., line 9 in Algorithm 1.

In Algorithm 1, making the quality level decision of segment $i \in \mathcal{I}$ requires a computational complexity of $\mathcal{O}(|\mathcal{I}_i|)$. In terms of the storage, the proposed algorithm requires the video player to store the newly observed information. That is, during the time period when the decisions of segments $i-1$ and i are made, the video player needs to store the information related to the user's downloading capacity and the FoV associated with the segments in set $\mathcal{I}_i$. The downloading capacity $\bar{d}_i$ is a real number, and the FoV information associated with segment i, i.e., $(\omega_{i,k}, k \in \mathcal{K})$, is a vector with K elements. Hence, such information consumes only limited memory for storage.

3.4 Performance Analysis

We proceed to show the performance of Algorithm 1. In the following, we first show some conditions regarding the system setting. Then, we formally state the bound of the dynamic regret of the proposed algorithm.

3.4.1 Conditions

In online convex optimization techniques considering time-varying parameters, without restricting the varying of the parameters, obtaining a bound on dynamic regret is not possible [29]. Here, we impose the following conditions on the varying of the parameters as follows.

Condition 1 (Empty $\mathcal{I}_i$)**.** *There exists a nonnegative value $V_\emptyset$ such that the number of the set $\mathcal{I}_i$ that satisfies $\mathcal{I}_i = \emptyset$ for all $i \in \mathcal{I} \cup \{I+1\}$ is bounded, i.e.,*

$$\sum_{i \in \mathcal{I} \cup \{I+1\}} \mathbb{1}(\mathcal{I}_i = \emptyset) \leq V_\emptyset, \tag{17}$$

where $\mathbb{1}(\cdot)$ is an indicator function, i.e., $\mathbb{1}(\mathcal{I}_i = \emptyset) = 1$ if $\mathcal{I}_i = \emptyset$, and is equal to zero otherwise.

In Condition 1, inequality (17) implies that the number of the quality decisions that are not updated based on (15) is bounded. To define the second condition, let $J_i^\dagger$ denote the index of the segment such that the quality level of segment J_i is made based on it, i.e., $J_i^\dagger = J_{i'}$ with $i' = J_i$.

Condition 2 (Time-Varying Parameters)**.** *There exists a nonnegative value V_q such that the following inequalities hold for any number of segments I:*

$$\sum_{i \in \mathcal{I} \cup \{I+1\}} |\mathcal{I}_i| \|\boldsymbol{q}_{J_i}^{c*} - \boldsymbol{q}_{J_i^\dagger}^{c*}\| \leq V_q, \tag{18}$$

where $|\mathcal{I}_i|$ is the cardinality of set $\mathcal{I}_i$.

Recall that $\boldsymbol{q}_i^{c*} = \arg\max_{\boldsymbol{q}_i \in \text{conv}(\mathcal{Q})^K} \widetilde{U}_i(\boldsymbol{q}_i)$. In Condition 2, inequality (18) ensures that the variations of the parameters are relatively small, such that the changes among the optimal solutions to the per-segment problems are bounded. For example, this condition holds when the downloading capacity remains relatively stable, or it holds when the user sets a high priority on avoiding quality switch.

3.4.2 Dynamic Regret

We show that the dynamic regret is upper-bounded. The corresponding proof is given in the Appendix.

Theorem 1. *Under Condition 1, the dynamic regret of Algorithm 1 is upper-bounded by*

$$Reg_I \leq \frac{R^2(1+V_\emptyset)}{2\alpha} + \frac{RV_q}{\alpha} + \frac{\alpha \overline{U}^2 I}{2} + \mathbb{1}(\widetilde{\mathcal{I}} \neq \emptyset) \frac{KQ}{d^{\min}} \left(\frac{3R^2}{2\alpha} + \frac{\alpha}{2} \overline{U}^2 \right) + IB. \tag{19}$$

The constant $\overline{U}$ is the bound of the norm of the subgradient $\nabla\widetilde{U}_i(\boldsymbol{r}_i)$, i.e., $\|\nabla\widetilde{U}_i(\boldsymbol{r}_i)\| \leq \overline{U}$ for all $i \in \mathcal{I}$ and $\boldsymbol{q}_i \in \mathcal{Q}^K$. The constant R is the radius of the convex hull $\mathrm{conv}(\mathcal{Q})^K$, i.e., $\|\boldsymbol{q}_i - \boldsymbol{q}_i'\| \leq R$, for all $\boldsymbol{q}_i, \boldsymbol{q}_i' \in \mathrm{conv}(\mathcal{Q})^K$. The set $\widetilde{\mathcal{I}}$ is a subset of set $\mathcal{I}$, which is defined as $\widetilde{\mathcal{I}} \triangleq \{j \in \mathcal{I} \mid j \notin \mathcal{I}_i, i \in \mathcal{I} \cup \{I+1\}\}$. The constant B is the bound of the change of the user's QoE resulting from the mapping in line 7 in Algorithm 1. That is, $|\widetilde{U}_i(\boldsymbol{q}_i) - \widetilde{U}_i(\boldsymbol{q}_i')| \leq B$ for $\boldsymbol{q}_i, \boldsymbol{q}_i' \in \mathcal{Q}^K$ and $i \in \mathcal{I}$, where $q_{i,k}$ and $q_{i,k}'$ are adjacent quality levels from set $\mathcal{Q}$ for all $k \in \mathcal{K}$.

Based on Theorem 1, the dynamic regret on the long-term average is bounded as follows.

Corollary 1 (Dynamic Regret). *Under Conditions 1 and 2, by setting $\alpha = \alpha_0 I^{-1/\gamma}$ for any $\gamma \in (1, \infty)$, the dynamic regret of Algorithm 1 on the long-term average satisfies $\lim_{I\to\infty} Reg_I/I \leq B$.*

The proof of Corollary 1 is provided in Appendix B. Corollary 1 implies that as the number of segments increases, the performance of the proposed online algorithm approaches the performance under the optimal solutions to the per-segment problems (OPT-SEGMENT-i) for $i \in \mathcal{I}$ asymptotically, with a bounded error B. The bounded error B is due to the fact that the quality level set is discrete such that decreasing the quality level of a tile by one unit leads to a noncontinuous drop in terms of the user's QoE. In other words, if the quality level set $\mathcal{Q}$ is large such that the adjacent quality levels in the set do not lead to remarkable changes in the user's QoE (i.e., B is around zero), then the difference between the performance of the proposed algorithm and the optimal performance approaches zero.

3.5 Algorithm Modification

Theorem 1 and Corollary 1 show the performance of Algorithm 1 on the long-term average. That is, as the number of segments increases, the performance of the proposed algorithm gradually approaches the offline optimal performance with a bounded error. When the number of segments that have been played back is small, the proposed algorithm may still be in the learning phase and may not have approached the offline optimal performance. In this case, there may exist unnecessary rebuffering and quality switch. To address this issue, for any quality level decision output of the algorithm (i.e., $\boldsymbol{q}_i^o$), we compute a modified quality level decision $\boldsymbol{q}_i^m$ to reduce the rebuffering and quality switch experienced by the user. Note that $\boldsymbol{q}_i^m$ is the actual quality level decision used for scheduling the video tiles. It does not affect the operation of Algorithm 1. Hence, Theorem 1 and Corollary 1 on dynamic regret, i.e., $\mathrm{Reg}_I \triangleq \sum_{i\in\mathcal{I}}(\widetilde{U}_i(\boldsymbol{q}_i^*) - \widetilde{U}_i(\boldsymbol{q}_i^o))$, still hold. When the number of segments that have been played back is large enough such that $\boldsymbol{q}_i^o$ converges to a particular quality level decision vector, the modification makes no difference in the quality level decision. In this case, we have $\boldsymbol{q}_i^m = \boldsymbol{q}_i^o$.

We compute $\boldsymbol{q}_i^m$ using a heuristic idea as follows. We first define a vector $\tilde{\boldsymbol{q}}_i^o \in \mathcal{Q}^K$, which is computed as $\tilde{q}_{i,k}^o = \max\{\min\{q_{i,k}^o, q_{i-1,k}^m + 1\}, q_{i-1,k}^m - 1\}$. This means that $\tilde{q}_{i,k}^o$ is at most one level higher than the previous quality level $q_{i-1,k}^m$, and it is at most one level lower than the previous quality level. Intuitively, this can reduce the chance of inter-segment quality switch in the future. Then, we compute $\boldsymbol{q}_i^m$ as the optimal solution to the following problem:

$$\underset{\boldsymbol{q}_i^m, \boldsymbol{z}}{\text{minimize}} \ \|\boldsymbol{q}_i^m - \tilde{\boldsymbol{q}}_i^o\| \tag{20a}$$

$$\text{subject to} \ \ q_{i,k}^m = \sum_{q\in\mathcal{Q}} z_{k,q} q, \ k \in \mathcal{K} \tag{20b}$$

$$\sum_{k\in\mathcal{K}} r_{i,k}(q_{i,k}^m)\beta/\bar{d}_{i-1} \leq b_{i-1}, \tag{20c}$$

$$\sum_{q\in\mathcal{Q}} \mathbb{1}(\sum_{k\in\mathcal{K}} z_{k,q}) \leq \zeta, \tag{20d}$$

$$q_{i,k}^m \in \mathcal{Q}, \ k \in \mathcal{K}, \tag{20e}$$

$$z_{k,q} \in \{0,1\}, \ k \in \mathcal{K}, q \in \mathcal{Q}. \tag{20f}$$

In problem (20), we introduce an auxiliary variable $\boldsymbol{z} = (z_{k,q}, k \in \mathcal{K}, q \in \mathcal{Q})$. For this auxiliary variable, $z_{k,q} = 1$ indicates that the k^{th} tile is at quality level q; $z_{k,q} = 0$ otherwise. Hence, we have $q_{i,k}^m = \sum_{q\in\mathcal{Q}} z_{k,q} q$ for $k \in \mathcal{K}$, i.e., constraint (20b). Intuitively, problem (20) aims to find a quality level decision $\boldsymbol{q}_i^m$ that is close to $\tilde{\boldsymbol{q}}_i^o$. The quality level decision $\boldsymbol{q}_i^m$ should have a low chance of rebuffering (constraint (20c)) and a low chance of intra-segment quality switch. The latter is captured by constraint (20d), i.e., the total number of non-identical quality levels of the tiles should be less than ζ. We empirically find that $\zeta = 3$ leads to a good performance.

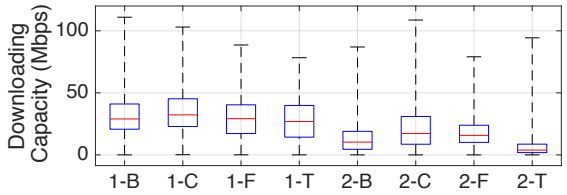


Fig. 6: Downloading capacity of the traces in datasets. In the x-axis, numbers 1 and 2 refer to the datasets in [23] and [24], respectively. Letters 'B', 'C', 'F', and 'T' refer to bus, car, foot, and train, respectively.

TABLE 1: Description of the Videos

Video	Resolution	Dur.	Video	Resolution	Dur.
360pa.	3840×1080	377 s	luther	2880×1440	265 s
cin.	2560×1280	60 s	smart	4096×2048	127 s
DB	4096×1024	238 s	vaude	4096×2048	145 s
jaunt	2304×1152	172 s	war	4096×1152	205 s

4 Performance Evaluation

In this section, we perform trace-driven simulations to evaluate the performance of our proposed OBS360 algorithm. We use three open datasets from [23]–[25] to simulate the 360 degree video streaming scenarios. In our simulations, each segment is divided into 24 tiles in a four by six grid. Each user can see 110 degree in horizontal direction and 90 degree in vertical direction, as in [8], [14]. We consider a setting where the videos are encoded and projected using equirectangular projection, as in [5], [16]. We set the initial buffer occupancy b^{INI} to two seconds and segment length β to one second. The simulation results are shown using boxplot [30]. Specifically, for each box, the central red mark shows the median, and the bottom and top edges indicate the 25th and 75th percentiles, respectively. The lower and upper bounds of the whiskers show the minimum and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

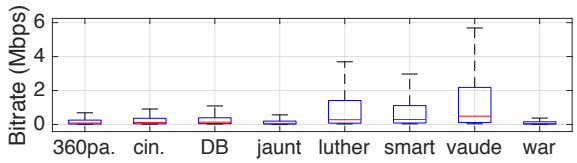


Fig. 7: Available bitrate ranges of different videos, where the x-axis corresponds to the title abbreviations of the videos.

maximum without considering the outliers, respectively. The red '+' marks show the outliers.

In the following, we first introduce the open datasets from [23]–[25]. Then, we present the simulation results.

4.1 Datasets

We use the datasets from [23] and [24] to simulate the users' downloading capacities. The dataset in [23] contains 40 bandwidth measurement traces in 4G networks in the city of Ghent, Belgium. The dataset in [24] contains 135 traces in 4G networks from Irish mobile operators. Both datasets are collected on various transportation modes, such as by bus, car, train, or on foot. We choose these two datasets because the data are collected in 4G networks and contain traces with different average downloading capacities. Fig. 6 shows the boxplot of the downloading capacity samples of the datasets in [23] and [24] under different transportation modes. In Fig. 6, the traces in the dataset from [23] (corresponding to boxes 1-B, 1-C, 1-F, and 1-T) have a higher average downloading capacity than the traces in the dataset from [24] (corresponding to boxes 2-B, 2-C, 2-F, and 2-T).

We use the dataset in [25] to simulate the user's FoV. This dataset contains the traces of eight videos. The video information is provided in Table 1, where 'dur.' is the short-form for 'duration'. An advantage of this dataset is that it contains the recommended viewing traces of the videos, which are marked by professional filmmakers. Consequently, we can compare the performance of our proposed algorithm with that of directly optimizing the quality levels based on the recommended viewing traces. For each video, there is one recommended viewing trace and 20 actual viewing traces from 20 users. In the dataset, either the recommended trace or each user's trace of watching a video is represented by a viewing degree trace $((p_1, y_1), (p_2, y_2), \ldots, (p_I, y_I))$, where $p_i \in [-90°, 90°]$ and $y_i \in [-180°, 180°]$ are the pitch (vertical degree) and yaw (horizontal degree) of the corresponding viewport of segment $i = 1, 2, \ldots, I$, respectively. To perform trace-driven simulations, we use FFmpeg [31] to encode each video into eight quality levels $\mathcal{Q} = \{1, 2, \ldots, 8\}$. We choose the value of the constant rate factor from set $\{6, 12, 18, 23, 28, 33, 38, 43\}$. Note that a lower quality level corresponds to a higher constant rate factor. The videos encoded at consecutive quality levels have a bitrate ratio of 1:1.7 on average. We use FFmpeg to divide the video at each quality level into segments with a duration of one second. We further divide each segment into $4 \times 6 = 24$ tiles. Fig. 7 shows the available bitrate ranges of the videos encoded at different quality levels. As shown in the figure, the bitrates of videos '360pa.', 'jaunt', and 'war' are relatively low, while those of videos 'luther', 'smart', and 'vaude' are relatively high.

4.2 OBS360 and Benchmark Methods

We consider the following performance metrics: viewing quality level, rebuffering, as well as inter-segment and intra-segment quality switch. We evaluate the performance of the following algorithms. (i) 'DC', which is a greedy algorithm that optimizes the quality level based on the recommended FoV (or director's cut) provided by the dataset in [25]. (ii) 'BAS360' in [16], which aims to minimize the bandwidth waste, i.e., the bandwidth that can be further utilized without inducing rebuffering and the bandwidth used for downloading the unviewed tiles. This bandwidth waste is minimized according to the user's predicted FoV derived based on the historical FoV of other users. (iii) 'Flare' in [5], which aims to maximize the user's QoE according to the predicted FoV derived based on the user's head movement. (iv) 'OBS', which is our proposed algorithm. For the proposed 'OBS' algorithm, the reference FoV can be the recommended FoV provided by [25], the predicted FoV based on the historical FoV of other users [16], and the predicted FoV based on the user's head movement [5]. The corresponding algorithms with the reference FoVs are denoted by 'OBS-R', 'OBS-H', and 'OBS-M', respectively. Note that we choose 'DC', 'BAS360', and 'Flare' for comparison, because they correspond to content based, fixation clustering based, and head movement based algorithms, respectively.

In these simulations, we assume that all the tiles are downloaded in the sequence of their indices (as we have assumed in Section 2.2.1) for all algorithms. We leave the optimization of the downloading sequence of the tiles as future work. For each simulation, we consider 20 users, because the dataset in [25] contains the FoV traces collected from a total of 20 participants. Each user corresponds to a FoV trace from the dataset in [25] and a randomly selected downloading capacity trace from the datasets in [23], [24]. We set parameters $l^{\text{RB}} = 1$, $l^{\text{QS-E}} = 0.5$, and $l^{\text{QS-A}} = 0.5$, as in other existing works such as [8], [32]. Furthermore, according to [8], we consider a fixed decoding time of 0.6 second for each segment, which is assumed to be independent of the bitrates of the tiles of the segment.[7]

4.2.1 Impact of Reference FoV

In this section, we evaluate the performance of the existing methods ('DC', 'BAS360', and 'Flare') as well as our proposed algorithm under different reference FoVs ('OBS-R', 'OBS-H', and 'OBS-M'). Note that these methods can be grouped into pairs. That is, 'DC' and 'OBS-R' are based on recommended FoV, 'BAS360' and 'OBS-H' are based on the predicted FoV according to the historical FoV of other users, and 'Flare' and 'OBS-M' are based on the predicted FoV according to the user's head movement.

From Figs. 8 and 9, we have the following observations. First, for each pair of methods, our proposed algorithm can improve the viewing quality level as well as reduce the inter-segment and intra-segment quality switch when comparing with the other method in the pair. Such performance enhancement is more significant for the pair of methods 'DC' and 'OBS-R' when compared with other pairs. This is because the recommended FoV is more different

7. As mentioned in [8], the decoding complexity is not significantly affected by the bitrate of the segment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

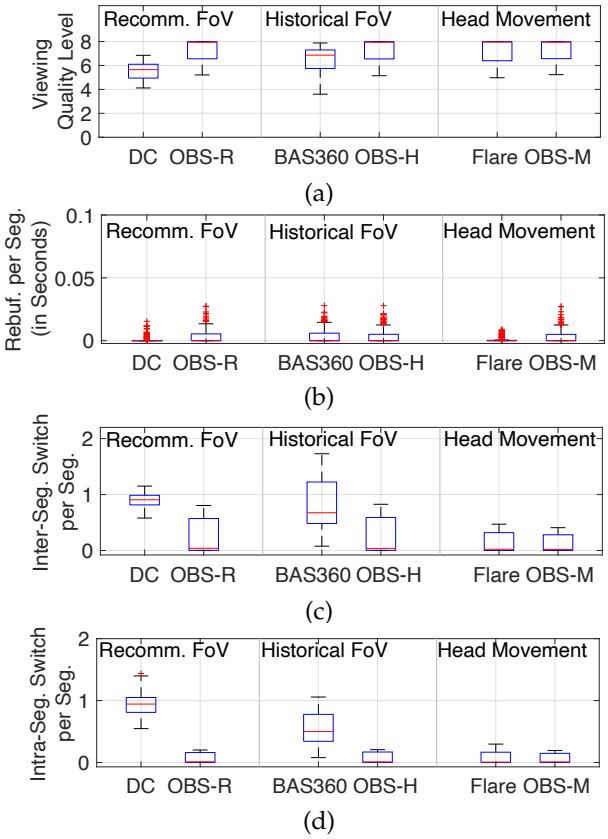


Fig. 8: Performance under different reference FoVs (with the dataset in [23]): (a) viewing quality level; (b) rebuffering; (c) inter-segment quality switch; (d) intra-segment quality switch.

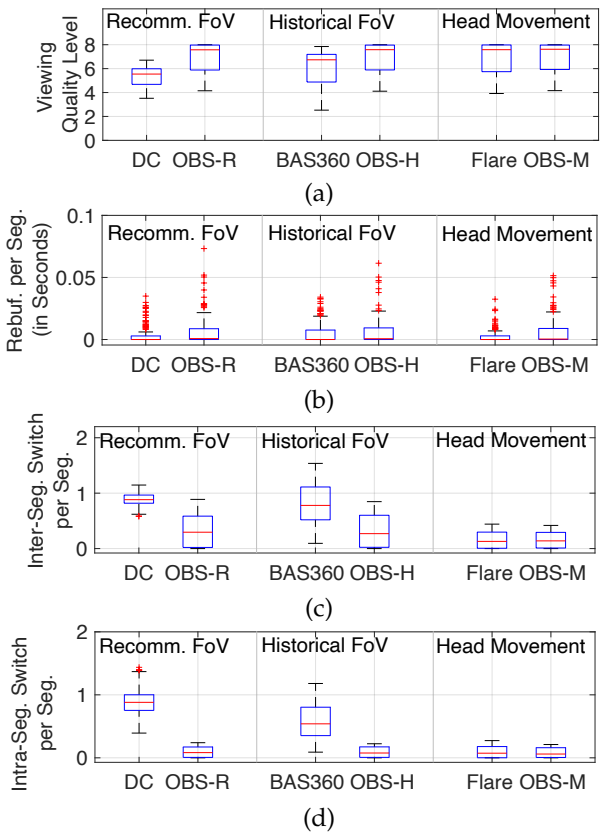


Fig. 9: Performance under different reference FoVs (with the dataset in [24]): (a) viewing quality level; (b) rebuffering; (c) inter-segment quality switch; (d) intra-segment quality switch.

from the user's actual FoV than the other predicted FoVs are. Due to this, method 'DC' based on the recommended FoV provides more potential for the proposed algorithm to characterize the difference between the reference FoV (i.e., the recommended FoV for 'OBS-R') and the user's actual FoV. Hence, it provides more potential for the proposed algorithm to enhance the system performance. Second, 'OBS-R', 'OBS-H', and'OBS-M' achieve similar performance in terms of the viewing quality level, rebuffering, as well as intra-segment quality switch, as shown in Figs. 8 and 9. This implies that the choice of the reference FoV does not have significant impact on the performance of the proposed algorithm. In other words, as long as the reference FoV is properly selected such that we can observe a particular pattern in terms of the difference between the reference FoV and the user's actual FoV,[8] the proposed algorithm can achieve a good performance by selecting the quality levels of the tiles according to the difference.

4.2.2 *Impact of Video*

In this section, we evaluate the performance of the algorithms under different videos. As shown in Fig. 7, different videos have different available bitrate ranges. Based on the bitrate ranges, we classify the videos into three groups: {360pa., jaunt, war}, {cin., DB}, {luther, smart, vaude}. The available bitrates of the videos in the first group are relatively low, and those in the third group are high.

Figs. 10 and 11 show the algorithm performance under different videos with the datasets in [23] and [24], respectively. Note that we omit the performance of 'OBS-R' and 'OBS-H' for presentation simplicity. From Fig. 10, we have the following observations. For videos in {360pa., jaunt, war} and {cin., DB}, both 'Flare' and 'OBS-M' can achieve the maximum viewing quality level (i.e., level 8) and have low rebuffering and quality switch. This is because the downloading capacity obtained from the dataset in [23] is sufficient for downloading most of the tiles at the maximum quality level. For videos in {luther, smart, vaude}, these videos require higher bitrates to achieve each particular quality level than the other videos do. Hence, it is more challenging to provide video streaming services with high viewing quality level as well as low rebuferring and quality switch. For these videos in {luther, smart, vaude}, the proposed algorithm 'OBS-M' improves the viewing quality level by $3.0\% - 21.6\%$ as well as reduces the average intra-segment switch by at least 21.5% when compared with the other methods. The proposed algorithm 'OBS-M' achieves similar intra-segment quality switch as 'Flare'.

In Fig. 11, as the average downloading capacity of the traces from the dataset in [24] is low than that from the dataset in [23], the viewing quality levels of all the methods in Fig. 11 (a) are lower than those in Fig. 10 (a). Meanwhile, the rebuffering of all the methods in Fig. 11 (b) are

8. Recommended FoV and predicted FoV are good examples of reference FoV, while a randomly generated FoV may not. This is because the difference between the randomly generated FoV and the user's actual FoV can be random, under which the random difference may not provide any intuition on how to improve the system performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2020.3038710, IEEE Transactions on Mobile Computing

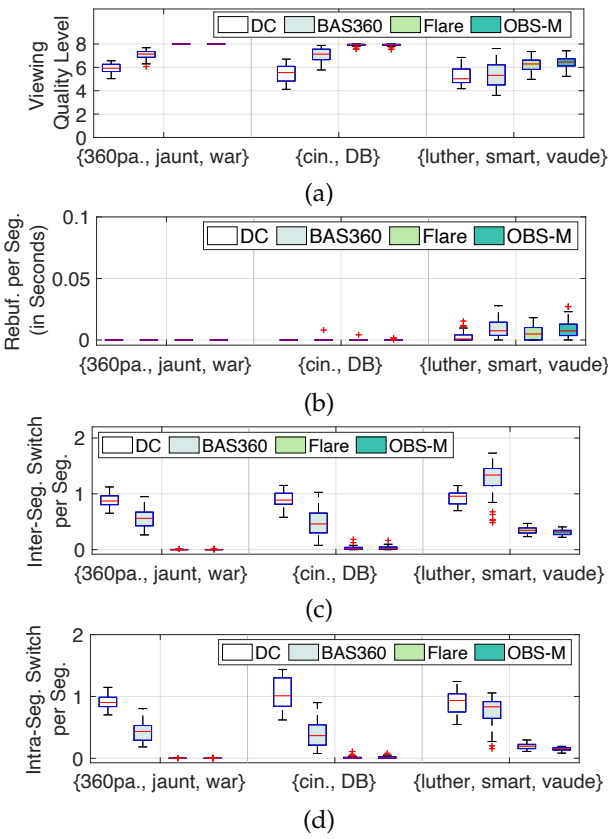


Fig. 10: Performance under different videos (with the dataset in [23]): (a) viewing quality level; (b) rebuffering; (c) inter-segment quality switch; (d) intra-segment quality switch.

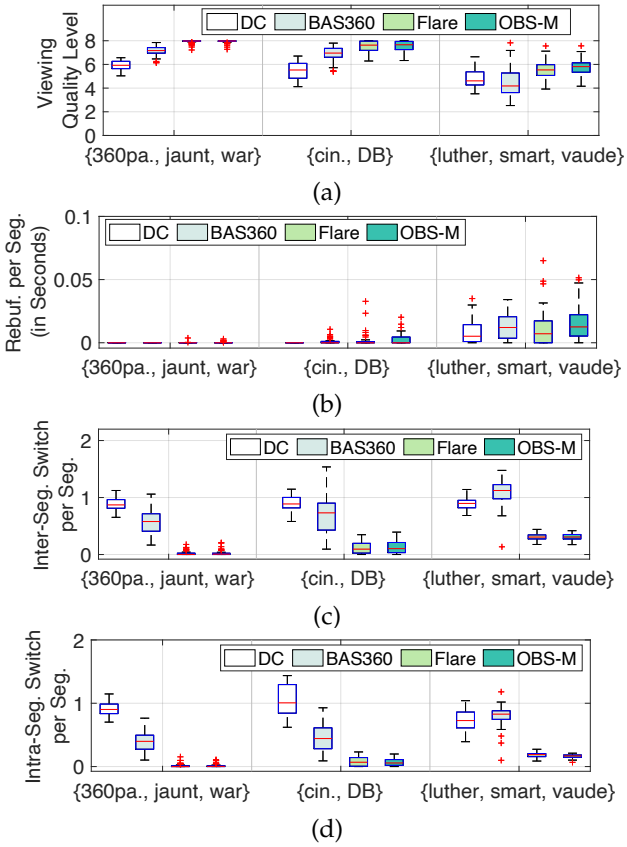


Fig. 11: Performance under different videos (with the dataset in [24]): (a) viewing quality level; (b) rebuffering; (c) inter-segment quality switch; (d) intra-segment quality switch.

higher than those in Fig. 10 (b). For these videos in {luther, smart, vaude}, the proposed algorithm 'OBS-M' improves the viewing quality level by $4.2\% - 29.4\%$ as well as reduces the average intra-segment quality switch by at least 12.4% when compared with the other methods.

5 Conclusion

In this paper, we proposed an online bitrate selection algorithm that can enhance the user's QoE by learning the user heterogeneity in terms of the user-specific FoV and downloading capacity in real time. We proved that as the number of segments increases, the performance of the algorithm approaches the offline optimal performance asymptotically with a bounded error. We performed trace-driven simulations with real-world datasets on the user's FoV and downloading capacities. The results show that our proposed algorithm can enhance the user's viewing quality level as well as reduce the inter-segment and intra-segment quality switch when compared with the existing methods.

The results in this paper can be extended in the following directions. First, it is interesting to take into account the optimization of the downloading and decoding sequence of the tiles, e.g., downloading and decoding the tiles which have higher probability of being viewed earlier than those with lower probability. This may further reduce the chance of rebuffering, as the users care about only whether those tiles being viewed are downloaded and decoded on time or not. Second, it is interesting to incorporate segment replacement, i.e., replacing the low bitrate tiles in user's buffer with high bitrate ones whenever the bandwidth allows. For example, if the bandwidth increases significantly during the downloading of the tiles of a segment, the video player may choose to download some of the tiles again in higher bitrates to potentially improve the viewing quality of the user. Third, it will be helpful to evaluate the performance of the proposed algorithm using a testbed to understand the performance under various videos, users, and network connection settings. With the testbed, many practical issues such as synchronization and rendering are needed to be addressed. Fourth, based on the testbed, it is interesting to evaluate the performance of the proposed algorithm with human participants. Such an evaluation may be able to characterize some aspects that are difficult to be captured by trace-driven simulations. For example, humans may have different sensitivity to the quality switch happening at different playback time.

References

[1] "Sony PlayStation VR," https://www.playstation.com/en-au/explore/playstation-vr/, Accessed on Oct. 29, 2020.

[2] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *Proc. IEEE Int'l Symp. on Multimedia (ISM)*, San Jose, CA, Dec. 2016.

[3] L. D'Acunto, J. van den Berg, E. Thomas, and O. Niamut, "Using MPEG DASH SRD for zoomable and navigable video," in *Proc. ACM Int'l Conf. on Multimedia Systems (MMSys)*, Klagenfurt, Austria, May 2016.

[4] T. Ballard, C. Griwodz, R. Steinmetz, and A. Rizk, "RATS: Adaptive 360-degree live streaming," in *Proc. ACM Int'l Conf. on Multimedia Systems (MMSys)*, Amherst, MA, Jun. 2019.

[5] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. ACM MobiCom*, New Delhi, India, Oct. 2018.

[6] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019.

[7] H. Pang, C. Zhang, F. Wang, J. Liu, and L. Sun, "Towards low latency multi-viewpoint 360° interactive video: A multimodal deep reinforcement learning approach," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019.

[8] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proc. ACM MobiSys*, Munich, Germany, Jun. 2018.

[9] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "An optimal tile-based approach for viewport-adaptive 360-degree video streaming," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 29–42, Mar. 2019.

[10] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 43–57, Mar. 2019.

[11] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Computational Visual Media*, vol. 5, no. 2, pp. 117–150, Jun. 2014.

[12] W. Shen, L. Ding, G. Zhai, Y. Cui, and Z. Gao, "A QoE-oriented saliency-aware approach for 360-degree video transmission," in *Proc. IEEE Int'l Conf. on Visual Communications and Image Processing (VCIP)*, Sydney, Australia, Dec. 2019.

[13] Z. Jiang, X. Zhang, W. Huang, H. Chen, Y. Xu, J. N. Hwang, Z. Ma, and J. Sun, "A hierarchical buffer management approach to rate adaptation for 360-degree video streaming," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2157–2170, Feb. 2020.

[14] H. Yuan, S. Zhao, J. Hou, X. Wei, and S. Kwong, "Spatial and temporal consistency-aware dynamic adaptive streaming for 360-degree videos," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 1, pp. 177–193, Jan. 2020.

[15] L. Xie, X. Zhang, and Z. Guo, "CLS: A cross-user learning based system for improving QoE in 360-degree video adaptive streaming," in *Proc. ACM Int'l Conf. on Multimedia (MM)*, Seoul, Republic of Korea, Oct. 2018.

[16] M. Xiao, C. Zhou, V. Swaminathan, Y. Liu, and S. Chen, "BAS-360°: Exploring spatial and temporal adaptability in 360-degree videos over HTTP/2," in *Proc. IEEE INFOCOM*, Honolulu, HI, Apr. 2018.

[17] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proc.ACM Int'l Conf. on Multimedia*, Seoul, Republic of Korea, Oct. 2018.

[18] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proc. of ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Taipei, Taiwan, Jun. 2017.

[19] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *Proc. ACM SIGCOMM*, Beijing, China, Aug. 2019.

[20] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *Proc. IEEE CDC*, Las Vegas, NV, Dec. 2016.

[21] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 24, pp. 6350–6364, Dec. 2017.

[22] S. Paternain and A. Ribeiro, "Online learning of feasible strategies in unknown environments," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2807–2822, Nov. 2017.

[23] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.

[24] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4G LTE dataset with channel and context metrics," in *Proc. ACM Int'l Conf. on Multimedia Systems (MMSys)*, Amsterdam, Netherlands, Jun. 2018.

[25] S. Knorr, C. Ozcinar, C. O. Fearghail, and A. Smolic, "Director's cut: A combined dataset for visual attention analysis in cinematic VR content," in *Proc. ACM SIGGRAPH*, London, United Kingdom, Dec. 2018.

[26] FFmpeg, "H.264 video encoding guide," https://trac.ffmpeg.org/wiki/Encode/H.264, Accessed on Oct. 29, 2020.

[27] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM*, Chicago, IL, Aug. 2014.

[28] YouTube Help, "Choose live encoder settings, bitrates, and resolutions," https://support.google.com/youtube/answer/2853702?hl=en, Accessed on Oct. 29, 2020.

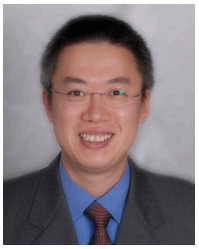
[29] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Proc. Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, San Diego, CA, May 2015.

[30] A. Biguri, "Multiple_boxplot," https://www.mathworks.com/matlabcentral/fileexchange/47233-multiple_boxplot-m?s_tid=srchtitle, Accessed on Oct. 29, 2020.

[31] FFmpeg, "About FFmpeg," https://ffmpeg.org, Accessed on Oct. 29, 2020.

[32] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, London, United Kingdom, Aug. 2015.

Ming Tang (S'16, M'18) is a postdoctoral research fellow in the Department of Electrical and Computer Engineering of the University of British Columbia, Vancouver, BC, Canada. She received her Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China, in 2018. Her research interests include mobile networking and network economics.



Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, BC, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microchip Technology Inc.). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to wireless networks, smart grid, mobile edge computing, and Internet of Things. Currently, Dr. Wong is an Executive Editorial Committee Member of *IEEE Transactions on Wireless Communications*, an Area Editor of *IEEE Transactions on Communications* and *IEEE Open Journal of the Communications Society*, and an Associate Editor of *IEEE Transactions on Mobile Computing*. He is a Technical Program Co-chair of the *IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. He has served as a Guest Editor of *IEEE Journal on Selected Areas in Communications* and *IEEE Wireless Communications*. He has also served on the editorial boards of *IEEE Transactions on Vehicular Technology* and *Journal of Communications and Networks*. He was a Tutorial Co-Chair of *IEEE Globecom*'18, a Technical Program Co-chair of *IEEE SmartGridComm*'14, as well as a Symposium Co-chair of *IEEE ICC*'18, *IEEE SmartGridComm* ('13, '17) and *IEEE Globecom*'13. He is the Chair of the IEEE Vancouver Joint Communications Chapter and has served as the Chair of the IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications. He is an IEEE Communications Society Distinguished Lecturer (2019 - 2020).